ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Automatická detekce dopravních objektů na pozemních komunikacích pro pasportizaci a navigaci

Vypracoval: Lukáš Picek
Vedoucí práce: Ing. Ivan Pirner

Květen 2014

Čestné prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou nazávěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a pouze za použití literatury a pramenů, jejichž úplný seznam je její součástí. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

Datum: 16.5. 2014

. . . . . .

podpis

Anotace

Cílem této práce bylo seznámit se s různými metodami video trackingu, neboli automatického sledováním jednoho či více pohybujících se předmětů za pomocí videokamery a následném vytvoření konkrétní aplikace, která bude tento problém řešit. Hlavním úkolem tedy bylo prostudovat známé metody, a dle získaných znalostí vybrat jednu, kterou poté implementuji v aplikaci. Tato aplikace je za pomoci již dříve získaných informací schopna říci, zdali se na následujícím respektive předchozím snímku, v porovnání s konkrétním snímkem opatřeným číselnou hodnotou, nalézá stejná dopravní značka. Pokud dojde k zjištění, že ano, je zdrojový soubor upraven. Konečný výsledek této bakalářské práce bude použit jako součást upravující problémové korespondence anotovaných objektů v sousedních snímcích, nutné pro trénování klasifikátoru, u projektu DOPANAR, který se zabývá Automatickou detekcí dopravních objektů na pozemních komunikacích pro aktualizaci navigačních prostředků, asistenci řidiče a pasportizaci.

Klíčová slova: Počítačové vidění, významný bod, SURF, SIFT, matching, Qt, C++, OpenCV, dopravní značení, aplikace, tracking

Abstract

The main aim of this bachelor´s thesis was to understand the various methods of video tracking, in other words the automatic monitoring of one or more moving objects, with the use of camcorder and the subsequent creation of a specific application which will solve this problem. The main objective was to study the known methods and according to acquired knowledge choose the only one method, which will be implemented in this application. Using the previously acquired information, the application is able to recognize, if the following or previous shot shows the same traffic sign in comparison with a concrete shot equipped with a numeric value. If there is a foundation that it works, the file will be modified. The final result of this work will be used as part which modifies problem correspondence of annotated objects at adjacent frame, necessary for classifier training of DOPANAR project. This project deals with automatic detection of traffic signs on roads for the update of navigational means, driver assistance and information retrieval.

Keywords: computer vision, keypoint, SURF, SIFT, matching, Qt, C++, OpenCV, traffic signs, app , tracking

Obsah

Kapitola 1

Úvod

Díky urychlování vývoje na poli mikroelektroniky a analýze zpracování obrazu, se záznam videa v poslední době stal až neuvěřitelně jednoduchou záležitostí. Spolu s dnešními moderními přístroji, které jsou schopny snímat a zároveň porozumět prostředí v jakém se vyskytují, se nám otevřelo mnoho možností pro rozvoj inteligentnějších aplikací. Tyto aplikace mohou být využity v moderních vědních odvětvích jako jsou vzdálený dohled, umělá inteligence, robotika, přirozená interakce člověk-stroj apod.

Nejdůležitější vlastností, kterou tyto přístroje potřebují ke snímání a reakci na okolní prostředí je schopnost detekovat a sledovat potřebné objekty. Proces pro odhalování lokace jednoho nebo více předmětů za pomocí kamery, se nazývá video tracking. A právě o problémech, možnostech a implementaci video trackingu pojednává následující práce.

Kapitola 2

Úvod do video trackingu

2.1 Tracking

Jednou z důležitých součástí počítačového vidění, jakožto oblasti kybernetiky, je video tracking. Video tracking se především zabývá automatickým sledováním zvolených pohybujících se prvků po určité dráze ve videu. Otázka trackování, neboli sledování se dá rozložit na dva základní problémy.

1. **Motion problem** se týká výhradně predikování lokace, ve které se zkoumaný objekt bude s největší pravděpodobností nacházet v dalším snímku videa.

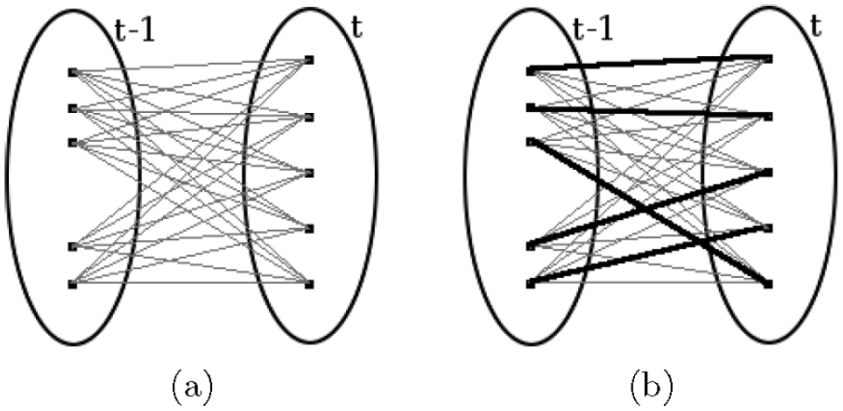
2. **Matching problem,** neboli detekce a lokalizace se snaží rozpoznat již zmíněný zkoumaný objekt v rámci vybrané části následujícího snímku.

2.2 Motion problem

Jak již bylo nastíněno, hlavním úkolem "Motion problému"je predikce lokace, ve které se na následujícím snímku bude sledovaný objekt nacházet. Tato predikce vychází zejména ze znalostí polohy a velikosti objektu na předchozím snímku. Bohužel získané znalosti nejsou vždy dostupné a překvapivě nemusí být ani spolehlivé, popřípadě časově nezávislé. Z důvodu možné chybovosti jsem se rozhodl tento krok přeskočit, nicméně nejpoužívanější metodou pro predikci v oblasti řízení a počítačového vidění je Kalmanův Filtr, který je konkrétně popsát zde [10]. Nesmím však opomenout další významnou metodu pojmenovanou Particle Filtr a zveřejněnou H. Tanizakim v [11].

2.3 Matching problem

Matching problem se zabývá výhradně porovnáváním elementů reprezentujících sledovaný objekt. Tyto elementy mohou být různé a konkrétněji jsou popsány ve 2.4. Pokud je použita reprezentace pomocí bodů, je pro všechny body hledán kandidát, který se po splnění určených podmínek stane korespondujícím s referenčním bodem, jak můžeme vidět na Obr. 2.1 . Pokud dojde k určitému počtu korespondujících bodů můžeme říci, že se na zkoumaném snímku nachází hledaný předmět. Dále budou vysvětleny principy dvou nejpoužívanějších metod pro získávání korespondencí mezi dvěma množinami bodů.



Obrázek 2.1: Zobrazení Matchingu v pseudosnímku, kde a) zobrazuje veškeré možné asociace, z kterých jsou následně vybrány korespondující body. Korespondence jsou zobrazené a zvýrazněné na obrázku b).[13]

2.3.1 Brute force matching

Princip Brute force matchingu spočívá v postupném nalezení dvou nejlepších shod pro všechny významné body v referenčním snímku s obrázkem porovnávaným a naopak. Tímto získáme pro každý významý bod dva nejpodobnější body z druhého snímku. Pokud je rozdíl mezi hodnotami deskriptorů dostatečně velký, je bod s nejvíce shodami vyhodnocen jako stejný. V opačném případě, kdy je rozdíl hodnot nepatrný, jsou obě shody vyřazeny a tím pádem u tohoto bodu nedojde ke shodě.

2.3.2 Flann Matching

Problém procházení nejbližších sousedů je jedním z hlavních problémů v oblastech počítačového rozpoznávání, komprese dat, klasifikaci a podobně. Bohužel vyřešit tento problém je v rozměrných „prostorech“ velice složité, až téměř nemožné. Zároveň neexistuje algoritmus, který by měl podstatně lepší výsledky než Brute force. Z tohoto důvodu se hlavní pozornost začala věnovat aproximacím přesných výsledků. Tyto aproximace mají překvapivě lepší výsledky a zároveň pracují rychleji. Proto

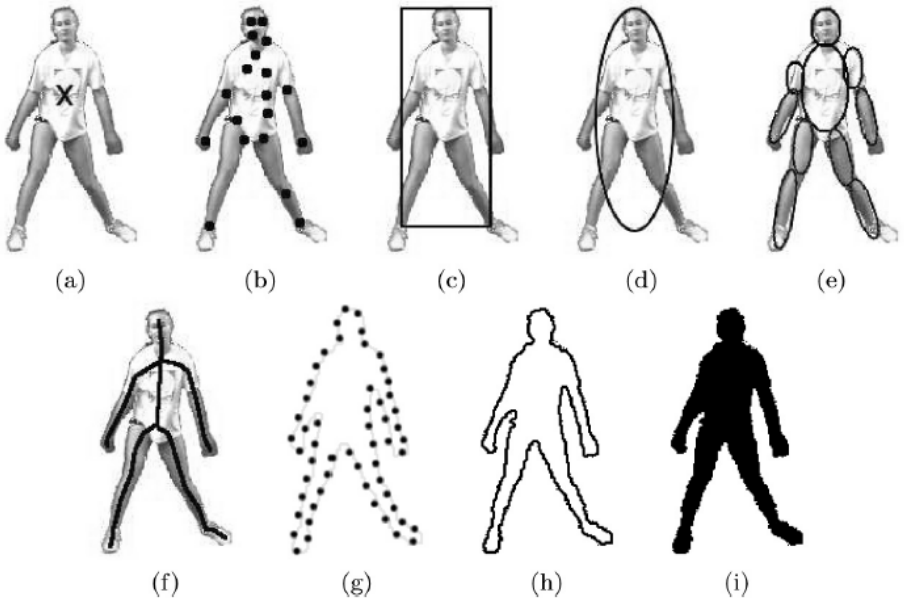
byla vytvořena knihovna, která některé z těchto algoritmů obsahuje a umožňuje s nimi pracovat. Tato knihovna se jmenuje FLANN (Fast Library for Approximate nearest neighbors), neboli knihovna pro rychlou aproximaci nejbližších sousedů a je nejpoužívanější knihovnou v rozpoznávací tématice. Zároveň nám umožňuje jednoduché použití za pomocí jazyků C, Matlab a Python. Jak již bylo řečeno, díky knihovně FLANN je možné rychle a efektivně dojít ke správným výsledkům.

2.4 Reprezentace sledovaných objektů

Jako zájmový (sledovaný) objekt může být definováno téměř cokoliv. Například člověk kráčející v davu, automobil projíždící křižovatkou, nebo letadlo letící po obloze. Je třeba brát zřetel na to, že tento objekt může nabývat různých tvarů, velikostí a barev. Z tohoto důvodu je důležité si před samotným trackováním zvolit vhodnou reprezentaci zájmového objektu. V této části se zaměříme na různé možnosti reprezentace objektů, zobrazené na Obr. 2.2

- **Bodová reprezentace (a,b)** Objekt může být reprezentován jedním bodem, který se většinou vyskytuje ve středu tohoto objektu a nazývá se centroid, nebo pomocí množiny bodů, jež se rozprostírá po sledovaném objektu. Tato reprezentace se nejčastěji používá k trackování objektů, kde tyto objekty zabírají pouze malou část snímku.

- **Reprezentace základních geometrických útvarů (c,d)** Tvar objektu je reprezentován například pomocí obdélníku a elipsy. Nejčastěji se tato reprezentace používá u pevných předmětů, ale není výjimkou ji použít i u pohybujících se objektů, například lidí.



Obrázek 2.2: Různé možnosti reprezentace objektů[13].

- **Kloubový model (e)** Sledovaný objekt je rozdělen na části, které jsou spolu spojené klouby. Jako nejlepší příklad si můžeme představit lidské tělo. Přičemž je mezi jednotlivými částmi přesně nadefinovaný vztah. Tato reprezentace je nejvhodnější k detekci lidského těla.

- **Skeletový model (f)** Skeletový model nám představuje kostru sledovaného objektu. Tato reprezentace je vhodná ke sledování pevných i kloubových modelů.

- **Reprezentace siluetou/konturou (g,h,i)** Kontura představuje hranu sledovaného objektu a oblast uvnitř kontury nazýváme siluetou. Konturu může představovat jak spojitá funkce, tak i množina bodů. Při detekcích objektů s výrazně měnícím se tvarem jsou doporučovány právě tyto metody.

2.5 Chyby při snímání obrazu

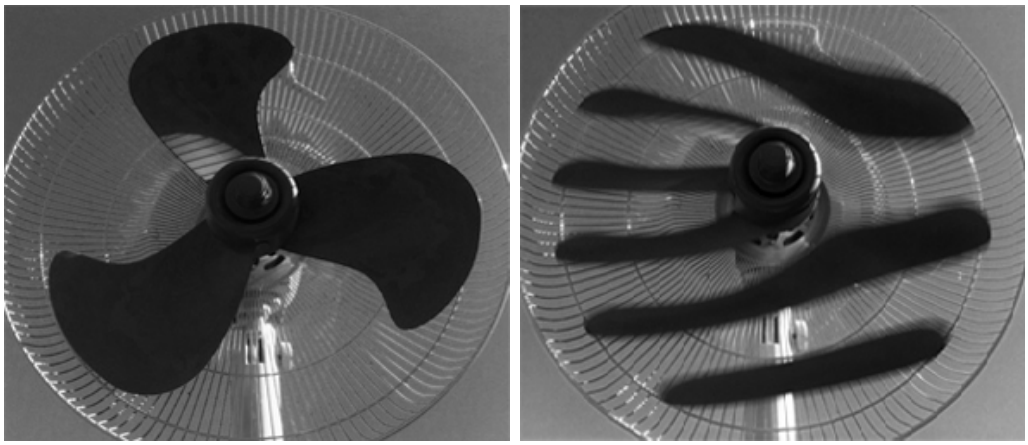
Největším nedostatkem v oblasti počítačového vidění je značná ztráta a možnost deformace informací, při snímání obrazů za pomocí videokamery, nebo fotoaparátu. Mezi hlavní příčiny těchto ztrát a deformací patří:

- perspektivní zobrazení 3D 2D
- velké množství dat
- šum
- rolling shutter
- sférické zkreslení
- úroveň hodnoty jasu

Některé z nich budou probrány v následující části.

2.5.1 Rolling shutter

Ve většině případů je digitální obraz snímán po jednotlivých řádcích nikoli jako celek a z tohoto důvodu dochází za určitých podmínek k poměrně značné deformaci obrazu nazývanou rolling shutter efekt a zobrazenou na Obr.2.3. Česky kácení svislic při švenkování. K tomuto efektu dochází nejčastěji při prudkém trhání se záznamovým zařízením a při snímání rychle rotujících, nebo pohybujících se předmětů, jako jsou vrtule letadel, ventilátory a podobně. Prudkými změnami v obraze dojde k deformaci řádků a tím i ke změnám na zaznamenaném obraze.

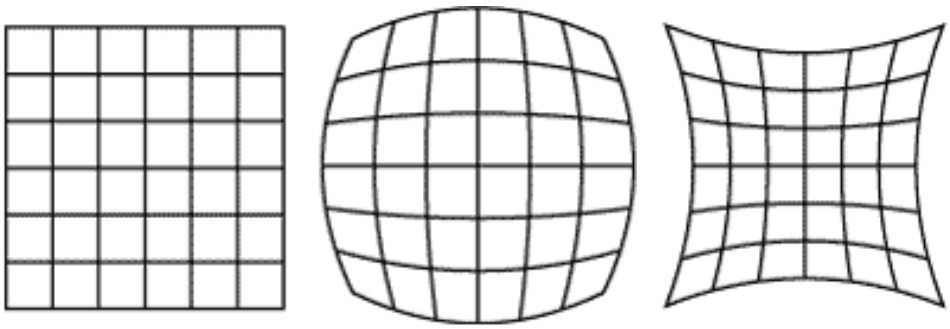


(a) (b)

Obrázek 2.3: Rolling shutter efekt

2.5.2 Sférické zkreslení

Ke geometrickému sférickému zkreslení na Obr.2.4 dochází především při pořizování snímků za pomocí širokoúhlých snímačů, kdy úhel optické osy snímače a snímané plochy je jiný než pravý. Tato chyba je nejpatrnější, pokud jsou v krajních oblastech snímku rovné útvary. Výhodou je, že pomocí geometrické transformace jsme schopni tuto chybu snadno potlačit.



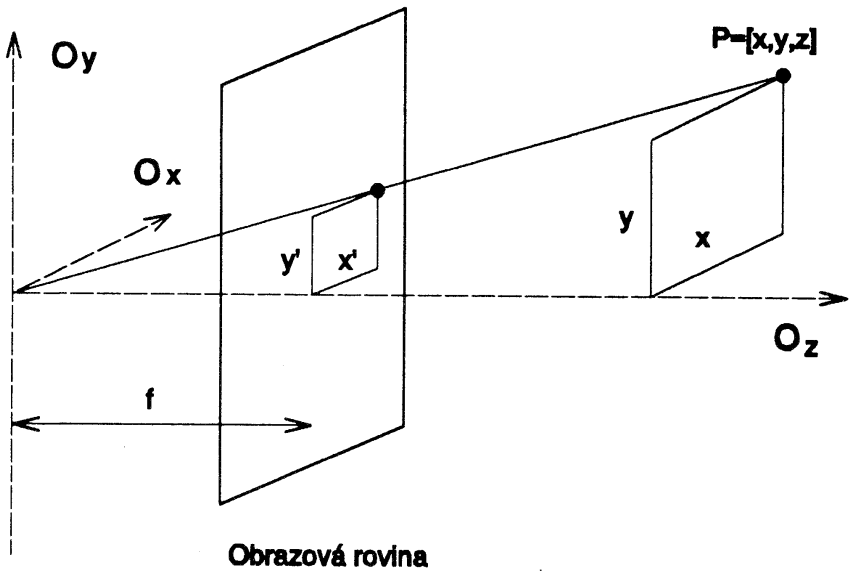
(a) (b) (c)

Obrázek 2.4: Možnosti sférického zkreslení, kde (a) je skutečný obraz bez zkreslení a následnují dva zkreslené snímky, kde u (b) došlo k soudkovému a u (c) poduškovitému zkreslením. [14]

2.5.3 Perspektivní zobrazení

Prostředí, v kterém se běžně pohybujeme, má trojrozměrnou povahu a při jeho snímání je získaná dvourozměrná obrazová funkce dle [12] výsledkem perspektivního zobrazení části 3D prostoru. Tento model je velice realistický a odpovídá získání obrazu v dírkové komoře. Geometrie perspektivního zobrazení části 3D prostoru můžeme vidět na Obr. 2.5., kde (x,y,z) představuje souřadnice v prostoru a f je ohnisková vzdálenost. Potom k získání souřadnic, pro jednotlivé body v perspektivním

zobrazení, využijeme následujících vztahů.



Obrázek 2.5: Geometrie perspektivního zobrazení [12]

$$x' = \frac{xf}{z}; y' = \frac{yf}{z} \tag{2.1}$$

Při pořizování snímku v reálném prostředí dochází k transformaci 3D prostoru do 2D zobrazení a zároveň s tím ke zkreslení velkého množství dat. Pokud nemáme k dispozici snímek stejného prostředí pořízený z jiné perspektivy, nejsme schopni na základě pouze jednoho 2D zobrazení rekonstruovat reálné prostředí.

Kapitola 3

Metody rozpoznávání obrazu pomocí významných bodů

Bodové detektory jsou využívány k nacházení významných bodů, jež se už poměrně dlouhou dobu využívají v pohybové a sledovací problematice. Tyto body se zpravidla vyznačují

- výraznou texturou ve svém okolí
- jasně daným umístěním
- stabilitou, která naznačuje znovulokalizovatelnost při jeho globálních a lokálních změnách

3.1 Historie

Vývoj metod používaných k detekci významných bodů a následnému rozpoznávání obrazu se neustále vyvíjí a jeho původ sahá až do sedmdesátých let minulého století, kdy byl pravděpodobně poprvé použit H.P. Moravcem pro řízení vozíku pomocí obrázků snímaných dvěma kamerami a následně v roce 1980 popsán [15].

3.1.1 Moravcův operátor

Úvod

Moravcův operátor, neboli také Moravcův detektor využívá pro vyhledávání významných bodů co největších odlišností malých oblastí od zbytku jeho okolí. Bohužel nejlepší výsledky má zejména v rozích a na hranách a z tohoto důvodu se hodí spíše pro detekci hran. Jeho jedinou výhodou je jeho jednoduchost spojená s výpočetní nenáročností.

Princip

Princip Moravcova operátoru vychází z rovnice 3.1, kde funkce g(i,j) představuje vstupní šedotónový snímek a f(i,j) výsledný obraz, který reprezentuje průměrný rozdíl jasu bodu v porovnání se všemi jeho osmi sousedy.

$$f(i,j) = \frac{1}{8} \sum_{k=i-1}^{k=i+1} \sum_{e=j-1}^{e=j+1} |g(k,e) - g(i,j))| \tag{3.1}$$

Tento výpočet se provede pro všechny body obrazu a tím vytvoří takzvanou mapu rohovitosti. Při výpočtech je třeba brát zřetel na možnost přístupu k neexistujícím sousedům bodů, nacházejících se na krajích snímku a tyto body ošetřit výjimkou.

3.2 SIFT

3.2.1 Úvod

V roce 1999 byla Davidem Lowem v [16] popsána nová metoda pro extrakci významných bodů z obrazového snímku. Metoda nese jméno SIFT (Scale-Invariant Featue Transform) vycházející především z jejího principu. Ten využívá transformování obrazových dat do měřítkově nezávislých souřadnic. Zmíněná metoda byla dále upravena i k následnému vyhledávání společných rysů ve dvou podobných snímcích a v roce 2004 spolu s publikováním [7] patentována. Tato práce vychází z mnoha dalších výzkůmu prováděných v oblasti počítačového vidění, ale nachází nový systém deskriptorů, pomocí kterých je možné porovnávat stejné lokace na podobných snímcích.

Image transformation	Match %	Ori %
A. Increase contrast by 1.2	89.0	86.6
B. Decrease intensity by 0.2	88.5	85.9
C. Rotate by 20 degrees	85.4	81.0
D. Scale by 0.7	85.1	80.3
E. Stretch by 1.2	83.5	76.1
F. Stretch by 1.5	77.7	65.0
G. Add 10% pixel noise	90.3	88.4
H. All of A,B,C,D,E,G.	78.6	71.8

Obrázek 3.1: Procentuální počet stejných významných bodů (Match %) a jejich orientace (Ori %) pro různé variace transformací použitých na 20 různých obrázků[16]

Bohužel při jejím vytváření byly kladeny požadavky především na spolehlivost, kdy je metoda nezávislá na měřítku, rotaci, šumu a kontrastu viz. Obr. 3.1, je metoda z hlediska rychlosti téměř nepoužitelná pro real-time aplikace.

3.2.2 Princip funkce

Funkci metody SIFT můžeme rozdělit do 4 základních fází, které si nejdříve stručně popíšeme a následně podrobně rozebereme.

1. **Scale-space a vyhledávání lokálních extrémů:** První část výpočtů se zabývá konstrukcí měřítkově nezávislé reprezentace obrázku, neboli Scale-space, pomocí které se dále vyhledávají lokální extrémy. Z důvodů zvýšení výkonu je používána aproximace pomocí rozdílu Gaussových funkcí.

2. **Lokalizace významných bodů:** Dle stability získaných bodů vytřídíme body nevyhovující a pro zbylé zpřesníme jejich polohu pomocí interpolace.

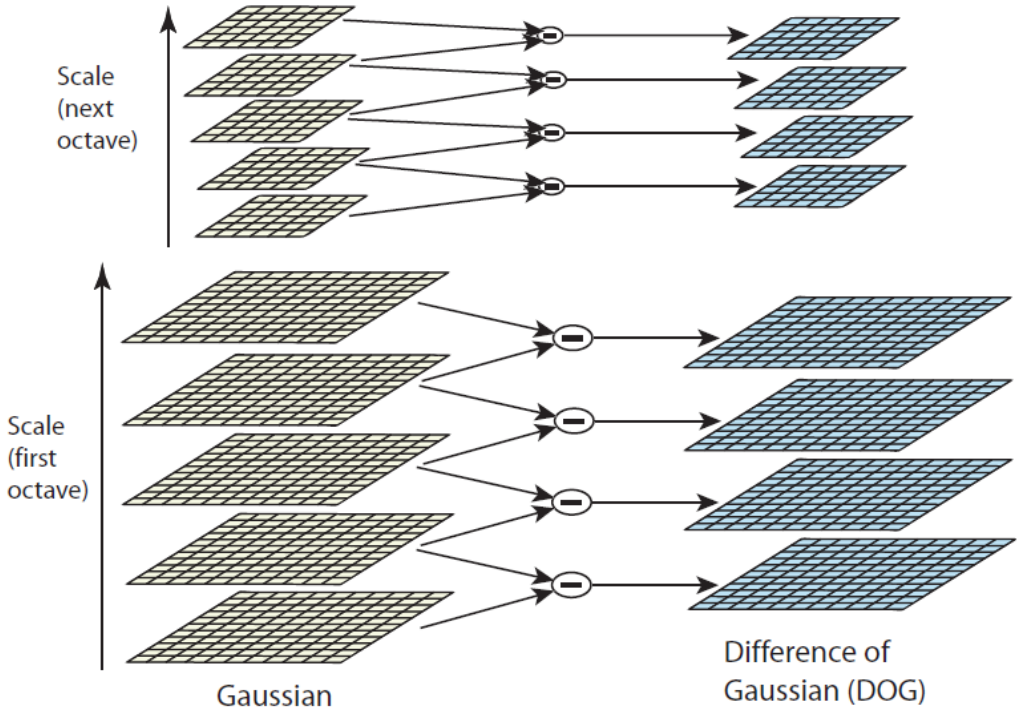
3. **Přiřazení orientací:** Každému významnému bodu je přiděleno po jedné, nebo více orientacích, které jsou založeny na lokálních směrech gradientů. Tímto zamezíme závisloti na rotaci.

4. **Vytvoření deskriptorů:** Na základě získaných orientací nastavujeme pro každý významný bod deskriptor. Pomocí deskriptorů jsme poté schopni tyto body porovnávat s jinými získanými z podobného snímku.

Scale-space a vyhledávání lokálních extrémů

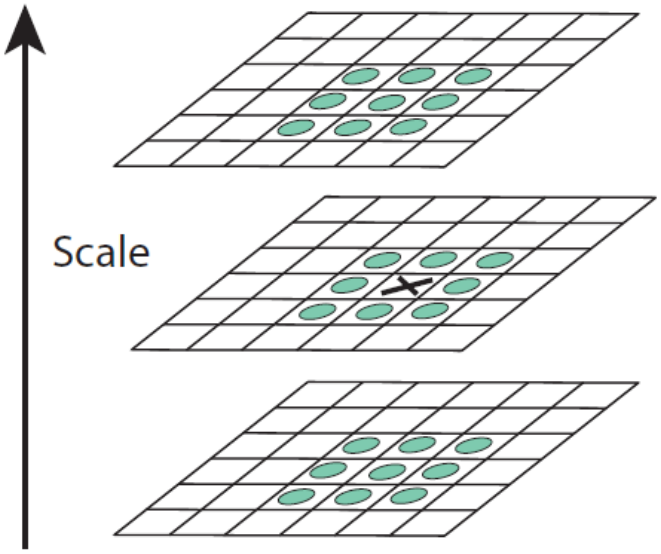
Jak již bylo zmíněno při popisu metody, v první části metody se snímek nejprve převede do měřítkově nezávislé reprezentace nazývané scale-space Obr. 3.2. V případě metody SIFT se k tomuto převodu, z důvodu zrychlení metody, využívá aproximace rozdílem Gaussových funkcí (DoG), místo generování pomocí Laplaciánu Gaussovy funkce (LoG). LoG má sice daleko lepší vlastnosti, ale jeho použití vylučuje vysoká výpočtová náročnost. Rozdílem Gaussových funkcí se rozumí odečtení dvou na sobě nezávislých Gaussových funkcí, které mají zároveň různá měřítka. Opakováním konvoluce Gaussianu, pro každou oktávu, postupně získáme vrstvy zobrazenené na Obr. 3.2 vlevo a jejich následném odečtením získáme DoG zobrazen na Obr. 3.2 vpravo.

Lokální extrémy, neboli maxima a minima DoG, jsou detekovány porovnáváním pixelu X se všemi jeho sousedy a to v jeho aktuálním, předchozím i následujícím prostoru o velikosti 3x3. Celkově je tedy porovnán s 26 sousedy, které můžeme vidět na Obr. 3.3. Tato porovnání se provádějí pro všechny vrstvy, kromě první a poslední, pro které neexistuje předchozí, respektivně následující vrstva.



Obrázek 3.2: Postup při generování Scale-Space za pomocí rozdílu gausových funkcí.[7]

Aby byl bod vybrán jako lokální extrém, musí splňovat jednu ze dvou podmínek a to být větší, nebo menší než všechny jeho výše popsaní sousedé. Pokud tuto skutečnost splňuje, state-space si jeho souřadnice zapamatuje pro další část metody, která se zabývá lokalizací významných bodů. I když se může zdát, že je tato metoda časově složitá, opak je pravdou. Většina vybraných bodů je eliminována při prvních kontrolách.



Obrázek 3.3: Detekce významných bodů pomocí porovnání pixelu X s jeho sousedy.[7]

Lokalizace významných bodů

V předcházejícím kroku jsme nalezli lokální extrémy a zapamatovali si jejich celočíselné souřadnice a vrstvy, ve kterých se tyto body nacházejí, ale z důvodu jejich nižší stability je výhodnější vyhnout se přesným polohám a aproximovat je pomocí funkcí popisujících okolí těchto bodů tak, jak to popsal M. Brown v [6]. Tyto funkce pro popis okolí bodů v souřadnicích a měřítku používají Taylorův rozvoj

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x, \tag{3.2}$$

kde D(x) je rozdíl Gaussových funkcí, který je aproximativním řešením LoG, $x = (x, y, \sigma)^T$ jsou souřadnice lokálního extrému ve State-space a kde hodnota D a její derivace jsou hodnoceny dle místa klíčového bodu. Položením rovnice rovno 0 a jejím zderivováním podle x vyjádříme sub-pixelovou a sub-měřítkovou pozici jako extrém $\hat{x}$.

$$\hat{x} = \frac{\partial^T D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \tag{3.3}$$

Pokud nám vyjde $\hat{x} < 0.5$, tak se tato hodnota přičte k souřadnicím počítaného klíčového bodu. V opačném případě, tedy při $\hat{x} > 0.5$, je nám naznačeno, že v blízkém okolí existuje jiný bod. Pokud k takovéto situaci dojde, je klíčový bod vyměněn právě za tento blízký bod.

Odstranění výnamných bodů z okolí hran

Velkou nevýhodou při použití DoG funkce je vysoká odezva v okolí hran. Z tohoto důvodu vznikne právě v těchto oblastech mnoho bodů, vyznačujících se velikou křivostí kolmou na hranu a malou podél hrany. Kvůli jejich špatné lokalizaci je třeba je odfiltrovat za pomoci Hessovi matice

$$\mathbf{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}, \tag{3.4}$$

kde hlavní křivost kolmou na hranu reprezentují její vlastní čísla. Z hlediska větší složitosti výpočtu těchto čísel se od konkrétních výpočtů upustilo, a dle následujících úvah se vychází pouze z poměru jejich velikostí. Nechť Hessova matice H má vlastní čísla α a β, potom

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$Det(\mathbf{H}) = D_{xx} - {D_{yy}}^2 = \alpha\beta.$$

Po vyjádření r vyjadřujícího poměr mezi větší a menší hodnotou vlastního čísla jako $\alpha = r\beta$ nám vychází

$$\frac{Tr(\mathbf{H})}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

Po této úpravě je zřejmé, že podíl $Tr(\mathbf{H})$ a $Det(\mathbf{H})$ je závislý pouze na poměru vlastních hodnot. Jestli je vhodné zkoumaný bod zachovat či ho vyřadit z množiny významných bodů ověříme pomocí zvolené mezní hodnoty:

$$\frac{Tr(\mathbf{H})}{Det(\mathbf{H})} = \frac{(r+1)^2}{r}.$$

Přiřazení orientací

Určování orientací ke každému významnému bodu je důležité pro získání nezávislosti na rotaci zkoumaného snímku. Jednotlivým bodům, již získaným pomocí předchozí části metody, je přidělena jedna nebo více orientací v závislosti na jejich okolí. Dle měřítka každého bodu se vybere nejbližší plocha zjištěná při generování Scale-space a označí se jako L. Následně se z rozdílů jednotlivých pixelů pro každý bod obrazu $L(x, y)$ vypočítá velikost gradientů $m(x, y)$ a jejich orientace $\Theta L(x, y)$.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\Theta(x,y) = tan^{-1}(L(x+1,y) - L(x-1,y)) + (L(x,y+1) - L(x,y-1))$$

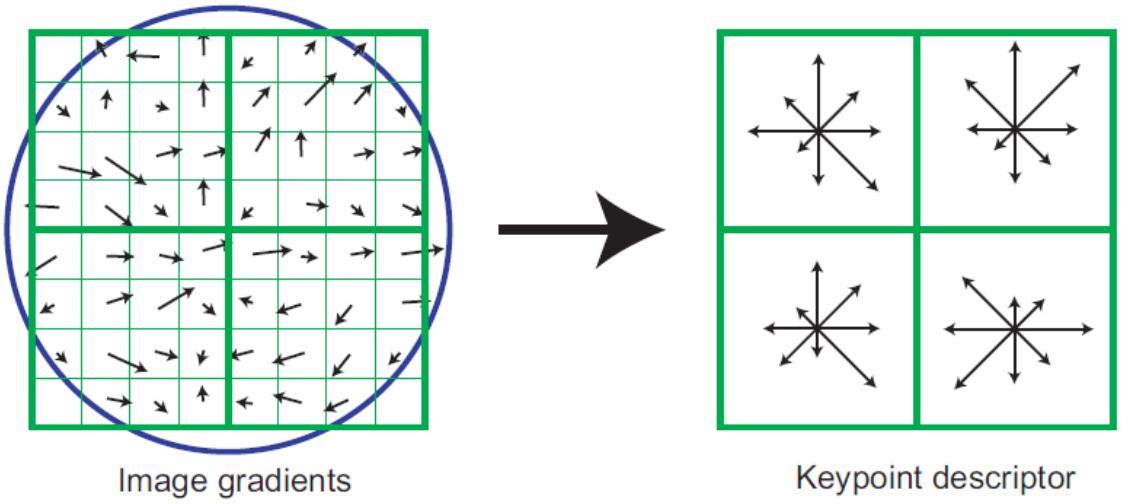
Na základě znalostí orientace gradientů v okolí zkoumaných bodů se sestaví histogram orientací, který pokrývá celých 360° pomocí 36 binů. Každému jednotlivému binu v histogramu je přičítána velikost gradientu vážená pomocí kruhového okna s Gaussovým rozložením a s ω rovnému 1,5 násobku klíčového bodu. Dominantní orientaci reprezentovanou nejdominatnější velikostí lokálního gradientu poté představuje orientaci významného bodu. Pokud ale existují další dominantní gradienty s alespoň 80 % maximální hodnoty, jsou pro ně vytvořeny nové významné body se stejným umístěním, ale jinou orientací. Těchto multi-orientačních bodů se v obrázku vyskytuje cca 15%, jejich existence výrazně zlepšuje stabilitu při hledání korespondujících bodů.

Vytvoření deskriptorů

V předešlých operacích se metoda zaměřovala především na určení množiny významných bodů s jednotlivými přesnými lokalitami a jejich orientací, přičemž byli všichni nestabilní kandidáti zavrženi. Zároveň každý významný bod v sobě tyto informace nese pro další využití, jako je vytvoření deskriptorů. Každý vytvořený deskriptor

nese informace o vzhledu okolí významného bodu a tím umožňuje porovnávávat korespondence s významnými body získanými na podobném, popřípadě stejném, ale upraveném obrázku. Metoda SIFT vychází ze získávání deskriptorů pomocí lokálních gradientů, které nepopisují okolí přesnými souřadnicemi a zároveň umožňují drobné posuvy v jejich umístění. Z toho důvodu nejsou závislé na umístění. Následkem použití gradientů je tato metoda odolná vůči menším geometrickým a světelným změnám ve snímku, což umožňuje lepší shody deskriptorů.

Pro jejich konstrukci je zapotřebí, stejně jako při přiřazování orientací, znát lokální gradienty v okolí jednotlivých významných bodů v obrázku. Jelikož už metoda má předpočítané hodnoty z předchozích fází je výhodnější je z důvodu ušetření výpočetního času použít a nepočítat je podruhé. Tímto využitím zároveň zanecháme nezávislost na měřítku a rotaci porovnávaného obrázku. Ke konkrétnímu výpočtu deskriptoru je potřeba si rozdělit okolí bodu na několik čtvercových oblastí, pro které se stejným způsobem jako ve 3. fázi vypočítají histogramy orientací. Data, podle kterých se dále deskriptory porovnávají, tvoří informace o gradientech reprezentujících tyto histogramy. Metoda SIFT běžně používá 128binové deskriptory, ve kterých je okolí významného bodu rozděleno na oblast 4x4 a histogramy orientace se rozdělují mezi 8binů. Největší výhodou takovýchto reprezentací je možnost drobné změny, způsobené například změnou perspektivy obrázku, která nemá vliv na konečný vzhled histogramu a tím i deskriptoru.



Obrázek 3.4: Vytvoření Deskriptoru významného bodu pomocí gradientů v jeho okolí.[7]

3.3 SURF

3.3.1 Úvod

Metoda SURF, neboli Speeded Up Robust Features, je jedna z novějších metod zabývajících se korespondencí podobných snímků. Byla prezentována v roce 2006 H. Bayem, T. Tuytelaarsem a L. Van Goolem formou vědeckého článku [17] a zaměřuje se především na rychlost a s tím spojeným použitím pro real-timeové aplikace. Jejich práce z velké části vychází z již běžně používaných metod, pro které se snažili najít možnosti urychlení. Například z již výše zmíněné metody SIFT. Tato metoda využívá deskriptor tvořen 128 biny, a proto při více členitých scénách dochází k vysoké výpočetní době při porovnávání. Právě zde autoři viděli možnost, jak proces urychlit. Například vytvořením deskriptoru, který bude složen z méně binů, ale bude i se zachováním stejné rozlišovací schopnosti daleko rychlejší.

3.3.2 Princip

Prezentovaný výzkum H. Baye, T. Tuytelaarsema a L. Van Goola probíhal spíše po cestě experimentální, ale zároveň se opíral o mnoho již dříve zjištěných a podložených informací, například [6][7][8][9]. Jak již bylo zmíněno v úvodu, metoda SURF víceméně vychází ze starší metody SIFT, kterou se snaží urychlovat. Z tohoto důvodu se její postup dá také rozdělit na 4 části:

- vytvoření Scale-space
- detekce významných bodů
- přiřazení orientací
- vytvoření deskriptorů

Přičemž se liší pouze jejich jednotlivé řešení snažící se urychlit celkový proces. U uvedených částí si podrobně probereme pouze jejich urychlující úpravy.

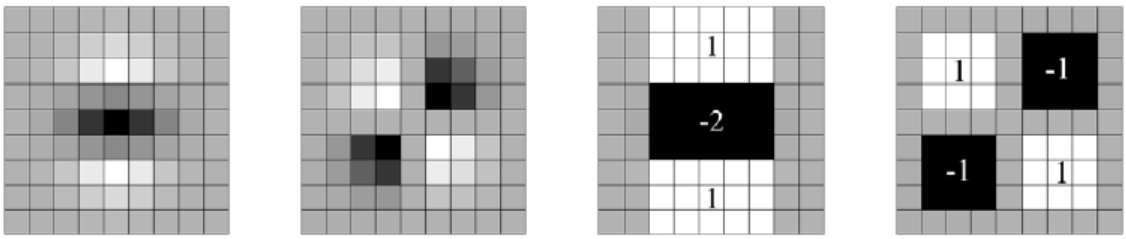
Urychlení v oblasti Scale-space pomocí determinantu Hessianu

Jak již bylo řečeno dříve, největší důraz byl kladen na zjednodušení a tím pádem i větší rychlosti výpočtů. V metodě SIFT byla pro generování scale-space používána aproximace Laplacianu Gaussovy funkce pomocí Gaussova diferenciálu. To sice metodu oproti použití pouze LoG urychlovalo, ale tato aproximace měla velké odezvy v okolí hran a z tohoto důvodu bylo třeba tyto body dále vyfiltrovávat pomocí determinantu Hessovy matice a tím docházelo k dalším časovým ztrátám. Z tohoto důvodu se autoři metody snažili přijít s přístupem jiným. Při bádání narazili na metodu používanou Mikolayczykem a Schmidem v [13], kteří využívali

opačný přístup. Nejprve pomocí Hessovy matice lokalizovali významné body a poté jim pomocí Laplacianu zkoušeli přiřazovat měřítka a tím se zbavili problému s vyfiltrováváním. Metoda SURF z tohoto důvodu používá jakýsi hybrid těchto dvou metod, kdy scale-space je generováno přímo pomocí determinantu Hessovy matice dle vztahu 3.5. Vezmeme-li ze snímku I bod $x = (x, y)$, potom je Hessova matice $H(x, \sigma)$ v bodě x a měřítku σ definována:

$$H(x, \sigma) = \begin{pmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{pmatrix}, \tag{3.5}$$

kde $L_{yy}(x, \sigma)$ je konvoluce snímku I v bodě (x,y) a druhé derivace Gaussovy funkce $\frac{\partial^2}{\partial x^2} G(\sigma)$ a podobně pro $L_{xx}(x, \sigma)$ i $L_{xy}(x, \sigma)$. Tato konvoluce tvoří podstatnou část výpočtu a z tohoto důvodu byla snaha urychlit právě tuto část úlohy. V metodě SIFT byla použita aproximace LoG pomocí DoG, která dle [7] i při použití velké aproximační chyby dojde k obstojným výsledkům. Z tohoto důvodu se autoři metody SURF rozhodli použít daleko radikálnější aproximaci a to pomocí obdélníkových funkcí, kde se spojité hladké funkce mění na nespojité skokové a tím výrazně upravují celkový vzhled zkoumaného snímku. I přes tuto výraznou změnu dochází k daleko lepší stabilitě, než při použití co nejpřesnější aproximace LoG. Zároveň je díky obdélníkové aproximaci možné použít takzvaný Integrální obraz a v nezávislosti na velikosti jednotlivých vniklých ploch je relativně rychle ohodnotit.



Obrázek 3.5: Použití obdélníkové aproximace. Zleva: druhé derivace Gaussovy funkce podle Y a XY. Následují dva obrázky jejich obdélníkových aproximací, kde šedivé lokace značí 0. [17]

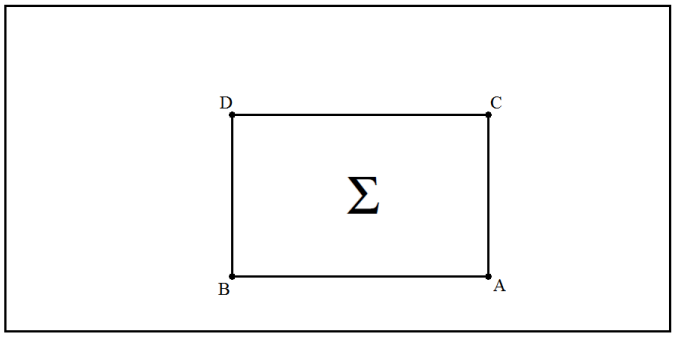
Integrální obraz

Integrální obraz se používá k rychlému sečtení všech hodnot v určité pravoúhelníkové oblasti obrázku. Při výpočtu se nad touto oblastí ve snímku vytvoří nová oblast se stejnou velikostí a pro jednotlivé pixely se vypočítá jejich hodnota, která odpovídá součtu všech pixelů od počátku obrazu k danému pixelu.

Pro Obraz I vypočítáme hodnotu pro zvolenou oblast jako:

$$I_{\sum}(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{z} I(i,j) \tag{3.6}$$

Jestliže máme nagenerovány hodnoty pro všechny pixely vybrané oblasti, jsme schopni zjistit celkovou hodnotu pouze jako součet čtyř krajních bodů, jak můžeme vidět na Obr. 3.6,



Obrázek 3.6: Ilustrace funkce integrálního obrazu

kde je tento součet je dám vztahem

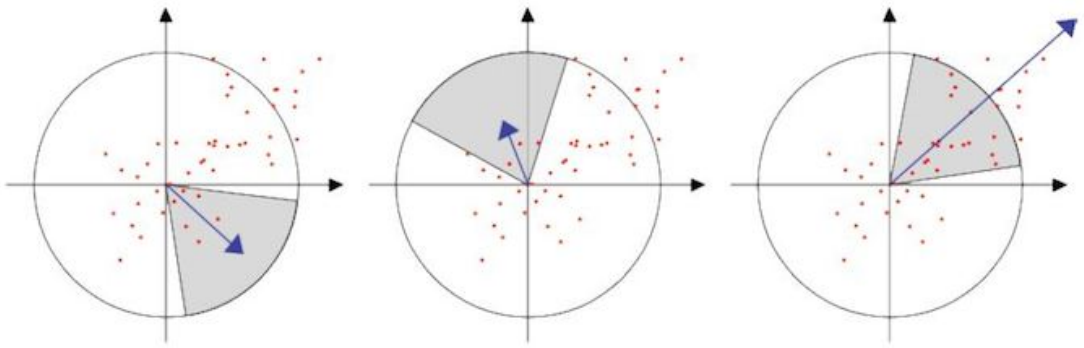
$$\sum = I_{\sum}(A) + I_{\sum}(D) - I_{\sum}(C) - I_{\sum}(B).$$

Bohužel v případě použití integrálního obrazu pro snímky s vysokým rozlišením, vyvstává drobný problém spojený se stále zvětšující se hodnotou směrem k pravému dolnímu rohu.

Přiřazování orientací

Stejně jako v dříve popsané metodě SIFT je umožněno nastavit nezávislost obrázku na rotaci. Tato nezávislost závisí na přiřazených dominantních směrech významných bodů, které jsou podobně jako v SIFT zjišťovány z jejich okolí. Hlavní rozdíl spočívá v tom, že se pro každý bod vybírá pouze nejdominantnější směr a ostatní se zanedbávají.

Dominantní směr se vybírá z kruhového okolí významného bodu o poloměru 6s, kde s představuje měřítko, ve kterém byl tento bod nalezen. Následně se v této oblasti vyčíslí odezvy takzvaných "Haarových vlnkových filtrů"ve směru os X a Y. S krokem s a velikostí hrany 4s. Tímto získáme v každé kruhové oblasti dvě odezvy, které po použití váhování Gaussovou funkcí se směrodatnou odchylkou $\sigma = 2{,}5$ a následném sečtení můžeme prohlásit za vektor. Poté ve stanoveném kruhu vybereme kruhovou výseč o příslušném úhlu rovnému $\pi/3$ a v ní sečteme všechny odezvy.



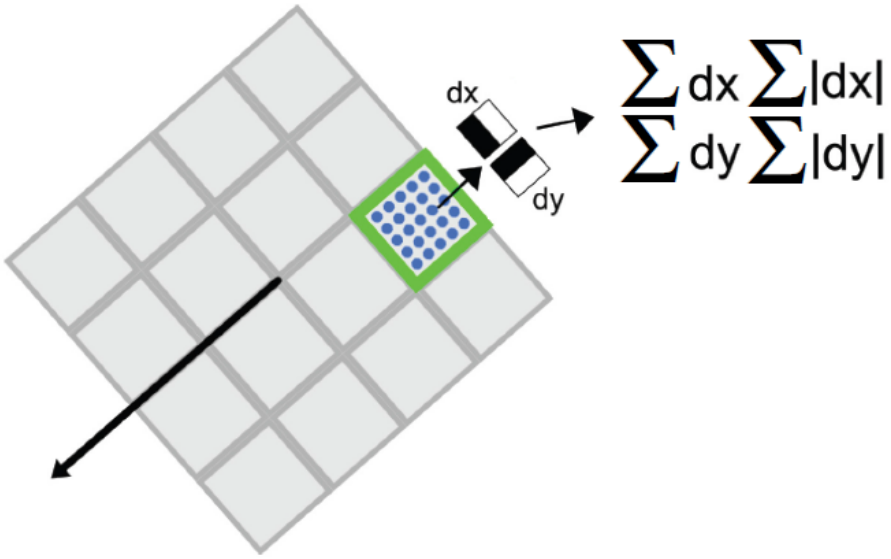
Obrázek 3.7: Grafické zobrazení výběru dominantních směrů

Výsledkem tohoto součtu je opět hodnota velikosti směrového vektoru. Tato část se opakuje dokud nedojde k úplnému pokrytí celých 360°. Následně ze všech iterací, sloužících k pokrytí celého kruhu, vybereme největší vektor, který je zvolen jenom dominantní pro právě počítaný významný bod. I když algoritmus umožňuje rotace v rozmezí ± 15°, je z důvodů zajištění stability, autory doporučována velikosti cca 10°.

Konstrukce deskriptorů

Jak již bylo řečeno dříve, autoři SURF se snažili ve všech oblastech metody SIFT přijít s nějakým vylepšením, které by snížilo výpočetní čas. V oblasti deskriptorů se nechali inspirovat popisem čtvercového okolí významného bodu pomocí gradientů. Zmíněné okolí popisuje čtvercový region o velikosti 20s, kde s značí měřítko počítaného bodu. Zároveň dochází k jeho otočení ve směru orientace z důvodu nezávisloti na rotaci obrázku. V dalším kroku, stejně jako v metodě SIFT, bude oblast popisující okolí významného bodu rozděleno na podoblasti, ale informace nacházející se v nich jsou jinak zpracovávány. Po již zmíněném vytvoření a natočení čtvercového okolí se toto okolí rozdělí na 4x4 částí o velikosti 5s s 5 rozmístěnými body. Pro ně následně spočteme odezvy Haarových vlnkových filtrů o hraně velikosti 2s ve směrech d_x a d_y. Vypočítané odezvy nepředstavují směry obrazových dat x a y, ale směry již pootočených os d_x a d_y, kde d_x reprezentuje odpovídající orientaci významného bodu a d_y směr na něj kolmý. Ke zvýšení robustnosti vůči různým deformačním chybám jsou získané hodnoty, stejně jako při zjišťování orientace, váhovány Gaussovou funkcí se směrodatnou odchylkou σ = 3,3s. Každé oblast z rozdělení 4x4 je poté popsána následujícím čtyřrozměrným vektorem:

$$v = [\sum d_x, \sum d_y, \sum d_{|}x|, \sum d_{|}y|].$$



Obrázek 3.8: Znázornění rozložení, otočení a postupu výpočtu SURF deskriptoru

Základ vytvořeného SURF deskriptoru nám tvoří 4x4x4 (64) vektorových hodnot, neboli binů. Právě počet binů, které deskriptor obsahuje má hlavní vliv na výpočetní dobu. Jelikož u SIFT detektoru byly vytvářeny deskriptory o velikosti 128 binů, výpočetní čas by měl být cca 2x rychlejší. Další významnou částí SURF deskriptorů je znaménko stopy Hessovy matice, které koresponduje s dvěma variantami:

- světlý významný bod na tmavém okolí(pozadí),
- tmavý významný bod na světlém okolí.

Protože k výpočtu Hessovy matice dochází v začátcích metody, je toto znaménko bez nároků na výpočetní čas používáno a tím je zlepšena celková robustnost porovnávání významných bodů.

Kapitola 4

Softwarové prostředky

Při vytváření zadané aplikace pro sledování dopravních značek jsem nebyl nikterak vázán a mohl si vybrat z velkého množství programovacích jazyků i různých vývojových prostředí. Jelikož jsem k programování v poslední době využíval především multiplatformní knihovnu Qt spolu s programovacími jazyky Python a C++, rozhodl jsem se pro jejich použití i v rámci této práce. Nicméně nesmím opomenout další významnou součást programování. Tou se stala knihovna OpenCV, která při tvorbě poskytuje základní funkce využívané v počítačovém vidění a tím mi programování znatelně ulehčila.

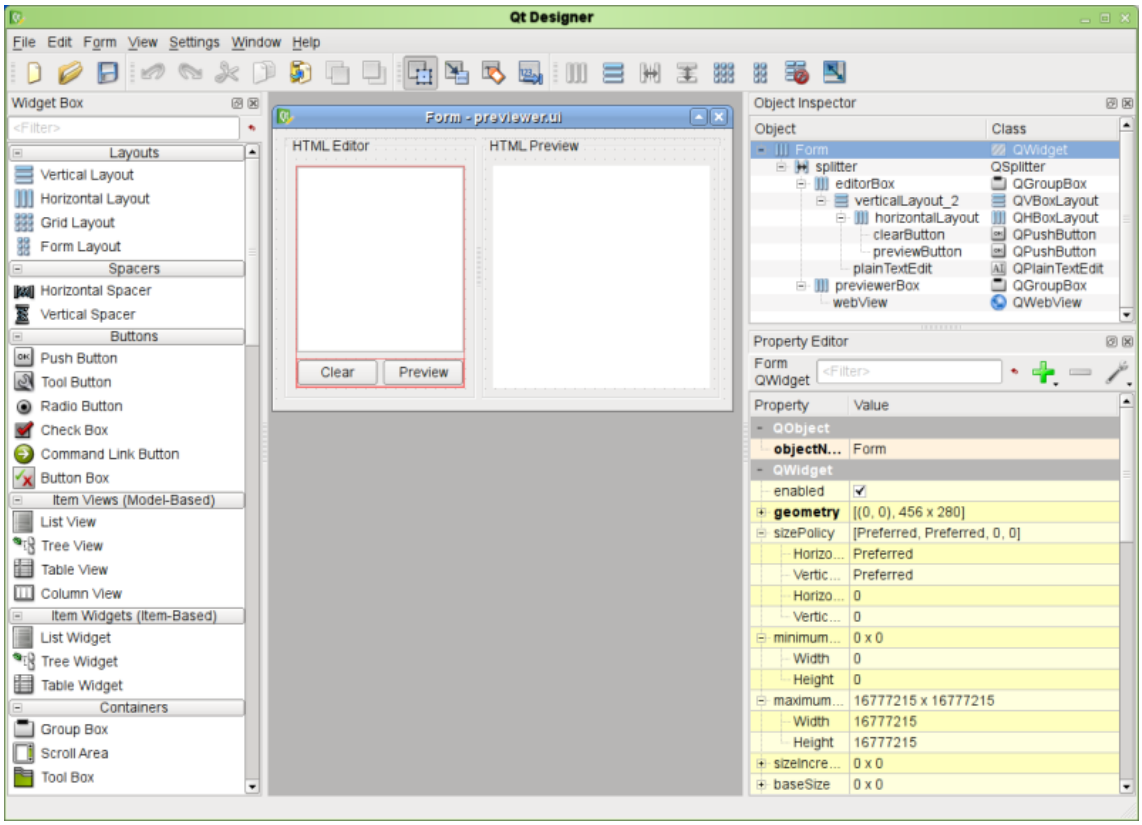
4.1 Qt

Qt je multiplatformní knihovna vytvořená v roce 1999 firmou Trolltech, která jej v roce 2008 prodala firmě NOKIA. Ta jí v dnešní době spravuje.

Hlavní části této knihovny, pomocí kterých je možné vytvářet aplikace s grafickým uživatelským rozhraním, ale i aplikace konzolové jsou:

- **QtCreator** , který je používán výhradně k psaní kódu v různých jazycích, jako jsou C, C++, Python a podobné.
- **QtDesigner** sloužící k návrhu grafické části aplikace, neboli uživatelskému rozhraní. Vizualizace prostředí QtDesigneru je zobrazeno na Obr. 4.1.

Mezi největší výhody této knihovny patří přehledná dokumentace a nativní vzhled (přizpůsobení dle operačního systému) vytvořených aplikací.



Obrázek 4.1: Defaultní zobrazení vývojové prostředí Qt Designer

4.2 Python

Python je dynamický, objektově-orientovaný programovací jazyk, který je využíván v mnoha oblastech vývoje softwaru. Je vyvíjen jako opensource a z tohoto důvodu je nabízen zdarma pro většinu běžných platforem.

Nabízí významnou podporu k integraci s ostatními jazyky a nástroji a přichází s mnoha standardními knihovnami. Při porovnání s ostatními programovacími jazyky nám Python umožňuje daleko efektivnější psaní kódu s čistelnějším kódem. Jeho největší výhoda spočívá v jednoduchosti z hlediska učení. Z tohoto důvodu se doporučuje jako nejvhodnější programovací jazyk pro začátečníky.

4.3 OpenCV

OpenCV je volně šířitelná multiplatformní knihovna určená pro práci v oblasti počítačového vidění a strojového učení. Byla vytvořena především k poskytování základních funkcí používaných v počítačovém vidění a zároveň k zrychlení a zjednodušení psaní konkrétního kódu. Knihovna obsahuje více jak 2500 optimalizovaných algoritmů, kreté mohou být použity k detekování a rozpoznávání tváří, identifikaci objektů, klasifikaci lidských činností, rozpoznávání scenérií, sledování pohybujících se předmětů apod. Zároveň podporuje tvorbu ve všech známých programovacích jazycích, jako jsou C++,C,Python, Java a MATLAB

```python
# import knihoven OpenCV
import cv2

# Načtení barevného obrázku
RGB_Image = cv2.imread('RGB_Image.png')

# Převedení barevného obrázku do šedého
Gray_Image = cv2.cvtColor(RGB_Image, cv2.COLOR_BGR2GRAY)

# Zobrazení šedého obrázku
cv2.imshow('Gray_Image',Gray_Image)

# Při stisku jakékoliv klávesy zavře okno s obrázkem
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Obrázek 4.2: Příklad kodu v jazyce Python za použití knihovny OpenCV provádějící načtení obrázku, převedení z formátu RGB do Grayscale a jeho následném vykreslení

4.4 C++

C++ je v dnešní době jedním z nejpoužívanějších programovacích jazyků, a to zejména díky tomu, že umožňuje využívat jak nízkoúrovňové, tak i vysokoúrovňové programovací vlastnosti. Jak je už z názvu jazyka zřejmé, C++ vychází z jazyka C, který rozšiřuje. Z tohoto důvodu byl do názvu přidán znak inkrementace ++, které značí rozšíření.

Kapitola 5

Implementace

5.1 Úvod

V předchozích částech jsem se věnoval především teoretickým informacím, ze kterých budu v této praktické části vycházet. Jak již bylo nastíněno v úvodu, hlavní náplní této bakalářské práce je vytvoření funkčního programu umožňujího rozpoznávání a trackování dopravních značek. Jako vstupní hodnoty jsou brány snímky zhotovené z videa reprezentující jedoucí auto a .JSON soubor obsahující informace o jednotlivých snímcích a značkách v nich nalezených. Mezi tyto informace patří především:

- číslo značky umožňující její identifikaci
- snímky na kterých se značka nachází
- lokace ve kterých se nachází

Hlavním úkolem bylo ověření platnosti těchto informací a v případě nesprávnosti umožnit jejich opravu. Ale nejdříve bylo nutné zajistit správné rozpoznávání jednotlivých značek.

5.2 Reprezentace značek

Po zvážení možných řešení pro reprezentaci objektů, zmíněných ve 2.4, jsem se rozhodl pro reprezentaci značek za využití výzmaných bodů. Jelikož má každá značka jiné významné body, mělo by dojít k jejich snadnějšímu rozeznávání než například při použití siluety, popřípadě kontury.

5.3 Určení významných bodů

K určení významných bodů, jsem se rozhodl vyzkoušet z doposud známých metod především novější metody SURF a SIFT, které jsou považovány za technologicky nejlepší k použití v rozpoznávací tématice. Podrobný popis jejich funkce jsem zmínil v kapitolách 4 a 5. V této části se zaměřím na jejich hlavní výhody a nevýhody a po jejich porovnání si jednu z metod vyberu a použiji.

5.3.1 Porovnání známých metod

Rozhodl jsem se porovnávat dvě běžné metody pro získávání a tvorbu deskripturů významných bodů. V kapitole číslo 5 jsem zmínil snahu autorů metody SURF o urychlení metody SIFT. Z výsledků zveřejněných v [17] můžu konstatovat, že se jim tato práce zdařila. Zároveň zachovali většinu výhod a umožnili jejich modifikace.

Společné výhody

Mezi hlavní společné rysy obou metod patří zejména nezávislost na:

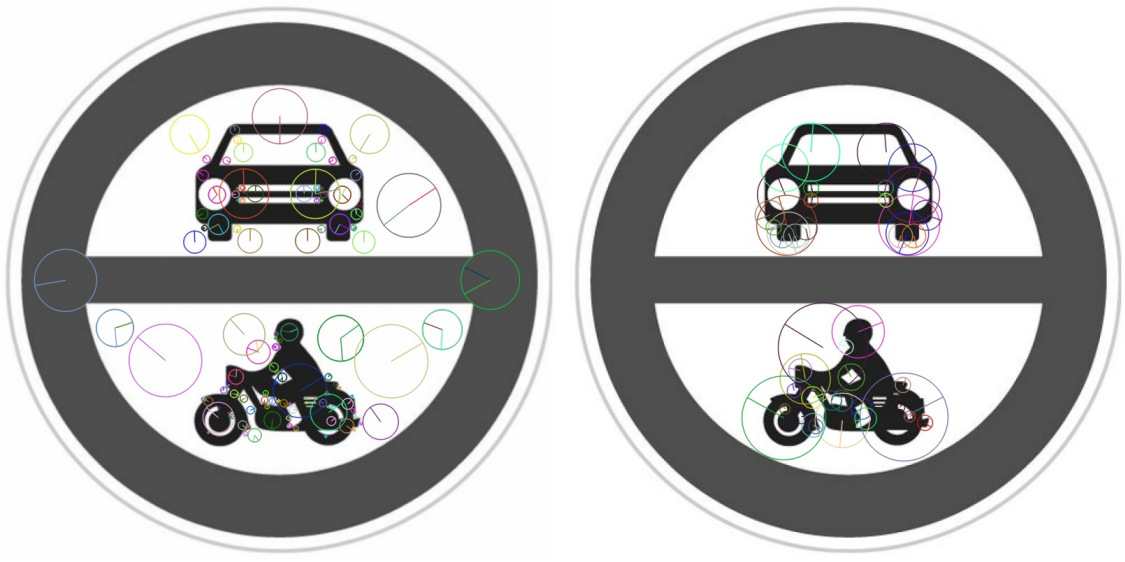
- rotaci
- měřítku
- posunu
- změně jasu/kontrastu
- různých geometrických zkreslením

Výhody metody SURF

Hlavní výhodou metody SURF je značné urychlení v oblastech vyhledávání významných bodů a následném zjišťování jejich deskriptorů. Autorům se i přes výrazné zjednodušení algoritmu podařilo dosáhnout téměř stejných výsledků jako v metodě SIFT, a tím zasadit důležitý pilíř při tvorbě trackovacích real-time aplikací. Dále je třeba zmínit, že při řešení některých úloh se výhody můžou snadno stát nevýhodami. V mém případě při trackování značek, které jsou vždy ve stejné výšce a nedochází u nich k téměř žádným rotací, u osově souměrných značek může v pozdější fázi algoritmu dojít ke spojení dvou bodů, které sice mají stejné okolí, ale vyskytují se na opačných stranách značky. Z tohoto důvodu není třeba aby významné body byly závislé na rotaci obrázku, a to právě metoda SURF umožňuje. Jako další výhodu vidím v možnosti zvětšení hodnoty deskriptorů z 64 binů na 128. Tato modifikace nám v určitých případech zlepší stabilitu významných bodů.

Vizuální porovnání

Na Obr. 5.1 můžeme vidět rozdíly mezi významnými body získanými oběma metodami. Metoda SIFT má tyto body víceméně rozložené po celém snímku s tím, že některé významné body se nacházejí v oblasti, ve které převládá pouze jedna barva, a to může zapříčinit fatální výsledky při matchingu. Nicméně při získávání významných bodů pomocí metody SURF jsou tyto body vyhledány zejména v potřebných lokalitách, které reprezentují jedinečný vzhled značky.



(a) (b)

Obrázek 5.1: Různé rozmístění a velikost deskriptorů popisujících okolí významných bodů, při použití metod (a) SIFT a (b) SURF

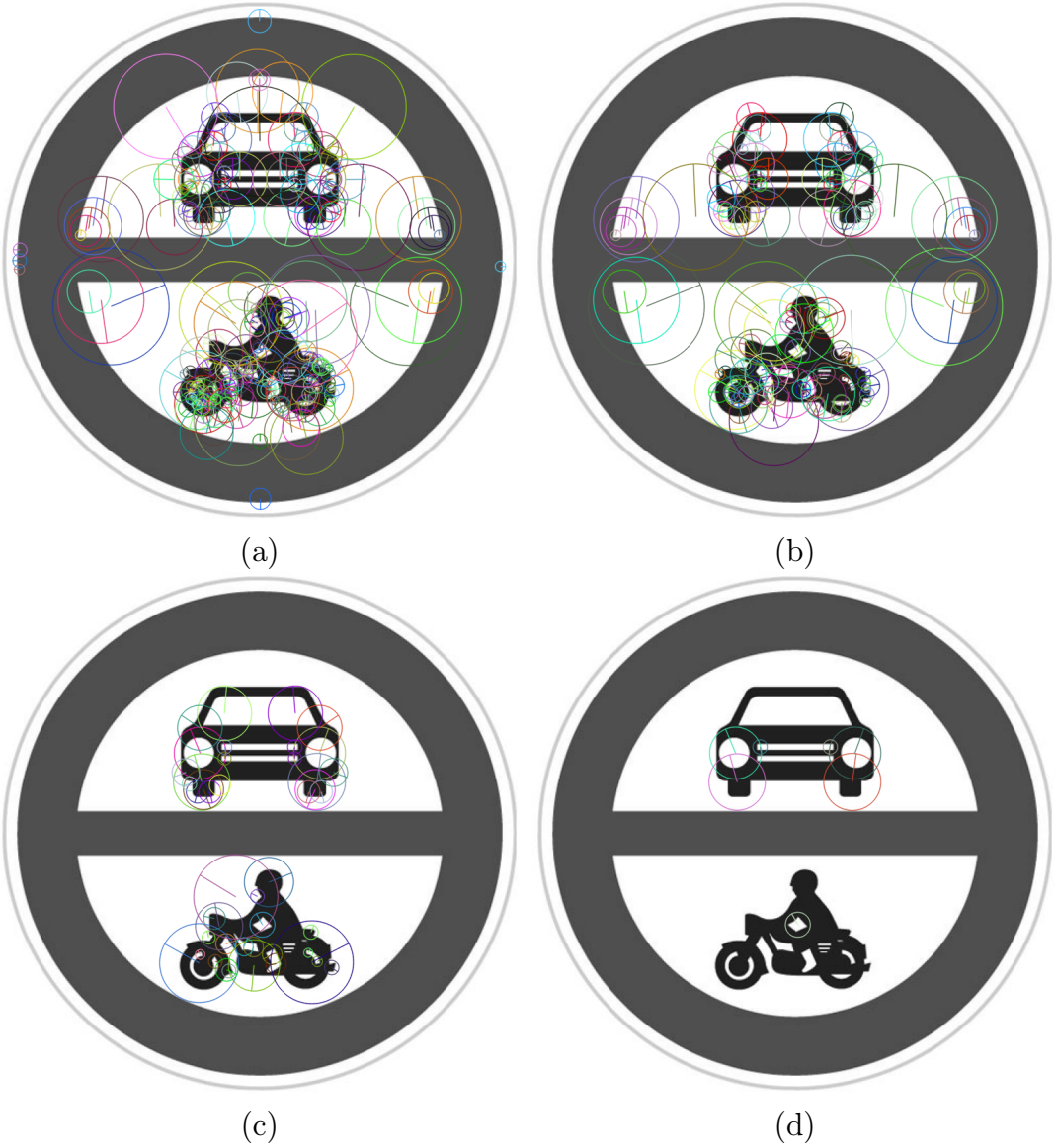
5.3.2 Implementace

Hessian Threshold

Po provedení porovnání a stanovaní si jasných priorit, jsem se rozhodl pro mojí práci využít metody SURF. Ačkoliv jsou získaná data podobná a se stejnými vlastnostmi jako při použití metody SIFT, je tento algoritmus podstatně rychlejší a navíc umožňuje zrušit nezávislost na rotaci.

Při vytváření programu jsem nejdříve zkoumal potřebnou hodnotu minHessianu redukující počet nalezených významných bodů. Tato hodnota je pro konečný program velice důležitá. Při použití malého čísla máme sice velké množství bodů, které můžeme porovnávat, ale tím také větší pravděpodobnost propojení nesprávných bodů. Naopak při zvolení velkého čísla tuto chybu sice minimalizujeme, ale v konečném důsledku můžeme mít malé množství správných propojení. V mém případě jsem zvolil dvě hodnoty, které jsou závislé na predikované velikosti hledaného předmětu.

Pokud je velikost značky větší než 70x70 pixelů, používám daleko větší hodnotu, než při opačné možnosti, a to hlavně díky potřebě minimalizovat počet významných bodů u větších značek a maximalizovat u značek menších.



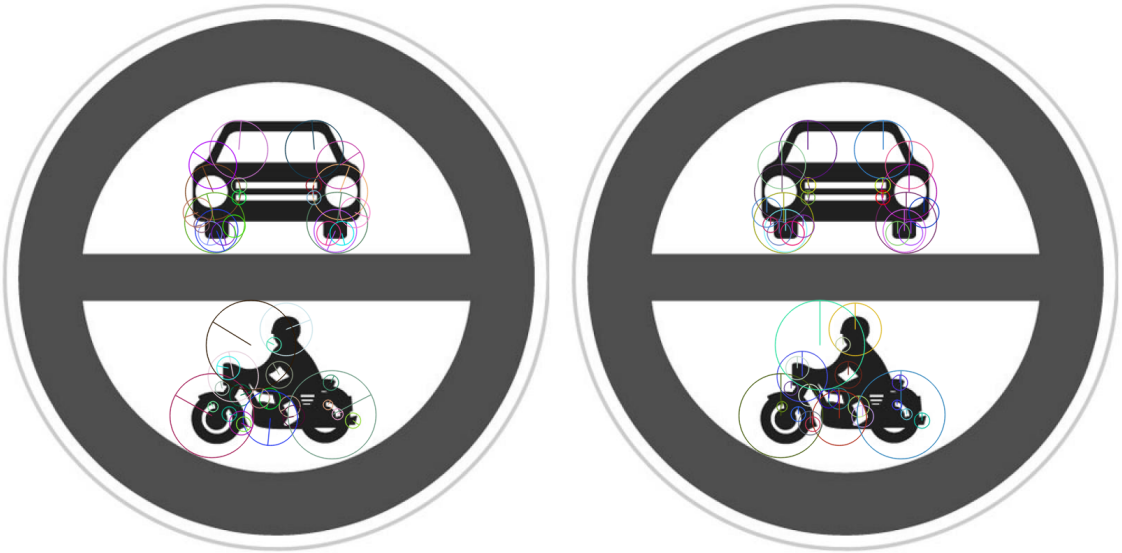
(a) (b)

(c) (d)

Obrázek 5.2: Různý počet významných bodů závislých na minHessianu. Obrázky představují hodnoty minHessianu a počet nalezených bodů. U obrázku (a)1000-231, (b)10000-135, (c)25000-45 a (d)50000-7.

Závislost na rotaci

Metoda SURF je primárně nastavena jako nezávislá na rotaci. Proto při rušení této nezávislosti je třeba nastavit určitou hodnotu v inicializaci funkce. Tato hodnota je typu boolean a tudíž nabývá pouze hodnot "TRUE"a "FALSE". Pro zrušení nezávislosti je třeba ji přepsat na hodnotu TRUE. Následně metoda přeskočí počítání směrů ke všem významným bodům a nastaví je defaultně na stejnou hodnotu. Z Obr. 5.3 je zřejmé, že zmíněné jednoduché řešení funguje.



(a) (b)

Obrázek 5.3: Natočení deskriptorů dle nastavení nezávislosti na rotaci. Obrázek (a) zobrazuje deskriptory nezávislé na rotaci a obrázek (b) závislé.

Velikost deskriptorů

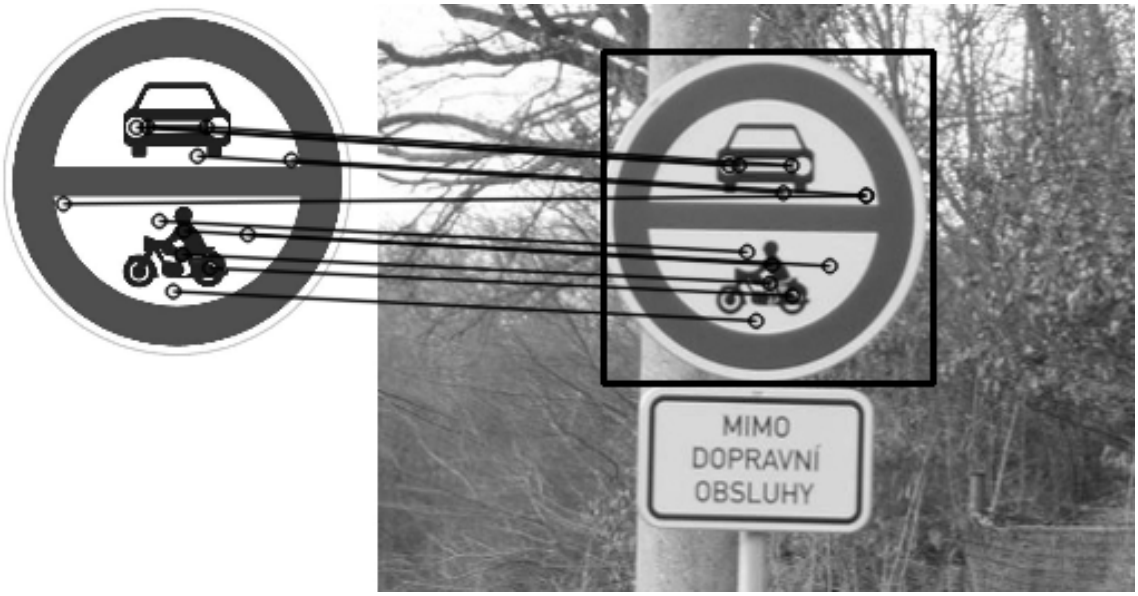
V základním nastavení metody SURF je deskriptor tvořen 64 biny. Pokud chce uživatel změnit velikost deskriptorů, musí stejně jako v předchozím bodě přenastavit booleanovou hodnotu, která nám tuto změnu umožňuje. K navyšení počtu binů, a tím zvětšení množství informací nesených jednotlivými významnými body, je třeba přenastavit booleovou hodnotu z "FALSE"na "TRUE". Toto navýšení by mělo mít vliv na zlepšení následujícího problému zabývajícím se matchingem, neboli hledáním stejných bodů ve dvou snímcích. Nicméně při použití v mé aplikaci byl viděn znatelný rozdíl pouze v době zpracování, která se zvětšila o přibližně 20%. Z tohoto důvodu jsem po různých experimentech použil základní nastavení s 64 biny.

5.4 Matching významných bodů

Další významnou částí programu je založena na procházení nalezených klíčových bodů ve dvou snímcích a hledání korespondujících dvojic. Pokud je takováto dvojice nalezena, je uložena a připravena na možné vykreslení. Dle množiny získaných dvojic jsme schopni dedukovat, zdali se ve zkoumaném snímku nachází hledaný objekt, či nikoliv.

5.4.1 Porovnání známých metod

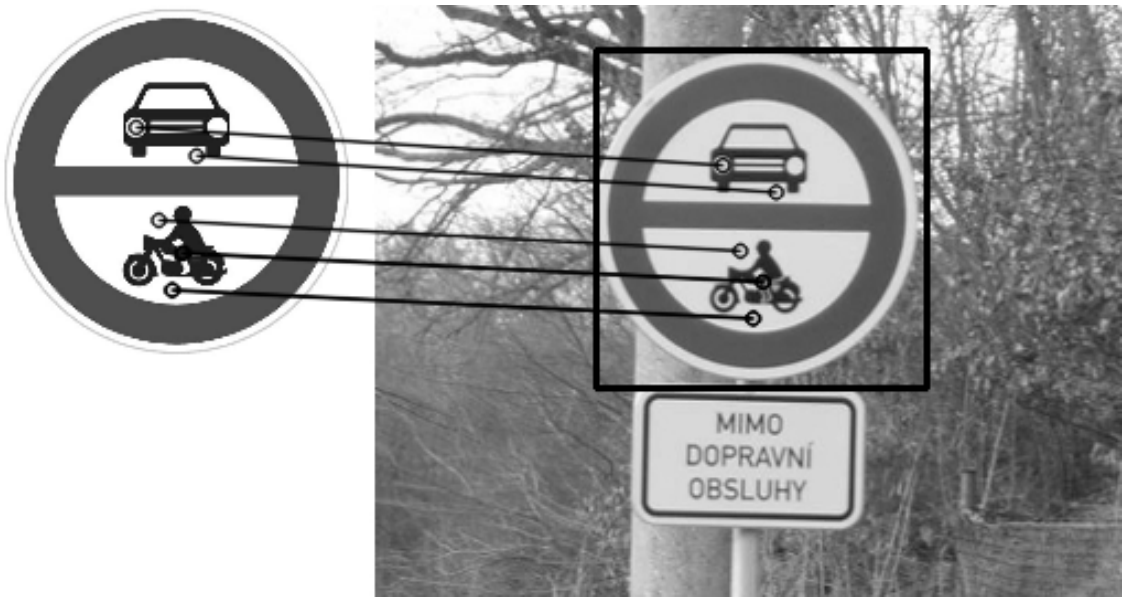
Mezi nejpoužívanější metody pro hledání korespondujících bodů patří především Brute force a FLANN matcher. Obě metody jsem blíže popsal v 2.3. Nyní si pouze okrajově zrekapitulujeme jejich vlastnosti a principy. Funkce brute force není nikterak složitá. Máme-li dvě množiny významných bodů, kde jedna množina představuje referenční obrázek a druhá obrázek zkoumaný, poté brute force macher vezme vždy jeden bod z množiny referenčního obrázku a porovná ho se všemy body z druhé množiny. Stejný postup provede i opačně, takže se vezme bod z druhé množiny a porovná se se všemi body z množiny první. Pro bližší představu je tento princip zobrazen na Obr. 5.4. Z důvodů porovnávání všech bodů jedné množiny se všemi body druhé množiny a naopak je tato metoda výpočetně složitější a může u ní docházet k chybným korespondencím častěji než při použití metody FLANN. Jedna z chybných korespondencí je zřetelná na obrázku Obr. 5.4.



Obrázek 5.4: Zobrazení korespondujících bodů a jejich propojení na referenčním a prohledávaném snímku

5.4.2 Implementace

Výhradně na základě možnosti chybných korespondencí u metody brute force, jsem se rozhodl pro použití metody FLANN, která neprochází veškeré možnosti, ale snaží se výsledek pouze aproximovat. Ačkoliv je princip založen pouze na aproximaci skutečných hodnot, její výsledky jsou ve většině případů přesnější a rychleji získané. Zmíněné korespondence a jejich přesnost jsou zřetelné při porovnání Obr. 5.5 s Obr. 5.4, kde u použití metody BF došlo k jedné chybné korespondenci, na rozdíl od metody FLANN, u které nedošlo k žádné chybě.



Obrázek 5.5: Zobrazení korespondujících bodů a jejich propojení na referenčním a prohledávaném snímku za použití knihovny FLANN

Před samotnou implementací metody FLANN je potřeba vytvořit strukturu <vector <vector DMatch >>, která nám umožňuje jednotlivé korespondence uložit. Následně tuto strukturu použijeme spolu s množinami deskriptorů, reprezentujících dva porovnávané snímky, při volání metody. Výstupem je nám tedy struktura podobná vícerozměrnému poli, která obsahuje všechny nalezené dvojice.

5.5 Výpočet přesné lokace značky

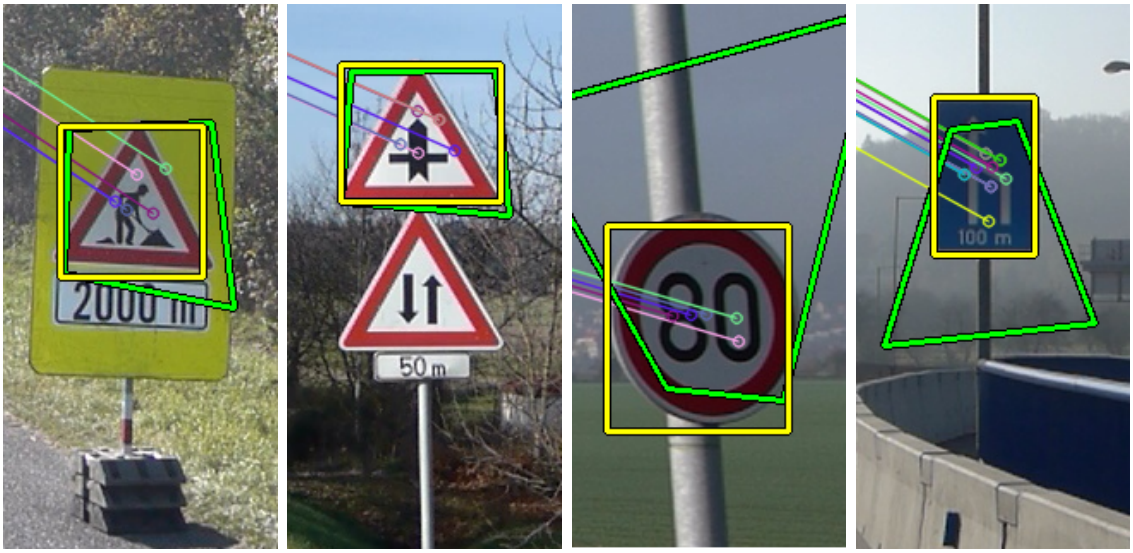
5.5.1 Úvod

Poslední částí při trackování objektů je stanovení oblasti, ve které se hledaný předmět nachází a tu následně zvýrazní respetive obkreslí. Běžně se v této problematice vychází ze znalostí souřadnic získaných korespondujících dvojic. Nejprve se pro ně vypočítá matice homografie a ta následně slouží k výpočtu perspektivní transformace. Tato transformace nám umožňuje zjistit a obkreslit přesnou lokaci, ve které se objekt nachází, i při různých geometrických změnách v obrazu.

5.5.2 Implementace

Pro výpočet a určení lokace hledaného objektu jsem nejprve používal běžně používané metody, využívající matice homografie a perspektivní transformace. Nicméně v mém případě docházelo k drobným odchylkám polohy jednotlivých významných bodů, které v některých případech měly až fatální důsledky při stanovení lokality. Z tohoto důvodu jsem byl nucen přijít s vlastním řešením. Princip mnou vyvinuté metody lze charakterizovat jako lineární transformaci souřadnic, na základě zna-

lostí poměrů hodnot v rámci osy X a Y, mezi dvojicí korespondujících bodů. Mnou vymyšlené řešení je i přes vynechání výpočtů matice homografie a perspektivní trasformace přesnější a zejména díky jeho jednoduchosti rychlejší. Navíc je pro zjištění polohy třeba pouze 2 korespondujících dvojic, narozdíl od nepoužité metody, pro kterou je nutné znát minimálně 4 dvojice. Pro lepší porovnání jsem v programu použil obě metody a výsledky si nechal zobrazit v obrazové formě. Rozdíly jednotlivých metod jsou patrné z Obr. 5.6.



Obrázek 5.6: Rozdíly mezi použitými metodami pro zjišťování souřadnic značek. Žlutá barva představuje mnou navržený a zelená konvenční algoritmus.

Kapitola 6

Výsledky

Funkčnost vytvořené aplikace jsem zjišťoval na snímcích vytvořených z videa, kdy toto video bylo pořizováno při jízdě osobním automobilem na území hlavního města Prahy a v jeho okolí. Vzdálenost mezi snímky odpovídá přibližně pěti metrům ve skutečnosti.

Nejprve jsem porovnával rozdíly při použití dvou metod zprostředkovávajících přesné souřadnice nalezené značky. Porovnání jsem provedl na 100 po sobě jdoucích snímcích a z Tabulky 6.1 je zřejmé, že při použití mnou vyvinuté metody aplikace nalezne o 30% více značek než konvenční algoritmus a zároveň nedochází k téměř žádným nepřesnostem.

Metoda	**Snímků**	**Správné** [%]	**Špatné** [%]	**Nepřesné** [%]
Konvenční	100	63	25	12
Vytvořená	100	93	6	1
Vytvořená	1500	94,5	5,125	0,0375

Tabulka 6.1: Tabulka popisující procentuální úspěšnost stanovení přesných souřadnic značky na procházeném snímku za použití dvou metod.

Následně jsem se zaměřil na zkoumání výsledků pouze mé metody. Provedl jsem vyhledávání na 1500 snímcích a při následném procházení a zjišťování správnosti, jsem došel k výsledkům potvrzující hodnoty již získané v předcházející kontrole.

Kapitola 7

Závěr

Úkolem této bakalářské práce bylo prozkoumat známé metody trackingu vhodné k použití v problematice rozpoznávání dopravních značek. Následně za pomoci získaných informací vytvořit aplikaci umožňující jejich sledování.

Nejprve jsem se věnoval především teoretické části a zkoumal dvě metody vhodné pro použití. Konkrétně SURF a SIFT. Na základě získaných informací jsem si zvolil metodu SURF. Oproti metodě SIFT je rychlejší a umožňuje některé modifikace parametrů, jako je zrušení nezávislosti na rotaci. Poté jsem se věnoval čistě vývoji aplikace. Při vývoji jsem narazil na řadu problémů, avšak právě díky nim jsem byl schopen dosáhnout finálních výsledků. Jedním z nich byl běžně používaný algoritmu pro zjišťování přesné lokace dopravní značky. Jeho úspěšnost byla v průměru pouze kolem 50%, z tohoto důvodu jsem se rozhodl vytvořit algoritmus vlastní. Ten i přes absenci perspektivní transformace umožňuje zejména díky jeho jednoduchosti přesné výsledky. Dle provedených testů, zahrnujících zkoušení na 1500 snímcích a více jak 200 druzích značek, jeho úspěšnost přesahovala 94%. Nicméně je třeba zmínit i nedostatky. Pokud dochází ke značným změnám ve velikosti značky mezi dvěma následujícími snímky, například vlivem perspektivy, z které se na značku díváme, může mnou vyvinutá metoda dojít k nepřesnému odvození souřadnic.

Další věcí znesnadňující sledování značek jsou zejména různé velikosti hledaných značek, a s tím spojené množství klíčových bodů, které jsme schopni zjistit. Jestliže je oblast reprezentující značku menší jak 90x90 pixelů, je jejich množství výrazně menší a bylo nutné pro tyto případy upravit algoritmus tak, aby množství nalezených bodů bylo větší. Mezi významné problémy bych také zařadil značky chudé na klíčové body (B1, B2, B26, P2, P4). Při zkoumání právě těchto značek je nalezeno pouze velmi malé množství významných bodů v oblasti značky a tím jsou další výpočty zkomplikovány. S některými z těchto značek se váže další problém. Ten vyvstává při symetričnosti procházených lokací. V takovémto případě dochází ke korespondenci mezi body nacházejícími se na protilehlých částích. V poslední řadě musím zmínit chyby způsobené fyzickými deformacemi značek a přiliš velkým jasem. Tyto podmínky zapřičinují chybovost i u běžně rozeznatelných značek.

Nicméně je třeba zmínit, že po provedených úpravách tvoří všechny výše uvedené chyby pouze malou část z velkého množství procházených značek nepřesahující 7%. Z tohoto důvodu můžu konstatovat, že aplikace až na drobné výjimky funguje bezproblémově.

Literatura

[1] SVOBODA, T., HLAVÁČ V.: Image processing, analysis, and machine vision: a MATLAB companion. Toronto: Thomson, 2008, xi, s.255.

[2] RUSS, J. C.: The image processing handbook. 5th ed. Boca Raton, 2007, s.817.

[3] MAGGIO, E., CAVALLARO A.: Video tracking: theory and practice. 5th ed. Chichester: Wiley, c2011, s.817.

[4] LAGANIÉRE, R., CAVALLARO, A.: OpenCV 2 computer vision application programming cookbook: over 50 recepies to master this library of programming functions for real-time computer vision. 1st ed. Brimmingham: Packt Publishing, c2011, III, s.287.

[5] Qt, Qt Project [online], 2014, Dostupné z: http://qt-project.org

[6] BROWN, M., LOWE, D.: Invariant features from interest point groups. In Proceedings of British Machine Vision Conference, pages 5–6, 2002.

[7] LOWE, D. G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):89–112, 2004.

[8] MIKOLAJCZYK, K., SCHMID, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision, 60(1):60–90, 2004.

[9] KADIR, T., BRADY, M.: Scale, saliency and image description. IJCV 45(2) (2001) 83 – 105

[10] MATTHIE, L., KANADE, T., SZELISKY, R.: Kalman filter based algorithms for estimating depth from image sequences, Int. J. Comput. Vis., vol. 3, 1989.

[11] TANIZAK, H., Non-gaussian state-space modeling of nonstationary time series. J. Amer. Statist. Assoc. 82, 1987.

[12] HLAVÁČ, V., ŠONKA, M. : Počítačové vidění. Praha: Grada, 1992.

[13] YILMAZ, A., JAVED, O., SHAH, M.: Object tracking: A survey ACM Computing Surveys, Vol. 38, No. 4, 2006

[14] SFÉRICKÁ VADA. [online], 2014, Dostupné z: http://www.fotoroman.cz

[15] MORAVEC, H. P. : Obstacle Avoidance and Navigation in the Real World bz a Seeing Robot Rover, Research at Stanford University, 1980

[16] LOWE, D. G.: Object Recognition from Local Scale-Invariant Features. International Journal of Computer Vision, 1999.

[17] BAY, H., TUYTELAARS, T., GOOL, L : SURF: Speeded up robust features. In Proceedings of European Conference on Computer Vision, pages I: 404–417, 2006.

Seznam obrázků

Příloha

Přílohou je vytvořená aplikace ve formátu .EXE, přiložená na kompaktním disku v kapse na zadní straně.