Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

Bakalářská práce

Analýza metod pro detekci příznaků v digitalizovaném obraze

Plzeň 2016 Petr Barborka

(originál zadání)

Prohlášení

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

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 9. května 2016

Petr Barborka

Poděkování

Chtěl bych poděkovat Ing. Petru Neduchalovi za vedení práce.

Abstract

This thesis contains a theoretical overview and a practical comparison of the main methods of detection and description of features in a digitalized image. It thoroughly describes principles of operation of point feature detection methods Moravec operator, Harris operator, Shi-Tomasi, SIFT, SURF, FAST, ORB a MSER and their corresponding descriptor algorithms along with BRIEF algorithm. Object detection methods Haar and Histogram of Oriented Gradients are also described. Descriptor comparison methods k-nearest neighbors and its approximation Best bin first are also noticed. Theoretical part concludes with description of the RANSAC method used here to approximate space transformation between two pictures from detected, described and matched points. The last part contains a comparison of detection and description methods and their combinations on the basis of distance between approximated homography method and its ground truth given as a part of the dataset used. The comparison is implemented in C++ using the openCV framework.

Keywords

Machine Vision, Point Features, SIFT, SURF, ORB, MSER

Abstrakt

Tato práce se zabývá popisem a srovnáním metod detekce a popisu příznaků v digitalizovaném obraze. Jsou v ní podrobně vysvětleny principy fungování metod detekce bodových příznaků Moravcův operátor, Harrisův operátor, Shi-Tomasi, SIFT, SURF, FAST, ORB a MSER a jejich příslušné deskriptorové algoritmy spolu s algoritmem BRIEF. Dále jsou popsány metody detekce objeků Haar a Histogram orientovaných gradientů. Zmíněny jsou i metody pro porovnávání deskriptorů pomocí algoritmu nejbližšího souseda a jeho aproximace Best bin first. Nakonec je uvedena metoda RANSAC sloužící zde k odhadu prostorové transformace mezi dvěma obrazy s nalezenými, popsanými a přiřazenými body. V poslední části jsou porovnány metody nalezení a popisu bodových obrazových příznaků na základě srovnání zadané matice homografie a její získané aproximace. Porovnání je implementováno v C++ s využitím frameworku openCV.

Klíčová slova

Strojové vidění, bodové příznaky, SIFT, SURF, ORB, MSER

Obsah

1 Úvod

Cílem této práce je poskytnout přehled možných přístupů k detekci a popisu příznaků v digitalizovaném obraze, ukázku jejich implementace a srovnání jejich výkonností. Práce sestává z teoretické části, kde jsou podrobně popsány použité algoritmy a části praktické, ve které je popsána implementace testovacího frameworku a prezentovány výsledky testů jednotlivých kombinací detektor - deskriptor na úloze nalezení homografie zobrazení při přechodu mezi párem testovacích obrazů.

V kapitole 2 je jsou vysvětleny klíčové pojmy detekce, desripce a porovnání příznakových bodů, možné scénáře aplikací těchto algoritmů a uveden kontext, ve kterém jsou jednotlivé metody uvažovány v této práci. V kapitole 3 jsou postupně představeny metody detekce a popisu příznaků. Nejprve jsou uvedeny metody použité k nalezení a popisu bodových příznaků (Moravcův operátor až MSER), poté jsou pro doplnění zmíněny metody indetifikace objektů Haar a Histogram orientovaných gradientů, které nejsou součástí porovnání, ale jsou zajímavou alternativou při řešení některých úloh zmíněných v kapitole 2. Taktéž jsou zmíněny možnosti využití detekce objektů a čar a jejich využitelnost pro systémy mapování. Následují dva příklady algoritmů použitelných ke spárování deskriptorů mezi dvěma obrazy: nejbližší soused a Best Bin First. Závěrem této kapitoly je popsán robustní regresní algoritmus RANSAC, který představuje jednu z možností odhadu prostorové transformace mezi dvěma obrazy na kterých byly nalezeny a přiřazeny bodové příznaky. Kapitola 4 se zabývá popisem impementace testovacího frameworku a popisuje výsledky porovnání výkonnosti párů detektor - deskriptor na použitém datsetu. Vyhodnocení výkonnosti je provedeno pomocí porovnání matice homografie zadané v datasetu a té aproximované prostřednictvím příznakových bodů nalezených porovnávanými metodami.

2 Příznaky v digitalizovaném obraze a jejich využití

V následujících kapitolách budou představeny především metody pro detekci a popis bodových příznaků. Tyto algoritmy se snaží v obraze nalézt body výrazné vzhledem k jejích okolí a popsat je tak, aby je při jejich opětovném nalezení v jiném obraze bylo možné identifikovat bez ohledu na transformaci mezi těmito obrazy (natočení, posunutí, roztažení, změna nasvětlení, ...).

Práce s bodovými příznaky zahrnuje:

- detekci příznaků - Nalezení nejvhodnějších příznakových bodů.
- deskripci příznaků - Popis příznaků tak, aby tyto byly správně identifikovány při opětovném nalezení za minimalizace vlivů osvětlení, prostorových transformací atd.
- způsob porovnání deskriptorů - Metodu, kterou budeme určovat, které dva deskriptory z různých obrazů popisují stejný bod nebo stejnou oblast.

Některé z prezentovaných metod práce s bodovými příznaky řeší všechny tři tyto problémy, jiné jenom některé z nich a pro jejich využití je tedy potřeba zbylé doplnit. Dále jsou uvedeny dvě metody pro detekci objektů, tj. Haarovská kaskáda a histogram orientovaných gradientů, což jsou učící se algoritmy, které v obraze hledají obecně objekty určitého charakteru (typicky například obličeje nebo postavy). Nakonec jsou ještě uvedeny metody porovnání popisů bodů a odhadu prostorové transformace pomocí množiny bodů identifikovaných mezi dvěma obrazy.

Jednou z možností využití bodových příznaků v digitalizovaném obraze je identifikace objektů v něm, která může být nasazena v bezpečnostních aplikacích, v případech, kdy je potřeba aby ovládací rozhraní systému identifikovalo uživatele, nebo například pro vyhledávání v databázi neoznačených fotografií a jejich přiřazování k sobě. Další možností je aplikace ve sledování objektů v obraze za účelem extrakce jejich pohybu, jejich počítání atd., rekonstrukce tvaru a charakteru prostředí, například za účelem pohybu a orientace v něm a lokalizace pozorovatele za týmž účelem.

V této práci jsou metody extrakce příznaků uvažovány zejmena v kontextu využití v algoritmu simultánní lokalizace a mapování (SLAM). Jedná se úlohu vytvoření mapy prostředí a zároveň určení pozice pozorovatele v tomto prostředí, přičemž sloučení těchto úloh do jedné je klíčem k jejich řešení. Metody extrakce příznaků jsou posuzovány v kontextu technické realizace tohoto algoritmu za využití jedné nebo více kamer snímajících prostředí a systému pracujícího v reálném čase. V tomto systému jsou v každém kroku algoritmu porovnány body nalezené v aktuálním obrazu s body nalezenými dříve a odhadnuta prostorová transformace (změna polohy pozorovatele), ke které muselo dojít mezi předchozím a aktuálním obrazem.

3 Přehled použitých metod

V této kapitole jsou popsány konkrétní metody detekce, popisu a asociace obrazových příznaků. U každé z nich je nastíněn kontext vzniku, popsán algoritmus fungování, vyzdvihnuty hlavní klady a upozorněno na nedostatky. Dále jsou uvedeny informace o využitelnosti a případná tvrzení autorů o výkonnosti metody doplněna o komentář zahrnující výsledky testů v kapitole 4.

3.1 Moravcův operátor

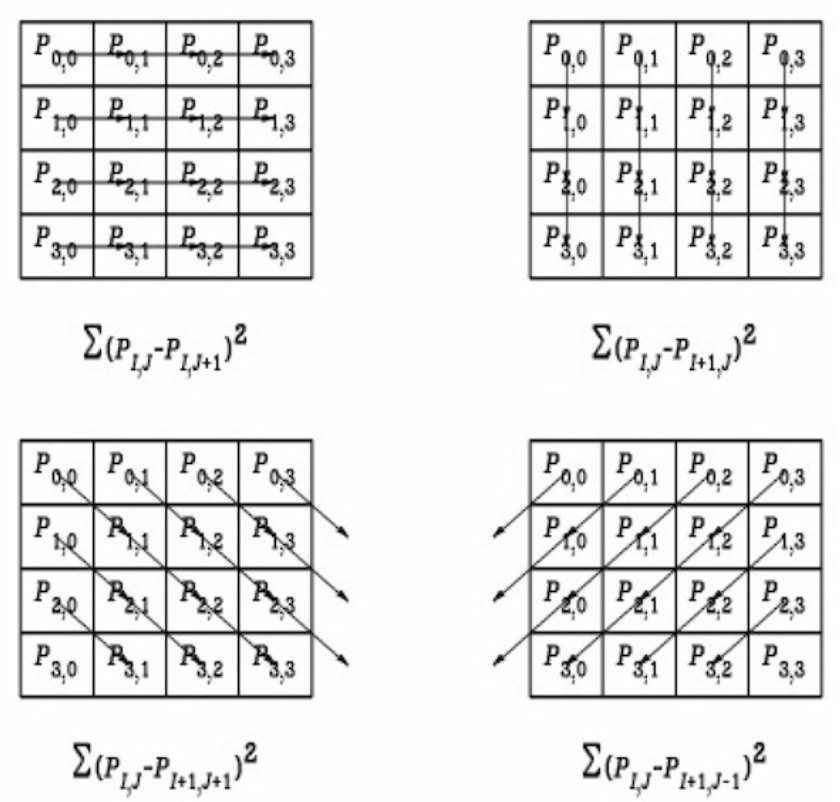
Jedná se o nejstarší a nejjednodušší uvedený operátor pro nalezení bodových příznaků [14]. Je uveden proto, že představuje ideový základ pro další uvedené operátory. Jeho principem je představa, že hledaný příznakový bod by měl vynikat ve všech směrech kolem sebe, tj. že by na všechny strany od něj měla být výrazná změna v jasu. Moravcův operátor je pouze detektor, deskriptor ani porovnávání bodů nejsou jeho součástí.

3.1.1 Algoritmus

Průměrná změna jasu v okolí bodu ve směru posunu (x,y) je definována jako:

$$\mathbb{E}_{x,y} = \sum_{u,v} w_{u,v} |\mathbb{I}_{u+x,v+y} - \mathbb{I}_{u,v}|^2, (x,y) \in \{(1,0),(1,1),(0,1),(-1,1)\}, \quad (3.1)$$

kde $\mathbb{I}$ je obraz ve formě jasových bodů, w je váhové okénko (algoritmus používá čtvercové binární), (u,v) je aktuální pozice v obrazu a (x,y) je aktuální posun. Provede se tedy posun všech obrazových bodů danými směry (obrázek **??**) a původní varianta se vždy odečte od posunuté. Tím je získána informace o změně jasu při posunutí ve směrech $\alpha \in \{0, 45, 90, 135\}$ stupňů. Poté je každé testované místo (každý pixel) překryto čtvercovým okénkem, hodnoty pod ním sečteny a výsledek umocněn na druhou. Pro každé testované místo tedy vzniknout 4 hodnoty čtverce změny v jednotlivých směrech. Ve zbytku algoritmu je pro každé místo vybráno $min\{\mathbb{E}_{x,y}\}$, v takto vzniklém obrazu se stanoví určitý práh a body s vyšší hodnotou, než je tento práh jsou



Obrázek 3.1: Vizualizace posunů v jenotlivých směrech. Zdroj: [14]

označeny jako výsledné příznaky. Jedná se tedy o nalezení bodů, které mají diskrétní diferenci v daných směrech nejméně takovou, jaká je odmocnina hodnoty prahu.

3.1.2 Využití

Moravcův operátor se stal základem dalších algoritmů. Přímo z něj vychází dále uvedené metody Harrisův operátor, Shi-Tomasi a se stejnou základní myšlenkou (bod odlišný od okolí nalezený pomocí numerické diference) pracují určitým způsobem všechny uvedené metody. Dnes se již prakticky nevyužívá, neboť má řadu nedostatků, zejména:

- Je anizotropní: uvažuje pouze změny v diskrétních úhlech, které jsou

násobky 45 stupňů. Změnám v ostatních úhlech bude logicky přiřazen nižší význam

- Binární čtvercové okénko výsledek výpočtu zašumuje
- Příliš citlivý: protože reaguje pouze na nejmenší změnu intenzity.

3.2 Harrisův operátor

Algoritmus Harris [11] vznikl snahou odstranit nedostatky Moravcova operátoru - pracuje se stejnou představou, ale efektivněji.

3.2.1 Algoritmus

Anizotropní vlastnosti lze odstranit pomocí Taylorova rozvoje:

$$\mathbb{E}_{x,y} = \sum_{u,v} \mathbf{w}_{u,v} |\mathbb{I}_{x+u,y+v} - \mathbb{I}_{u,v}|^2 \approx \sum_{u,v} \mathbf{w}_{u,v} |\mathbb{I}_{u,v} + x\mathbb{X} + y\mathbb{Y} - \mathbb{I}_{u,v}|^2, \quad (3.2)$$

$$\mathbb{E}_{x,y} = \sum_{u,v} \mathbf{w}_{u,v} |x\mathbb{X} + y\mathbb{Y}|^2, \quad (3.3)$$

kde (x, y) je vektor podle kterého se derivuje a $\mathbb{X}$ a $\mathbb{Y}$ jsou aproximované parciální derivace obrazu ve směrech osy x a osy y:

$$\mathbb{X} = \mathbb{I} \otimes (-1, 0, 1) \approx \frac{\delta \mathbb{I}}{\delta x}, \quad (3.4)$$

$$\mathbb{Y} = \mathbb{I} \otimes (-1, 0, 1)^T \approx \frac{\delta \mathbb{I}}{\delta y}, \quad (3.5)$$

Po roznásobení rovnice 3.3 dostaneme:

$$\mathbb{E}_{x,y} = \sum_{u,v} \mathbf{w}_{u,v} [x^2\mathbb{X}^2 + 2xy\mathbb{X}\mathbb{Y} + y^2\mathbb{Y}^2], \quad (3.6)$$

označme:

$$\mathbb{M} = \sum_{u,v} \mathbf{w}_{u,v} \begin{bmatrix} \mathbb{X}^2 & \mathbb{X}\mathbb{Y} \\ \mathbb{X}\mathbb{Y} & \mathbb{Y}^2 \end{bmatrix}, \quad (3.7)$$

výsledný vztah:

$$\mathbb{E}_{x,y} = [x, y]\mathbb{M}\begin{bmatrix} x \\ y \end{bmatrix}. \tag{3.8}$$

Tento vztah vyjadřuje velikost gradientu obrazu ve směru (libovolného) vektoru (x, y). Matice $\mathbb{M}$ je někdy nazývána Harrisova matice. Pokud je místo čtvercového okna nyní zvoleno gaussovské:

$$\mathbf{w}_{u,v} = e^{-(u^2+v^2)/2\sigma^2}, \tag{3.9}$$

je odstraněn první problém Moravcova operátoru: zanášení šumu do výpočtu nevhodným ("ostrým") okénkem. Směr a velikost derivací je nyní popsán elipsou ve formě matice $\mathbb{M}$, která respektuje celý kruhový prostor kolem bodu a velikost os této elipsy není závislá na rotaci obrazu. Velikosti os této elipsy jsou popsány velikostí vlastních čísel matice, tj. pro:

- $\lambda_1 \approx 0 \wedge \lambda_2 \approx 0$ - se v tomto bodě příznak nenachází, pro
- $\lambda_1 \approx 0 \wedge \lambda_2 \gg 0$ nebo naopak - tímto bodem prochází hrana a pro
- $\lambda_1 \gg 0 \wedge \lambda_2 \gg 0$ - se v tomto bodě nachází roh.

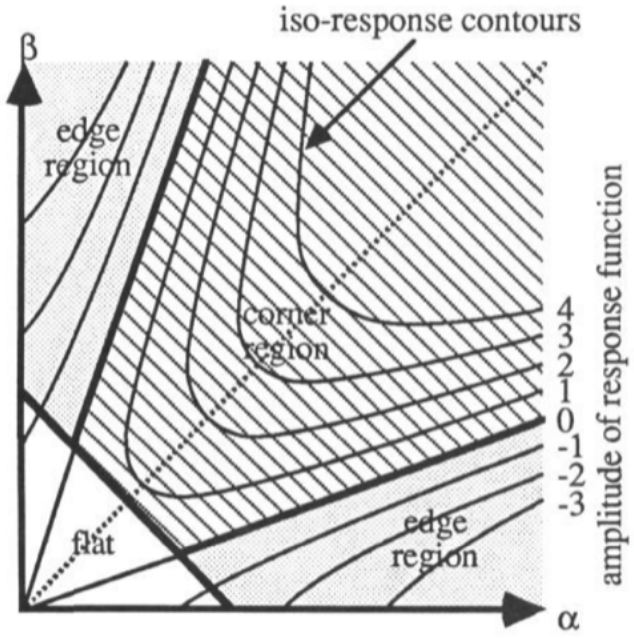
Protože výpočet vlastních čísel matice je relativně náročná operace, zatímco determinant a stopu matice 2×2 lze získat prostým odčítáním a násobením, původně Harris navrhuje tento výpočet aproximovat jako kritérium $\mathbb{R}$:

$$\mathbb{R} = \det(\mathbb{M}) - k * \text{stopa}(\mathbb{M})^2, \tag{3.10}$$

kde $\det(\mathbb{M})$ je determinant matice $\mathbb{M}$, $\text{stopa}(\mathbb{M})$ je součet prvků na hlavní diagonále čtvercové matice a k je nastavitelná konstanta, která určuje citlivost algoritmu na hrany. Jak je zobrazeno na obrázku 3.2, kritérium $\mathbb{R}$ bude nabývat hodnot:

- malých pro "plochou" oblast bez velkých změn
- kladných pro roh nebo jiný bodový orientační bod
- záporných pro hranu

Body se tedy označí porovnáním kritéria $\mathbb{R}$ se zvoleným prahem.



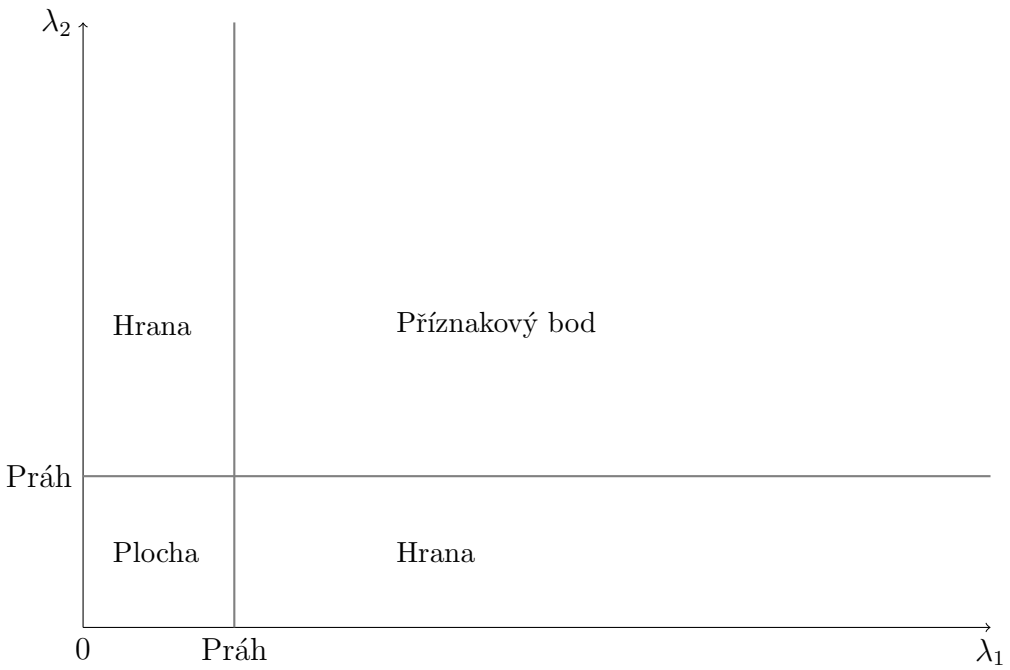
Obrázek 3.2: Vliv tvaru okolí bodu na rozložení vlastních čísel matice $\mathbb{M}$. Vlastní čísla λ_1 a λ_2 jsou tu označena α a β. Zdroj: [11]

3.2.2 Využití

V Harrisově operátoru jsou odstraněny hlavní nedostatky Moravcova operátoru. Zašumování výsledků je odstraněno nahrazením čtvercového okna gaussovským. Anizotropické vlastnosti a přílišná citlivost na nejnižší derivaci jsou opraveny reprezentováním derivací v bodě jako elipsy jejíž osy jsou určeny vlastními čísly matice $\mathbb{M}$ a aproximovány výše uvedeným vztahem pro kritérium $\mathbb{R}$. Jeho výhodou oproti dále uvedeným pokročilejším metodám je nenáročnost na výkon hardwaru.

3.3 Shi-Tomasi

Protože body nacházející se na hranách nebo liniích v obraze nejsou jako příznaky vhodné (body na hranách jsou logicky nejednoznačné a navzájem velmi podobné), navrhli Shi a Tomasi [19] vylepšení Harrisova algoritmu tak, aby detekoval pouze rohy. Toho lze dosáhnout pomocí výpočtu kritéria $\mathbb{R}$ přímo z vlastních čísel (vysvětlení viz 3.2.1).



Obrázek 3.3: Vliv velikosti vlastních čísel na nalezení příznakového bodu v metodě Shi-Tomasi

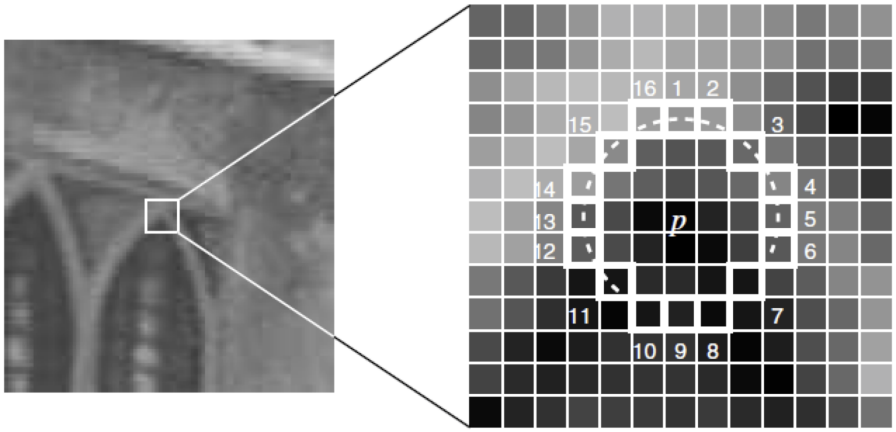
$$\mathbb{R} = min(\lambda_1, \lambda_2) \tag{3.11}$$

Prostor určený vlastními čísly $lambda_1$ a $lambda_2$ se tím značně zjednoduší, viz obrázek 3.3. Body se opět označí porovnáním kritéria $\mathbb{R}$ se zvoleným prahem. Výhodou tohoto řešení je zlepšení vlastností příznaků pro sledování za cenu mírného zvýšení výkonové náročnosti. Tento algoritmus bývá také nazýván Good Features to Track, zkráceně GFTT, podle názvu článku ve kterém byl původně popsán.

3.4 FAST

Jak název algoritmu napovídá, jedná se o rychlou a jednoduchou metodu nalezení bodových příznaků v obraze. Autoři uvádějí dvakrát větší rychlost, než vykazuje SIFT (sekce 3.5) a dokonce větší, než Harris. Obě tato tvrzení byla potvrzena v kapitole 4. Jedná se o čistou detekci příznaků, algoritmus

neobsahuje deskriptor nebo metodu porovnávání příznaků. FAST [16] je ale využíván detektorem systému ORB popsaným v sekci 3.8.



Obrázek 3.4: Extrakce FAST příznaku vyhodnocením bodů v jeho okolí. Zdroj: [16]

3.4.1 Algoritmus

1. Okolo testovaného bodu t se zkonstruuje kružnice sestávající z 16 bodů t_{1-16} (obrázek 3.4).

2. Je-li v kružnici n (doporučuje se $n = 12$) nebo více spojených bodů takových, že $|I(p) - I(t_i)| > T$, bod je označen za příznak. $I(.)$ je intenzita daného bodu, T je definovaný práh rozdílu intenzit.

3. Protože algoritmus má tendenci označit jako příznaky mnoho sousedících bodů, je vhodné po nalezení všech příznaků provést potlačení nemaximálních hodnot (ang. Non-maximum supression). To zajistí, že se z každé spojené oblasti sousedících příznakových bodů vybere jen ten, který má největší nebo nejmenší intenzitu a ostatní se zahodí.

Autoři dále navrhují vylepšení popsaného algoritmu testováním nejprve 2 nebo 4 pixelů v bodu 2 a pokračování pouze pokud tyto body splňují definovanou podmínku. Otázka, které body na pro tento test zvolit, je řešena nasazením neuronové sítě, která se na určitých datech natrénuje tak, aby tento test byl pro detekci příznaků co nejinformativnější.

3.4.2 Využití

FAST příznaky našly využití například ve SLAM systému PTAM (parallel tracking and mapping) a často jsou nasazovány v aplikacích na mobilních telefonech, jejichž výkon je oproti desktopům stále limitován.

3.5 SIFT

SIFT [12] je metoda vyhledání příznakových bodů v obraze spolu s výrobou deskriptorů pro jejich zpětnou identifikaci při dalším nalezení. Navazuje na předchozí algoritmy typu Harris, její výhodou je však větší robustnost vzhledem k zašumění obrazu, změnám osvětlení, afinním transformacím příznakových bodů a jejich pohybu v prostoru. Deskriptory se vyznačují velkou rozpoznatelností, tzn. různé body nebudou mít podobný deskriptor.

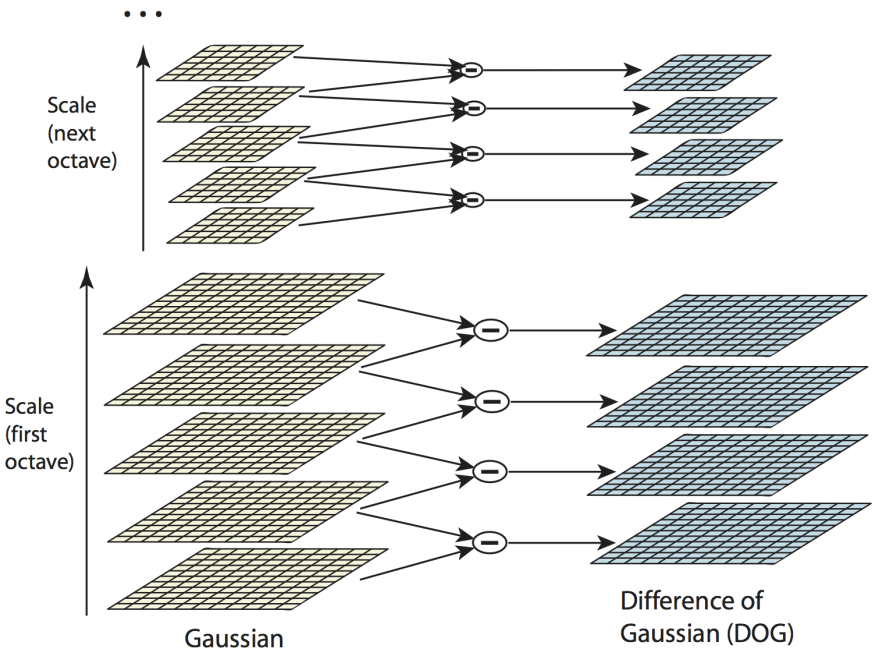
Lze ji využít nejen k identifikaci bodů pro prostorovou lokalizaci, ale též k robustnímu hledání definovaných objektů v obrazu, kdy objekty jsou reprezentovány množinou charakteristických bodů.

Algoritmus extrakce deskriptorů je uspořádán do kaskádovité struktury za účelem urychlení výpočtu: Složitější operace jsou umístěny co nejdále v algoritmu tak, aby byly aplikovány až po filtraci, tj na co nejmenší množství dat.

3.5.1 Algoritmus detekce příznaků

1. Hledání jasových extrémů v celém obrazu nezávisle na zvětšení, neboli měřítku: Provede se pomocí rozdílů gaussianů (Difference-of-Gaussians, DoG).

   Pracuje se s matematickou konstrukcí laplaciánu gaussiánů, což znamená rozdělení obrazu na pyramidu postupně více rozostřených verzí (rozostření se provede pomocí gaussovského jádra - masky se vzrůstajícím rozptylem σ - obrázek 3.5). Laplacián poskytne spojitou alternativu rozdílů mezi jednotlivými úrovněmi tohoto postupného rozostření. Tento postup zajišťuje, že najdeme-li někde v těchto "rozdílech" příznakový bod, nalezneme ho stejně i v jiném obrazu, kde se bude nacházet v jiném měřítku. V takovém obraze se bude příznak nacházet jinde



Obrázek 3.5: Tvorba DoG pyramidy. Zdroj: [12]

v tomto prostoru rozostření, ale bude vypadat stejně, což umožňuje porovnání nezávisle na měřítku.

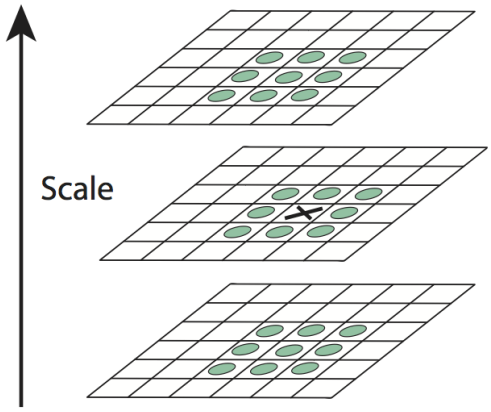
Protože výše popsaný postup platí pro spojitý prostor, při práci s diskrétním obrazem se aproximuje vytvořením pyramidy postupně více a více rozostřených vrstev, kdy tuto pyramidu ještě rozdělíme do oktáv (anglicky octaves). Nejvyšší rozostření oktávy v SIFTu by mělo mít oproti nejnižšímu dvojnásobný rozptyl σ. Jedná se o analogii hudebního názvosloví, kdy nejvyšší tón oktávy má oproti nejnižšímu dvojnásobnou frekvenci v hertzích. V další oktávě se též oproti předchozí pracuje s dvojnásobně redukovaným rozlišením obrazu (každý druhý řádek, každý druhý sloupec), protože se předpokládá, že dojde jenom k malé ztrátě informace při značném snížení výkonových nároků.

Tuto pyramidu gaussiánů doporučují autoři článku [12] konstruovat tak, že se skládá ze 4 oktáv po 5 rozostřeních (měřítkách, ang. scales), první rozostření se doporučuje $\sigma_0 = 1.6$ a rozdíly jednotlivých rozostření jsou $k = \sqrt{2} : \sigma_2 = k\sigma_1 = k^2\sigma_0$ atd. Podstatné je dodržet konstantní k mezi vrstvami.

Z této pyramidy gaussiánů se vytvoří pyramida jejich rozdílů prostým odečtením následujících vrstev v pyramidě a vznikne v úvodu zmíněný

DoG operátor.

2. Lokalizace klíčových oblastí: Na každé kandidátské lokaci se provede rozhodnutí o umístění a relativní velikosti oblasti. Klíčové oblasti jsou vybrány na základě měřítek jejich stability.



Obrázek 3.6: Testování příznaku v DOG pyramidě. Zdroj: [12]

Každý bod výsledné pyramidy z předchozího kroku se porovná s celkem 26 svými sousedy: 8 bezprostředně přiléhajícími ve své vrstvě a oknem 3x3, tedy 9 bodů na stejné lokaci ve vrstvě nad a pod, jak je zobrazeno na obrázku 3.6. Za kandidáta se bod označí v případě, že má ze všech těchto sousedů největší nebo nejmenší intenzitu. (v krajních vrstvách oktáv se nehledá, neboť jim chybí sousední vrstva nad nebo pod).

Kolem takového kandidátského bodu lze za účelem dosažení subpixelové přesnosti zkonstruovat trojrozměrnou plochu pomocí Taylorova rozvoje:

$$D(\mathbf{x}) = D(\mathbf{x_0}) + \frac{\delta D(\mathbf{x_0})}{\delta \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x^T}\frac{\delta^2 D(\mathbf{x_0})}{\delta \mathbf{x^2}}\mathbf{x}, \tag{3.12}$$

$$\mathbf{x} = [x, y, \sigma], \tag{3.13}$$

a za skutečné umístění příznaku označit její extrém, tedy bod, kde je derivace této funkce nulová:

$$\hat{\mathbf{x}} = -\frac{\delta^2 D(\mathbf{x_0})^{-1}}{\delta \mathbf{x^2}}\frac{\delta D(\mathbf{x_0})}{\delta \mathbf{x}}, \tag{3.14}$$

Pro další zpřesnění lze do rovnice 3.12 dosadit vypočtený bod $\hat{\mathbf{x}}$. Pokud má funkce v tomto bodě v kterémkoli směru větší hodnotu než 0.5, znamená to, že by se jako střed příznaku měl zvolit spíše bod, který se nachází tímto směrem.

Poslední operací tohoto kroku je vyřazení bodů, které se nacházejí na hranách, neboť ty nelze považovat za spolehlivé příznaky. Pomocí Hessovy matice 3.15 se vypočte zakřivení plochy 3.12 v okolí bodu a to se porovná s níže prahovým výrazem. Jde o podobný princip jako je eliminace hran v algoritmech Harris nebo Shi-Tomasi.

$$H = \begin{bmatrix} D_{xx} & Dxy \\ D_{xy} & Dyy \end{bmatrix}, \tag{3.15}$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1))}{r}, \tag{3.16}$$

r je volitelný práh. Jeho hodnotu autoři [12] doporučují 10.

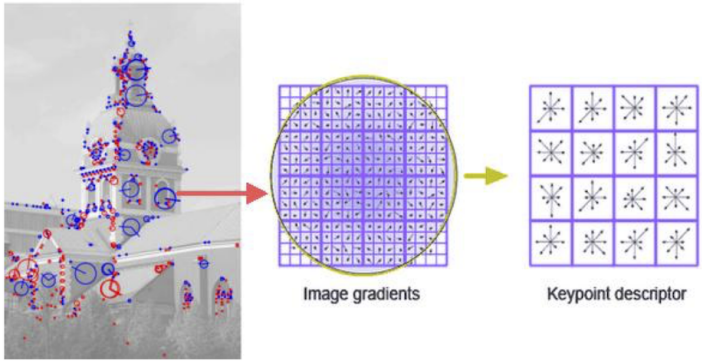
3. Určení orientace: Ke každé klíčové oblasti je přiřazena jedna nebo více orientací podle směrů lokálních gradientů. Další operace se provádějí na oblasti, která je transformovaná pomocí informací o relativní velikosti, umístění a orientaci aby bylo dosaženo nezávislosti deskriptoru na těchto vlastnostech

   Pomocí lokálních diferencí se určí velikost a směr gradientů zvoleného počtu bodů okolo nalezeného příznaku podle os x a y. Tyto gradienty se kvantizují do 36 kategorií po 10 stupních. Pak se stanoví maximum tohoto histogramu a pokud druhá nejvyšší hodnota histogramu dosahuje alespoň 80% hodnoty té nejvyšší, stanoví se pro tento příznak dvě orientace. Přesné úhly orientací se nakonec stanoví pomocí proložení kvadratické křivky maximem histogramu a jeho dvěma sousedy a maximalizací této křivky.

4. Výroba deskriptoru oblasti: Na každé dané oblasti jsou vypočteny lokální gradienty. Ty jsou převedeny do formy invariantního k deformaci tvaru obrazu a změnám osvětlení.

   V tomto bodě se vypočtou gradienty čtvercového okolí 16x16 pixelů okolo zvoleného bodu. Jejich velikosti se přenásobí gaussovým oknem pro zvýraznění těch, které jsou blízko středu. Tato oblast se poté rozdělí na 16 oblastí se 4x4 pixely. V každé této oblasti se vypočte histogram gradientů a nakvantizuje se do 8 kategorií podobně jako v předchozím kroku.

Výsledkem je SIFT deskriptor: 16 histogramů gradientů po 8 orientačních kategoriích (binech), tedy matice 4x4x8. Jako poslední krok se tento vektor normalizuje na jednotkovou délku aby se potlačily vlivy osvětlení.



Obrázek 3.7: Extrakce SIFT deskriptoru. Zdroj: [1]

Algoritmus vytváří velké množství příznaků, které hustě pokrývají celou plochu obrazu (cca 2000 příznaků na obraz 500x500px). Tyto příznaky je potřeba uchovávat v databázi a implementovat algoritmus pro její správu.

Výsledný deskriptor je trojrozměrná matice 4x4x8, příznaky se porovnávají pomocí algoritmu nalezení nejbližšího souseda, resp. jeho aproximace. Celý proces je zachycen na obrázku 3.7.

3.5.2 Porovnávání příznaků

Pro porovnávání příznaků autoři doporučují algoritmus nejbližšího souseda a pro větší databáze jeho aproximaci Best Bin First (sekce 3.15). Tento algoritmus dělí prostor příznaků na diskrétní disjunktní oblasti, odhaduje, ve kterém se hledaný prvek pravděpodobně nachází a hledá ho nejprve tam. Autoři uvádějí, že oproti standartnímu nejbližšímu sousedovi tento algoritmus zrychluje výpočet o dva řády za cenu pouze 5% ztráty přesnosti.

[1] `http://www.codeproject.com/KB/recipes/619039/SIFT.JPG`

3.5.3 Využití

SIFT je stále jedním z nejspolehlivějších algoritmů hledání a identifikace příznaků, což dokazuje i jeho široká využívanost v aplikacích identifikace objektů podle známých příznaků i v mapovacích algoritmech (Monoslam, [7]). I když ho lze využít pro mapování v reálném čase, je v porovnání s ostatními metodami velmi výkonově náročný. Jeho převaha nad ostatními uvedenými metodami s výjimkou SURF se zřetelně projevila při testech v kapitole 4.

3.6 SURF

Speeded up robust features neboli SURF [2] je metoda, která ideově navazuje na SIFT - jedná se o jeho aproximaci. Namísto konstrukce aproximace laplaciánu postupným rozostřováním jako v u SIFTu se předpokládá, že výskyt příznaků závisí čistě na determinantech Hessovských matic, jejíž hodnota se velmi rychle odhadne pomocí filtrací obdélníkovými filtry.

Při výrobě deskriptoru se namísto gradientů z lokálních diferencí použijí Haarovské waveletové filtry, z jejichž aplikací se odhadne orientace i přímo sestaví deskriptor. Podrobně se Haarovskými filtry a rychlým výpočtem odezvy na ně pomocí integrálního obrazu zabývá sekce 3.10 pojednávající o Haarovském detektoru objektů.

3.6.1 Algoritmus

1. Nejprve je zkonstruována aproximace laplaciánu - konvolucí daných diferenčních filtrů s postupně rostoucí velikostí, kdy jeden filtr reprezentuje D_{xx}, druhý D_{xy} a poslední D_{yy} je aproximována Hessova matice:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{3.17}$$

   Prvky jedné úrovně pyramidy se skládají z aproximace determinantu této matice: $det(H) = D_{xx}D_{yy} - (0.9D_{xy})^2$. Jednotlivé úrovně se potom liší velikostí filtru viz výše: začíná se na velikosti 9x9, která odpovídá $\sigma = 1.2$ v SIFTu. S každou vrstvou se k velikosti filtru přičítá fixní konstanta, začíná se na 6, což reprezentuje zdvojnásobení parametru σ

u SIFTu. Pyramida se opět dělí do oktáv. Změna velikosti filtru se mezi oktávami zdvojnásobuje.

2. Za příznak se opět označí extrém v tomto prostoru. Stejně jako u SIFTu je použito potlačení nemaximálních hodnot(non-maximum supression). Bod se porovná se svým okolím 3x3x3 v rámci pyramidy a za příznak je vzat v momentě, kdy je z něj nevětší nebo nejmenší.

3. Deskriptor je z okolí bodu syntetizován následujícím způsobem: Nejprve se vypočte reakce na Haarovy waveletové obdelníkové filtry ve směrech x a y. Označíme d_x, d_y. Výpočet probíhá na kruhovém okolí bodu s poloměrem 16σ, filtry mají velikost 4σ.

   Výsledek (d_x, d_y) je převážen gaussovskou maskou se středem v bodě příznaku a rozptylem $\sigma_G = 2.5\sigma$. Výsledný prostor je rozdělen na 16 úhlových výsečí po 60 stupních. Hodnoty v jednotlivých výsečích se sečtou a maximum těchto součtů určuje dominantní směr.

   Okolo bodu příznaku se extrahuje okno $20\sigma \times 20\sigma$. Toto okno se otočí o vypočtený dominantní úhel za účelem dosažení invariance k rotaci příznaku. Opět se vypočtou reakce na Haarovy waveletové filtry ve směrech x a y, ty se znovu převáží Gaussovskou maskou, tentokrát s rozptylem $\sigma_G = 3.3\sigma$.

   Toto okolí se rozdělí na výseče 4x4 body a pro každou se vypočte

$$v = [\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|], \tag{3.18}$$

   což je finální deskriptor.

3.6.2 Využití

Přestože je SURF původně navržen jako rychlá aproximace SIFT, v hodnocení detektorů v kapitole 4 měl vyšší průměrný čas detekce než SIFT, zato ale vykazoval čtyřikrát kratší celkový čas deskripce a dokonce překonal SIFT v celkovém hodnocení zejmena díky lepším výsledkům na datasetech s rotací.

3.7 BRIEF

Binary Robust Independent Elementary Features neboli BRIEF [5] je deskriptorový algoritmus, zabývá se tedy pouze popisem příznaků, nikoli jejic nalezením. Jeho principem je popis příznaků pomocí řetězce binárních hodnot. To je výhodné, protože takové řetězce lze porovnávat pomocí Hammingovy vzdálenosti (počet permutací, které je potřeba provést k přechodu z jednoho na druhý), což je zvláště na moderních procesorech velice rychlá operace. Oproti algoritmům jako je SIFT nebo SURF vyniká zejména jednoduchostí principu (a tím i implementace) a hlavně řádově vyšší rychlostí. Přitom při testech ale vykazuje podobné nebo větší množství správně identifikovaných příznaků jako SURF.

3.7.1 Algoritmus

Principem fungování algoritmu je porovnávání intenzit párů obrazových bodů (x, y). Neporovnává se každý s každým, ale v operátoru se stanoví mapa párů n_d. Autoři experimentálně stanovili jako nejlepší variantu navzorkování bodů z gaussovského rozložení $G(0, \frac{1}{25}s^2)$, kde S je strana čtvercového okna kolem místa výskytu příznaku.

1. Nejprve je vstupní obraz konvolvován čtvercovým gaussovským oknem velikosti 9x9 s rozptylem 2.

2. definujme funkci testu τ:

$$\tau(p, x, y) = \begin{cases} 1 & p(x) > p(y) \\ 0 & jinak \end{cases}, \tag{3.19}$$

   kde $p(x)$ je hodnota intenzity pixelu v okně kolem bodu výskytu příznaku velikosti SxS.

   Vektor příznaku (binární řetězec) je potom definován funkcí $f_{d_n}(p)$:

$$f_{d_n}(p) = \sum_{i=1}^{n_d} 2^{i-1}\tau(p, x_i, y_i) \tag{3.20}$$

3. Porovnání příznaků se provede pomocí Hammingovy vzdálenosti deskriptorových vektorů vypočtených v předchozím bodě a určením prahu pro souhlasící příznak

3.7.2 Využití

Podle autorů se jedná o rychlejší a stejně efektivní alternativu k SURF deskriptoru. Výhodou je, že tento algoritmus narozdíl od něj není licencován pro komerční využití. Jedná se o čistě deskriptorový algoritmus, ale je vhodné zmínit, že z BRIEF vychází deskriptor systému ORB. Při testování se potvrdilo, že skutečně vykazuje výrazně nižší časy detekce, ale platí za to řádově nižší výkonností (celková výkonnost okolo 13% naznačuje, že pro aproximaci prostorové transformace je v tomto nastavení na zkoumaném datasetu prakticky nepoužitelný).

3.8 ORB

Oriented Fast and Rotated Brief neboli ORB je algoritmus, který kombinuje FAST detektor příznaků a BRIEF deskriptor [17]. Zároveň do obou algoritmů přináší některá vylepšení, která mají za cíl především zajistit invarianci vůdči rotaci a maximálně zefektivnit mapu testování v BRIEF (sekce 3.7). Vznikl snahou tvůrců knihovny openCV poskytnout alternativu k SIFT a SURF, která by byla stejně efektivní, rychlejší a nepodléhala licenci pro komerční využití.

3.8.1 Algoritmus detekce příznaků

Oproti výchozímu algoritmu FAST je přidáno hodnocení kvality příznaků pomocí Harrisovského měřítka hranovosti a výpočet orientace příznaku.

1. Pro podpoření invariance k velikosti příznakové oblasti je zkonstruována scale space pyramida metodou popsanou v sekci o algoritmu SURF (diference gausialnů).
2. V této pyramidě jsou nalezeny FAST příznaky podle algoritmu popsaného v příslušné sekci této kapitoly. Z nich se vybere N nejlepších podle měřítka R Harrisova algoritmu.
3. Pro každý příznak jsou na kruhové oblasti kolem něj s poloměrem r vypočteny momenty jako $m_{p,q} = \sum_{u,v} u^p v^q I(u,v)$, kde $I(x,y)$ je intenzita obrazu v bodě (u,v). Z momentů je vypočten centroid oblasti

$C = (\frac{m_{1,0}}{m_{0,0}} \frac{m_{0,1}}{m_{0,0}})$. Orientace příznaku je určena směrem vektoru $\vec{OC}$ ze středu příznaku O do jeho centroidu. Jeho směr je vypočten jako $\theta = atan2(m_{0,1}, m_{1,0})$

3.8.2 Algoritmus popisu příznaků

Příznaky jsou popsány pomocí BRIEF deskriptoru, který navíc využívá informaci o orientaci příznaku získanou při detekci. Pro výrobu optimální mapy porovnávaných bodů je stanoven algoritmus strojového učení: Z trénovacích příznaků se vybírají takové páry bodů, které mají střední hodnotu porovnání co nejbližší $0,5$, co největší rozptyl a jsou co nejméně korelovány s ostatními vybranými páry.

1. Před samotným porovnáváním se obraz upraví nějakou vyhlazovací operací. Autoři doporučují integrál na okně 5x5.

2. matice testů

$$S = \begin{Bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{Bmatrix} \tag{3.21}$$

   se přenásobí vhodnou maticí rotace s úhlem θ vypočteným při detekci tak, že $S_\theta = R_\theta S$. Vznikne tak mapa porovnání invariantní k rotaci. Tento bod se implementuje pomocí kvantizace úhlů po 12 stupních a konstrukce lookup tabulky s předvypočtenými rotovanými mapami.

3. S touto mapou se na vyhlazeném příznaku provede výpočet 256 bitového deskriptoru BRIEF jak je popsáno v sekci, která je mu věnována.

3.8.3 Využití

Algoritmus ORB je alternativou k SIFT nebo SURF, která je při srovnatelné efektivitě podstatně rychlejší a nepodléhá licenci pro komerční využití. Při testování (kapitola 4) se jeho detektorová část umístila v celkovém hodnocení jako nejlepší, deskriptor ovšem vykazoval velmi slabé výsledky. Jako jedna z nejvýkonnějších testovaných kombinací se ukázala kombinace detektor ORB, deskriptor SURF.

3.9 MSER

Metoda maximálně stabilních extremálních oblastí neboli MSER [13] je relativně novým přístupem k detekci obrazových příznaků spočívajícím v identifikaci nikoli výrazných bodů, ale celých obrazových struktur. Obraz I je zobrazením $I : D \subset \mathbb{Z}^2 \rightarrow S$, kde D vyjadřuje dvourozměrnou celočíselnou polohu pixelu a S jeho intenzitu. Je-li S uspořádaná množina a existuje operátor A sousednosti dvou prvků v D: $A \subset D \times D$, lze v prostoru D definovat extremální oblast. Oblast Q je spojená, existuje mezi každými dvěma jejími prvky p, q cesta po jejích prvcích pomocí operátoru sousednosti. Vnější hranice oblasti Q, δQ, je složená z bodů, které neleží v oblasti Q, ale sousedí s bodem, který ano. Extremální oblast je taková, pro všechny jejíž body $q : q \in Q$ a body její vnější hranice $p : p \in \delta Q$ platí: $I(p) > I(q)$ nebo naopak. Maximálně stabilní extremální oblast je taková extremální oblast, pro kterou má v posloupnosti extremálních oblastí $Q_i : Q_i \subset Q_i + 1$ měřítko $q(i) = \frac{|Q_{i+\delta} \setminus Q_{i-\delta}|}{|Q_i|}$ minimum v bodě i, kde δ je volitelný parametr metody.

Nalezení MSER je tedy algoritmus dynamického prahování v obrazu, kdy je pro každou oblast obrazu zvolen práh, který je maximálně robustní vůči jeho změnám, tzn. při volbě prahu o něco většího nebo menšího než je, zvolený zůstane plocha a charakter nalezené oblasti maximálně podobný tomu, který je určen jako MSER.

3.9.1 Algoritmus

Popis příznaků vektorů

1. Nejprve je potřeba nalézt MSER oblasti. Ve zdrojovém obraze se seřadí všechny pixely podle hodnoty jejich intenzity. Ty se potom sestupně vkládají do obrazu. V každém kroku se updatuje datová struktura, ve které jsou zaneseny jednotlivé spojené komponenty a jejich plochy. Výsledkem tohoto postupu je množina komponent jako funkcí prahu. Pro každou komponentu je podle výše popsaného kritéria maximální stability nalezen práh. Ve výsledku je komponenta reprezentována hodnotou lokálního maxima intenzity a hodnotou ideálního prahu. Celý postup se provede na zdrojovém obraze i inverzi jeho intenzit (v článku označeno jako MSER+ a MSER-).

2. Pro každý extremální region jsou definovány oblasti jeho popisu. Jedná

se o elipsy opsané konvexnímu obalu (anglicky convex hull) oblasti. První ho přímo obepíná, další mají 1.5x, 2x, a 3x takovou velikost.

3. Každá elipsa z předchozího bodu je zpracována jako deskriptor: diagonalizuje se kovarianční matice (z elipsy se stane kruh). A otočí se podle dominantního úhlu z matice momentů (ta samá matice jako v Harrisově operátoru). Vznikne invariantní popis pomocí kruhu, který má stále stejnou orientaci bez ohledu na to, jak je původní elipsa nalezena.

Porovnání a identifikace příznaků

1. Pro příznakový kruh A z obrazu o_1 se snažíme najít odpovídající příznakový kruh B v jiném obrazu nebo databázi. Vzorek M_A^i z příznaku A porovnáváme s odpovídajícím vzorkem $M_{B_k}^i$, kde k je pořadí porovnávaných příznaků. Výsledkem porovnání je rozhodnutí ve tvaru ano - souhlasí, ne - nesouhlasí. Předpokládá se, že odpovídající oblast bude vykazovat vysokou míru souhlasících porovnání, kdežto výsledky porovnání nesouhlasící oblasti budou náhodné. Příznaky s největším počtem kladných hlasů jsou prohlášeny za kandidáty na shodu.

2. Kandidáti z předchozího kroku jsou s hledaným příznakem korelováni přes všechny úhly natočení - pokud korelace pod určitým úhlem do stanovené míry souhlasí, kandidát je prohlášen za vítěze - shoda je nalezena.

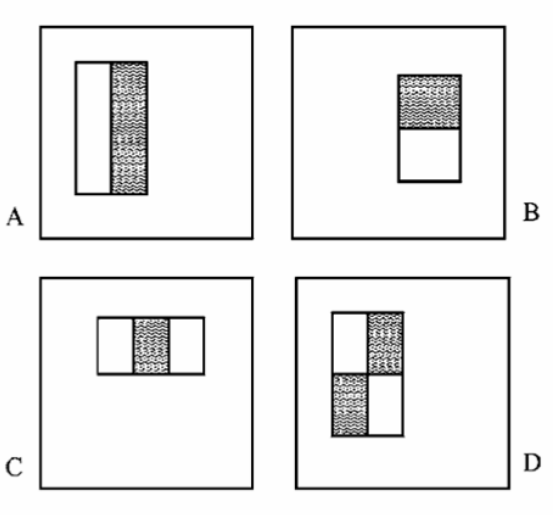
3. Z nalezených shod příznaků je možné pomocí RANSAC (viz dále v sekci 3.16) odhadovat fundamentální matici zobrazení jako odhad řešení přeurčené soustavy rovnic.

3.9.2 Využití

Algoritmus MSER vyniká velkou robustností příznaků umožňující znovunalezení příznaků ve velmi odlišných zdrojových obrazech (například ze značně rozdílných úhlů), ovšem platí za to značným výpočetním výkonem. V současné době je například využíván v systému rozpoznávání textu v obecném prostředí vyvíjeném na ČVUT [15]. Při testování (kapitola 4) se ukázal jako výkonnostně i časově podobný SIFTu.

3.10 Haar

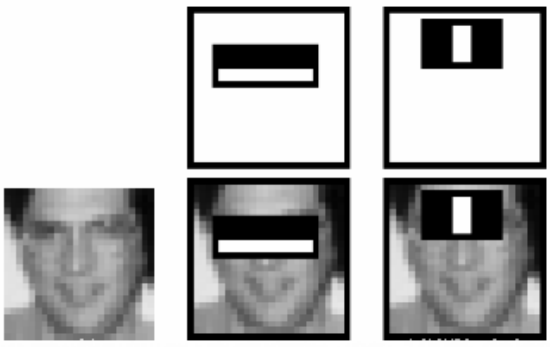
Pojem Haar nebo Haar algoritmus je v kontextu strojového vidění učící se algoritmus, který se využívá k detekci objektů v digitalizovaném obrazu [21]. Jeho principem je aplikace dvojrozměrných filtrů založených na Haarových bázových funkcích 3.8 na zkoumaný obraz (resp. výseč obrazu na které se provádí detekce - obrázek 3.9). Tím vznikne pro každý zkoumaný obraz velké množství příznaků (logicky násobně větší než je počet pixelů obrazu), které se liší svou diskriminační schopností.



Obrázek 3.8: Bázové filtry použité v Haar detektoru. Zdroj: [21]

Obecně jde ale o příznaky, jejichž průměrná chyba je jenom o o málo menší než 0.5, čili jsou jenom o málo lepším ukazatelem než náhodný odhad. Z této počáteční množiny příznaků se však pomocí algoritmu AdaBoost (sekce 3.10.2) vytvoří kaskáda detekčních vrstev se vzrůstajícím množstvím detektorů (detektor je zde klasifikátor založený na jednom z příznaků) a kvalitou dektekce tak, že nejdiskriminativnější příznaky jsou umístěny v prvních (nejdříve vyhodnocovaných) vrstvách. Pokud zkoumaný obraz neprojde jednou vrstvou detekční kaskády, další se již nevyhodnocují a celý obraz je zamítnut.

Tím je dosaženo vyloučení co největšího množství kandidátů za cenu co nejmenšího výpočetního výkonu. Autoři původního článku například uvádějí, že z asi 160000 možných klasifikátorů pro obraz 24x24 bodů sestavili kaskádu pouze 6000 z nich, ale průměrný počet vyhodnocovaných klasifikátorů na



Obrázek 3.9: Aplikace Haarových bázových filtrů na obraz při detekci
Zdroj: [21]

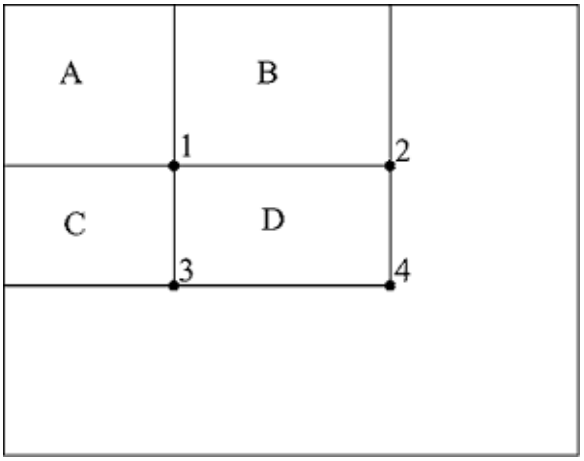
jednu zkoumanou výseč obrazu byl pouze 10. K efektivnímu výpočtu reakcí obrazu na Haarovy filtry je využita metoda integrálního obrazu.

3.10.1 Integrální obraz

Metoda integrálního obrazu reprezentuje každý bod obrazu jako součet odpovídajícího bodu zdrojového obrazu a všech bodů, které se ve směrech nacházejí nalevo a vzhůru od něj. Tato reprezentace se zde používá proto, že reakce na Haarovy filtry v bodě je dána sčítáním a odečítáním hodnot jasu obrazu v obdélníkových výsečích kolem bodu. Součet hodnot bodů v obdélníkové výseči obrazu je totiž dán součtem hodnot integrálního obrazu v pravém dolním a levém horním rohu obdélníku a odečtením hodnot integrálního obrazu v ostatních dvou rozích obdélníku viz obrázek 3.10. To pro počítání velkého množství součtů takových výsečí značně zrychlí výpočet, neboť integrální obraz se počítá pouze jednou.

3.10.2 AdaBoost

Příznak získaný reakcí na konkrétní haarův filtr na konkrétní pozici v obraze je označen t. Klasifikátor $F_t(x)$ založený na tomto příznaku se pro jeho obecně slabé diskriminační schopnosti nazývá slabý klasifikátor. T označuje kaskádu takových příznaků a příslušný klasifikátor na ní založený je nazván $F_T(x)$. Protože jeho diskriminační schopnosti jsou v žádoucím případě řádově vyšší než schopnosti slabého klasifikátoru, označuje se jako silný klasifikátor.



Obrázek 3.10: Výpočet ploch v integrálním obrazu. Součet hodnot pod plochou A je hodnota integrálního obrazu v bodě 1. Plocha B: hodnota v bodě 2 - hodnota v bodě 1. Plocha D: bod 4 + bod 1 - bod 2 - bod 3. Zdroj: [21]

AdaBoost je algoritmus, který z množství slabých klasifikátorů $f_t(x)$ zkonstruuje silný klasfikátor $F_T(x)$ jako $F_T(x) = \sum_{t=1}^{T} f_t(x)$. V každém kroku algoritmu je do silného klasifikátoru z množiny všech dostupných slabých klasifikátorů přidán jeden nový podle měřítka jeho kvality, jímž je celková chyba klasifikace na trénovacích datech.

Mějme trénovací data $(x_1, y_1), ..., (x_n, y_n)$, kde x_i je obraz a

$$y_i = \begin{cases} 1 & \text{v } x_i \text{ se nachází hledaný objekt} \\ 0 & \text{jinak} \end{cases} \tag{3.22}$$

Váhy $w_{1,i}$ (viz dále) se inicializují jako

$$w_{1,i} = \begin{cases} \frac{1}{2m} & \text{pro } y_i = 0 \\ \frac{1}{2l} & \text{pro } y_i = 1 \end{cases}, \tag{3.23}$$

kde m je počet negativních příkladů v trénovacích datech (obraz na kterém se objekt nenachází) a l je počet pozitivních příkladů.

Pro každé $t = 1, ..., t_{\max}$ se:

1. znormalizují váhy:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}, \tag{3.24}$$

   takže představují diskrétní pravděpodobnostní rozložení,

2. pro každý dosud nepoužitý příznak j (filtr s konkrétní velikostí na konkrétní pozici ve zkoumaném obraze) se vytvoří slabý klasifikátor $f_j(x)$. Ke každému takovému klasifikátoru náleží jeho chyba na trénovacích datech $\epsilon_j = \sum_i w_i |f_j(x_i) - y_i|$.

3. vybere se klasifikátor $f_t(x)$ s nejmenší chybou ϵ_t

4. aktualizují se váhy $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$, kde $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ a

$$e_i = \begin{cases} 0 & \text{pro } x_i \text{ správně klasifikované} \\ 1 & \text{pro } x_i \text{ špatně klasifikované} \end{cases} \tag{3.25}$$

5. finální silný klasifikátor je součtem všech dosud vybraných slabých klasifikátorů:

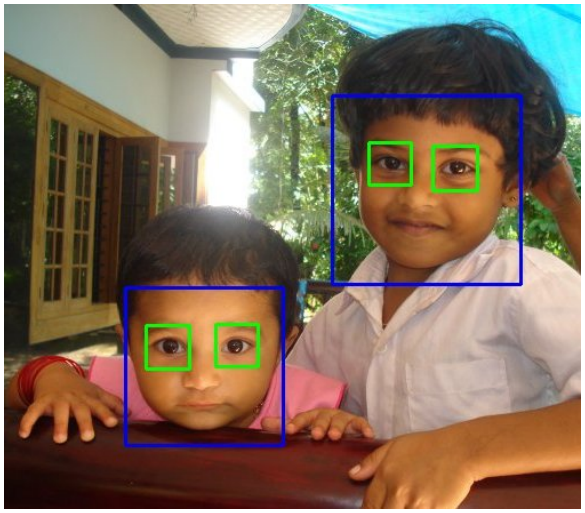
$$F_T(x) = \begin{cases} 1 & \text{pro } \sum_{t=1}^{T} \alpha_t f_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{jinak} \end{cases}, \tag{3.26}$$

   kde $\alpha_t = \log \frac{1}{\beta_t}$

3.10.3 Algoritmus trénování kaskády a detekce objektů

Pro hledaný objekt je pomocí AdaBoost natrénována klasifikační kaskáda z trénovacích dat, tzn. obrazů, ve kterých se nachází hledaný objekt a těch, ve kterých se nenachází spolu s touto informací. Jako příznaky se použijí reakce na obecně libovolné filtry, autoři článku používají obdélníkové filtry založené na haarových bázových funkcích aplikované na všechny dostupné velikosti a pozice těchto filtrů. K výpočtu těchto reakcí použijeme metodu integrálního obrazu popsanou výše. Strukturu kaskády (počet vrstev a počet klasifikátorů v nich) je třeba zvolit. Autoři doporučují 1, 10, 25, 25 a 50 klasifikátorů v prvních vrstvách a "postupně vzrůstající" počet klasifikátorů v dalších vrstvách s celkovým počtem klasifikátorů 6061.

Ve fázi detekce jsou zkoumaném obrazu vybírány výseče, a na každou z nich je aplikována kaskáda vytvořená v trénovací fázi. Reakce na filtry v



Obrázek 3.11: Příklad výsledku detekce pomocí Haar algoritmu. V tomto případě byly použity dva nezávislé detektory: jeden na detekci obličejů, druhý pro detekci očí. Zdroj: [2]

kaskádě je pro úsporu výkonu opět vypočtena pomocí předem vytvořeného integrálního obrazu. Příklad výsledku je vidět na obrázku 3.11

3.10.4 Využití

Hlavním pozitivem algoritmu je jednoznačně jeho rychlost. Už v roce 2001, kdy byl uveřejněn původní článek [21], bylo možné na tehdy běžném desktopovém PC (Pentium III 700 MHz) dosáhnout detekce v 15 snímkcích za vteřinu. Jeho hlavním problémem je potřeba pečlivě volit parametry při konstrukci kaskády v závislosti na konkrétní aplikaci tak, aby byly minimalizovány falešně pozitivní reakce a nedetekování objektů. Dalším problémem je častá několikanásobná detekce stejného objektu v jednom obrazu. Ten ale řeší algoritmus potlačení nemaximálních hodnot (non-maximum supression).

[2] `http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html`

3.11 Histogram orientovaných gradientů

Metoda histogramu orientovaných gradientů, též označovaná HoG [6] je metodou detekce objektů v digitalizovaném obrazu pomocí příznaků podobných SIFT deskriptorům. Klíčovým předpokladem metody je, že pro rozpoznání určitého tvaru v obrazu jsou klíčové hodnoty a směry gradientů obrazu, ale ne jejich přesné pozice. Stejně jako u Haar algoritmu se jedná o algoritmus trénovaný pomocí učení s učitelem.

3.11.1 SVM

Algoritmus HoG vzužívá učení a klasifikace pomocí algoritmu SVM neboli mechanismu podpůrných vektorů. SVM hledá v trénovacích datech nadrovinu, která co nejefektivněji rozdělí trénovací data (oddělí pozitivní příklady od negativních). Její důležitou součástí je jádrová transformace, která umožnuje transformaci zkoumaných vektorů do prostoru vyšší než původní dimenze, kde mohou být lineárně separabilní i ta data, která v původním prostoru nebyla. K nalezení optimální nadroviny stačí využít nejbližších dat z obou trénovacích množin. Tato data se nazývají podpůrnými vektory, odsud název metody.

V HoG se SVM trénuje ve dvou fázích. Nejprve se natrénuje na výchozích předklasifikovaných datech. Ve výsledcích klasifikace negativních příkladů jsou potom vyhledány případy falešně pozitivní detekce. SVM se potom natrénuje znovu s využitím těchto "těžkých negativních" příkladů.

3.11.2 Algoritmus výpočtu deskriptoru

1. Obraz se konvertuje do odstínů šedé
2. Zkoumaná obrazová výseč se rovnoměrně pokryje tzv. buňkami - menšími obrazovými výsečemi. Na těchto výsečích se vypočtou gradienty. Úhly těchto gradientů se kvantizují do 9 binů na rozmezí 0 až 180 stupňů, kde se v každém tomto binu nachází součet velikostí odpovídajících gradientů.
3. histogram gradientů v buňkách se normalizuje podle gradientů v odpovídajícím bloku, což je obrazová výseč větší než buňka. Normalizace

histogramu v proběhne pomocí L_2 normy:

$$v \leftarrow \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}}, \tag{3.27}$$

kde ϵ je malá konstanta, tzv. regularizace.

4. výsledným deskriptorem zkoumané obrazové výseče je spojení všech histogramů gradientů v buňkách na oblasti

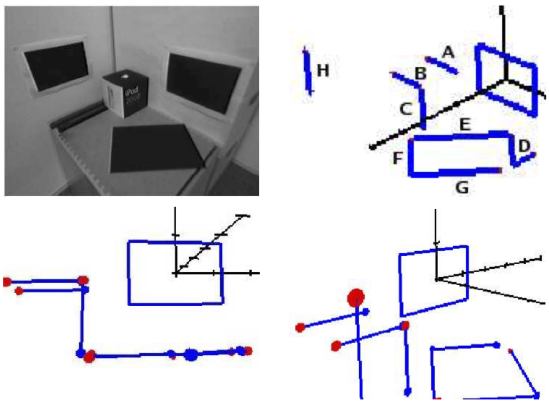
3.11.3 Využití

HoG je modernější alternativou k Haar algoritmu, která je též schopna fungování v reálném čase. Stejně jako Haar má také problémy s mnohonásobnou detekcí jednoho objektu, ale narozdíl od něj není citlivý na nastavení parametrů a tím pádem bez nutnosti zdlouhavého experimentáování dosahuje menšího množství falešných detekcí a nedetekovaných objektů.

3.12 Liniové příznaky

Tradiční detektory bodových příznaků předpokládají, že hrany nebo linie v obrazu nepředstavují vhodné orientační body a snaží se je z detekce vyloučit. Existují nicméně i přístupy, které se orientují právě na vyhledávání a reprezentaci linií. Pro tento přístup mluví například fakt, že zatímco bod, jenž je v obraze nějakým způsobem skryt (změnou podmínek, zastíněním objektem) nemůže být nalezen, dobře fungující detekce hran tuto hranu zrekonstruuje i pokud je viditelná pouze její část.

Liniové přiznaky jsou z důvodu větší míry nejistoty při hledání příznaku doménou především oblasti struktury z pohybu, neboli SFM, která řeší obecně problém 3D rekonstrukce a lokalizace, ale neklade si požadavky na fungování v reálném čase. Autoři [10] už ale na liniových příznacích staví celý realtimový SLAM systém. Obraz je nejprve předzpracován algoritmem zvýrazňujícím kontury, který je založen na numerických diferencích. Výstupem tohoto předzpracování je binární obraz (1 pro pixely, na kterými prochází hrana, 0 jinak). Na tento obraz jsou potom mapovány přímky pomocí lineární regrese. Hloubka detekovaných čar je potom odhadnuta pomocí kalmanova filtru (každý liniový příznak má svůj vlastní nezávislý filtr) a z nich se potom



Obrázek 3.12: Vizualizace běhu liniového slamu Zdroj: [10]

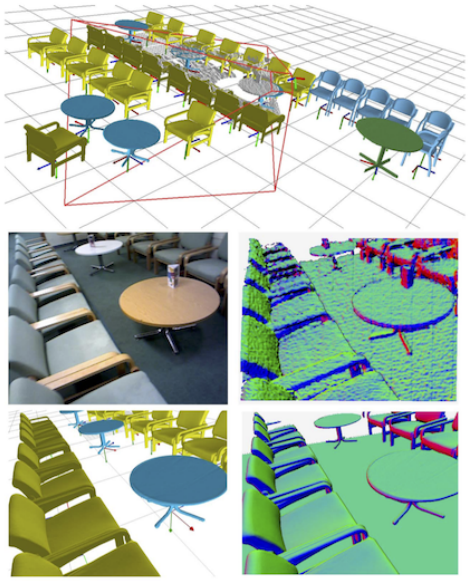
staví 3D model, oproti kterému se nově detekované čáry porovnávají. Autoři uvádějí uspokojivou výkonnost na malých datasetech (obrázek 3.12), jako všechny SLAM systémy se i tento potýká s problémem škálovatelnosti při omezených zdrojích výkonu a paměti.

3.13 Objektové příznaky

Detekci objektů v obraze lze snadno realizovat pomocí bodových příznaků (objekt je popsán body, které ho charakterizují), pomocí zmíněných metod Haar a HoG (sekce 3.10 a 3.11). To jsou ale postupy, jež jsou nejdříve natrenovány a poté hledají, co byly naučeny hledat. To není přístup, který by se hodil pro realtimovou lokalizaci v neznámém prostředí. Tak jako liniové příznaky, i objekty jsou spíše doménou SFM ([1]).

Z využití v reálném čase stojí za zmínku LSD SLAM [8]. Tato metoda odhaduje tvar 3D scény přímo z obrazových jasových dat, přičemž hloubku jednotlivých míst v obraze odhaduje. Obrazy a 3D scénu reprezentuje lieovou algebrou a optimalizuje pomocí podobnostních transformací v ní. Výsledem algoritmu je struktura, kterou si lze představit jako látku nataženou přes pozorované objekty. Jedná se vpodsatě o odhad tvaru povrchu z velkého množství bodových příznaků.

SLAM++ [18] je systém lokalizace a mapování založený na orientaci v prostředí osazeném známými objekty. Nejprve se stanoví databáze takových



Obrázek 3.13: Slam založený na detekci známých objektů: SLAM++ Zdroj: [18]

objektů a ty se potom v reálném čase hledají v hloubkové mapě získané pomocí senzoru jako je například Microsoft Kinect. V experimentech publikovaných autory článku vykazuje velmi přesnou a v čase stabilní schopnost lokalizace pozorovatele i tvorby mapy. Jeho nevýhodou je potřeba aproirní znalosti objektů, které v prostředí hledá. Metoda také v publikované fázi neuvažovala využití čistě obrazových dat.

3.14 K-Nearest Neighbours

K nejbližších sousedů neboli kNN je jedním z nejzákladnějších a nejjednoduších metod klasifikace dat. Namísto trénování diskriminačního modelu obvyklého u pokročilejších metod se ke klasifikaci testovaného vektoru používá přímo trénovací množina. Pro testovaný vektor se vypočte vzdálenost ke všem vektorům trénovacích dat a jeho příslušnost ke konkrétní třídě je určena na základě příslušnosti k jeho nejbližších sousedů. Metriku vzdálenosti je obecně možné zvolit jakoukoli, obvykle se používá euklidovská.

Primární výhodou metody je principiální i implementační jednoduchost. Protože výpočetní nároky se posouvají do fáze klasifikace (vzdálenosti je

nutno počítat pro každý vektor znovu), je zároveň možné přidávat do klasifikátoru data za běhu.

Nevýhodami jsou náročnost na výpočet i na paměť (je potřeba si při klasifikaci pamatovat potenciálně velmi rozsáhlou trénovací množinu).

3.15 Best Bin First

Best bin first, dále BBF [3], je algoritmus aproximující hledání k nejbližších sousedů. Jak je zmíněno v příslušné kapitole, náročnost klasifikace s použitím základního algoritmu kNN je z důvodu nutnosti vyhodnocení celé trénovací množiny pro každý zkoumaný vektor značná a přirozeně roste s rostoucí dimenzí prostoru příznaků (vektorů) a jejich množstvím. Oproti původnímu algoritmu kNN je BBF přesunutím části výpočetní náročnosti z fáze klasifikaci do fáze učení.

Při trénování modelu jsou vzorová naklasifikovaná data rozdělena do stromu. Na každé úrovni stromu se vždy rodičovský uzel rozdělí na dvě poloviny podle mediánu rozměru, na kterém mají data největší rozptyl, čímž vzniknou dva nové uzly (biny s vektory) se stejným počtem vektorů v každém z nich. Kořenem tohoto stromu je pochopitelně celá oblast $\mathbb{R}^n$, jeho listy jsou biny, které obsahují po jednom vektoru.

Při klasifikaci se tento strom prohledává tak, že se v každém nelistovém uzlu rozhodne o dalším postupu pomocí vzdálenosti zkoumaného vektoru a vektorů binů o kterých se rozhoduje nejblíže mediánu, podle kterého byla dotyčná úroveň stromu rozdělena. Protože se jedná o dobrou aproximaci binu, ve kterém se hledaný nejbližší soused (nebo nejbližší sousedé) skutečně nacházejí, lze omezit celkový počet listových binů které jsou během jedné klasifikace vyhodnoceny a tím výrazně uspořit výkon a dosáhnout zrychlení o 1-2 řády.

3.16 RANSAC

RANSAC, neboli Random Sample Consensus [20] je algoritmus vycházející z metody nejmenších čtverců. Ta řeší úlohu nalezení vztahu mezi daty z datasetu X jako kombinaci bázových funkcí pomocí minimalizace odchylky od tohoto předpokládaného vztahu (modelu). Matematicky se jedná o řešení

přeurčené soustavy rovnic.

Tento přístup předpokládá, že jsou-li data v X zatížena chybou, ta má nějaké vhodné statistické vlastnosti (typicky střední hodnotu 0) a její účinek se s přibývajícím množstvím dat vyruší.

To nemusí být nutně pravda. V případě přítomnosti chyby s nevhodnými statistickými vlastnostmi by bylo vhodné identifikovat data, na kterých se tato chyba projevuje a ty pro konstrukci modelu nevyužívat.

3.16.1 Algoritmus

1. Z X se náhodně vybere množství dat, které jednoznačně určí vztah dat jako kobinace daných bázových funkcí (Pro přímku v rovině například dva body).

2. Na zbytku dat z X se postupně provede konstrukce téhož modelu. Modely se porovnají a spadá-li jejich odchylka pod definovaný práh ϵ, jsou brány jako souhlasící, v opačném případě nesouhlasící.

3. Předchozí body jsou opakovány $k$krát. Na konci se vybere model s nejvíce hlasy (Nejvícekrát označen jako souhlasící). Pokud má tento model více souhlasících hlasů než je definovaný práh t, je označen za výsledek. Jinak algoritmus selhal.

Existuje vztah pro očekávaný počet opakování k pro nalezení m bodů spadajících pod odchylku ϵ : $E(k) = w^{-m}$, kde m je pravděpodobnost, že náhodně vybraný bod z X patří do hledaného modelu [9] .

Výhodou algoritmu oproti standartní metodě nejmenších čtverců je fakt, že data, která jsou označena jako nevěrohodná nebo zatížená chybou (produkují nesouhlasící modely) nejsou pro konstrukci výsledného modelu vůbec použita.

3.16.2 Využití

V diskutované oblasti se algoritmus RANSAC využívá především k odhadu fundamentální matice zobrazení nebo v tomto případě matice homografie mezi dvěma obrazy pomocí poloh párů přiřazených příznaků. Mimo to má

velmi široké využití kdekoli, kde je potřeba regresně odhadnout parametry modelu a je důvod se domnívat, že chyby, které na data působí nemají nulovou střední hodnotu a jiné ideální statistické vlastnosti.

4 Implementace a testování metod

Praktická část práce se zabývá porovnáním výkonnosti metod nalezení a popisu bodových příznaků na použitém datasetu. Je stanovena metrika výkonnosti a kombinace metod jsou testovány na výkonnost, rychlost detekce a počet nalezených bodů na celém datasetu a jeho částech.

4.1 Dataset

Obrázek 4.1: Subset Belledonne z datasetu - stejná scéna s postupně se zmenšujícím zoomem. Porovnává se vždy první obrázek vlevo nahoře s jedním z ostatních

Pro experimenty v této práci byl použit dataset volně dostupný na webu [1]. Všechny datasety na tomto webu byly prozkoumány skriptem `create_configs.py` a byly z nich vytěženy všechny páry obrázků, ke kterým je zadána zároveň

[1] `http://kahlan.eps.surrey.ac.uk/featurespace/web/related_papers/affine.html`

matice homografie (viz sekce 4.2). Výsledný dataset sestává z jednotlivých subsetů obsahujících vždy několik obrázků zobrazujících jednu scénu pod různými prostorovými transformacemi. Příkladem je subset Belledone na obrázku 4.1, nebo subsety Monet a Asterix, jejichž vždy jeden vybraný pár je vidět na obrázcích 4.6 a 4.3. Dále byly z této množiny vytvořeny subsety podle transformace, která se v nich odehrává. Některé subsety obsahují skutečnou prostorovou transformaci, tj. rotaci podle osy procházející středem fotoaparátu (subset rot), změnu úhlu pozorování (angle), nebo zoom, jiné jsou téměř nebo zcela statické a testují robustnost detektorů a deskriptorů vůči jiným transformacím: rozostření(blur), změnám světelných podmínek (light) nebo změně rozlišení obrazu(res). Porovnává se vždy jeden z obrázků s postupně všemi ostatními (obr. 4.1).

4.2 Homografie

Homografie [4], nebo také projektiní transformace je invertibilní transformacemezi dvěma projektivními pohledy (tzn. pohledy například fotoaparátu do 3D scény). Přímce z jednoho pohledu přiřazuje vždy přímku v druhém pohledu, bodu přiřazuje bod. Vyjadřuje tedy, jak se mění vjem pozorovaného předmětu v závislosti na změnách pozice, rotace nebo úhlu pohledu pozorovatele. Homografie je popsána transformační maticí $\mathbb{H}$ o rozměru 3×3. Pro transformaci bodu z jedné projektivní plochy na druhou $x_i \leftrightarrow x_i'$ platí:

$$x_i' = \mathbb{H}x_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix}, \tag{4.1}$$

kde souřadnice w' představuje měřítko. Matici homografie lze potom nalézt spojením těchto rovnic pro asociované páry nalezených bodů ve zdrojových obrazech a aproximací řešení přeurčené soustavy rovnic například metodou nejmenších čtverců nebo RANSAC.

Vzdálenost deklarované a nalezené homografie je v experimentech této práce brána jako měřítko kvality konkrétní metody nebo kombinace metod na daném datasetu. Kvalita homografie nabývá hodnot od 0 do 100% a vypočítává se jako:

$$pi_1 = H_1 * eig(H_1) \tag{4.2}$$
$$pi_2 = H_2 * eig(H_2) \tag{4.3}$$
$$dif = pi_1 - pi_2 \tag{4.4}$$
$$100 * (\frac{pi}{2} - atan(dif \times 10^{-}4)) \tag{4.5}$$

kde H_1 je homografie deklarovaná v datasetu, H_2 je matice homografie nalezená programem, $eig(H)$ jsou vlastní čísla matice H.

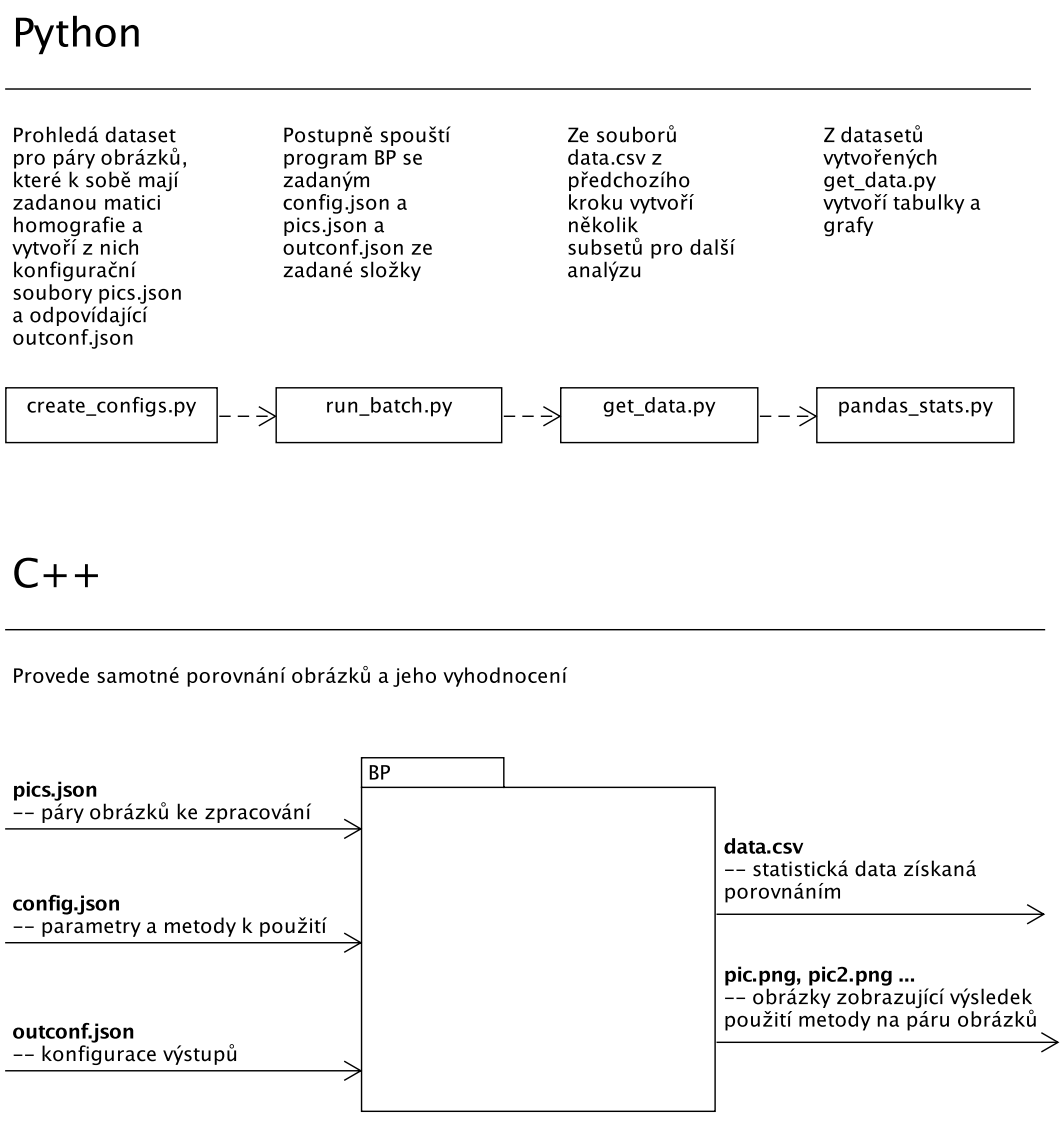
4.3 Implementace

Porovnání jednotlivých metod bylo implementováno v hlavním programu v C++ s využitím frameworku openCV. Zpracování datasetu, dávkové spouštění porovnání a statistické vyhodnocení výsledků bylo implementováno v jazyku Python s využitím knihovny Pandas. Schema implementace lze vidět na obrázku 4.2. Celá implementace společně se zdrojovými soubory pro tento dokument je k nalezení na githubu autora [2]

Data o souborech v datasetu jsou vytěžena skriptem `create_configs.py` v Pythonu a zkompilována do konfiguračních souborů pro hlavní program `BP`. Skript `run_batch.py` poté tyto konfigurační soubory načte a postupně s nimi spustí hlavní program. Ten pro každou vybranou složku datasetu vytvoří výstupní složku s obrázky, které zobrazují nalezené a spojené body mezi jedním a druhým obrázkem z vyhodnocovaného páru a soubor `data.csv`, který obsahuje informace o jednotlivých párech, rychlostech vyhodnocení a kvalitě odhadu homografie. Skript `get_data.py` ze souborů `data.csv` vytvoří jeden globální soubor a několik souborů se subsety podle transformace, kterou reprezentují: Úhel (ve smyslu změna polohy pozorovatele směrem do stran), rotace (okolo osy procházející středem fotoaparátu), zoom, nasvětlení, rozostření a změna rozlišení. Tyto soubory jsou zpracovány skriptem `pandas_stats.py` do obrázků a tabulek v této kapitole.

Hlavní program sestává ze čtyř tříd. První tři zajišťují obalení detektorů, deskriptorů a nástrojů výpočtu homografie z openCV tak, aby spolu všechny varianty vzájemně fungovaly a aby byly jednotlivé metody implementačně

[2] `https://github.com/PetrBarborka/BPrace`



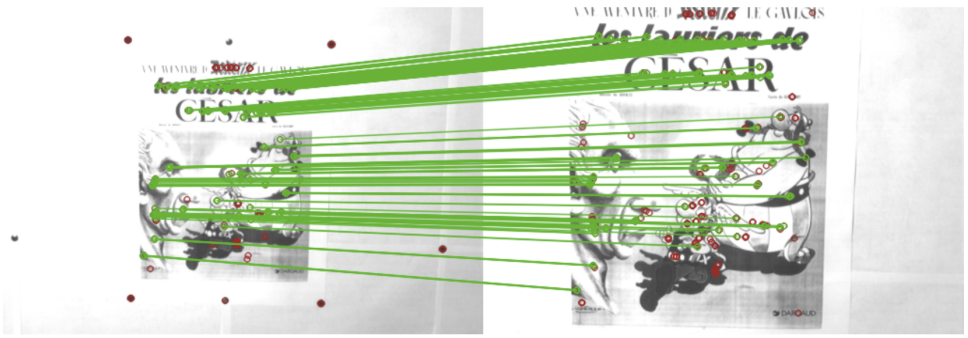
Obrázek 4.2: Schema implementace programů provádějících experimenty a jejich vyhodnocení

zaměnitelné. Poslední třída zajišťuje servisní funkce jako vstup a výstup a podobně. Tyto třídy jsou využity v hlavním souboru main.cpp, který zpracuje vstupní argumenty z příkazového řádku a spustí příslušné procesy. K práci s formátem json je využita knihovna Nielse Lohmanna (`https://github.com/nlohmann/json`).

Pythonové skripty procházejí sobourový systém pomocí `os.walk()` a vytvářejí a čtou soubory. Ve skriptu `run_batch.py` je k dávkovému spouštění programu `BP` použit modul subprocess, který umožňuje spustit libovolné množství instancí paralelně. Skript `pandas_stats.py` k práci s databází výsledků používá statistickou knihovnu Pandas.

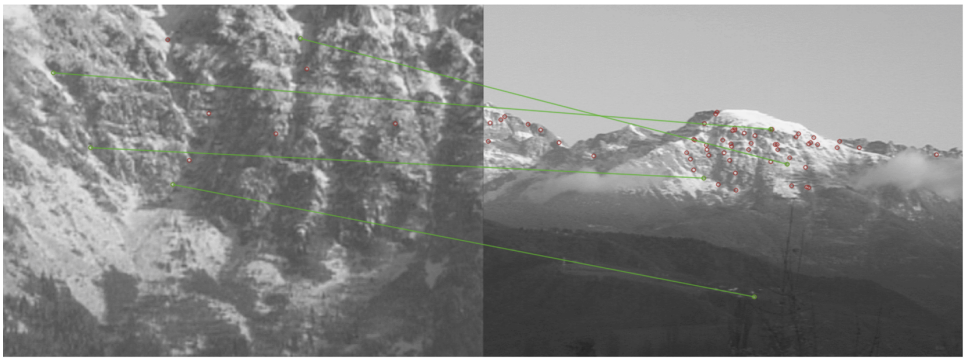
4.4 Experimenty

Na datasetu jsou zkoumány detektory příznakových bodů Harris, GFTT (neboli Shi-Tomasi), SIFT, SURF, FAST, ORB a MSER a deskriptory BRIEF, SIFT, SURF a ORB. Body nalezené a popsané těmito algoritmy jsou potom mezi jednotlivými obrazy přiřazeny a metodou na bázi RANSAC je z nich aproximována matice homografie. Jsou označeny body (páry bodů), které byly pro tuto aproximaci vzaty jako správné a ty, které byly zavrženy jako chybně přiřazené.

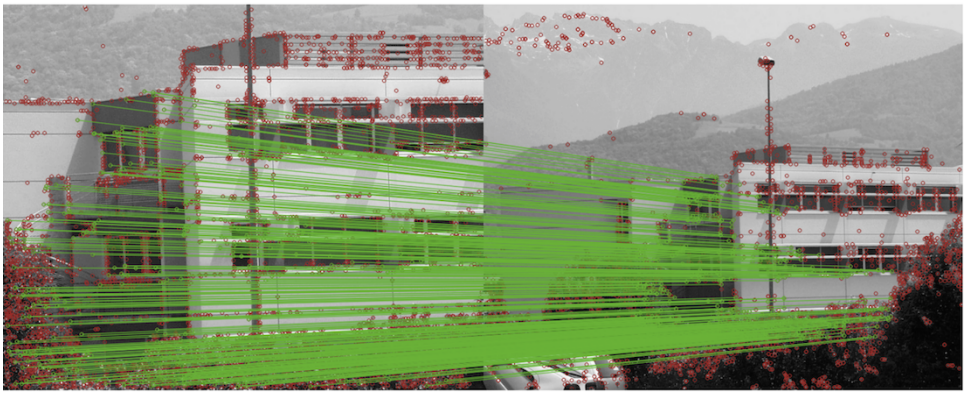


Obrázek 4.3: Transformace zoom ze subsetu Asterix, detektor MSER, deskriptor SIFT

V tabulkách 4.1 a 4.2 je uveden přehled celkových průměrných výkonností jednotlivých deskriptorů a detektorů. Tento přehled je získán vždy testováním uvedeného subsetu uvedenou metodou a všemi metodami z druhé kategorie. Tedy například skóre deskriptoru SURF je průměrem kombinace deskriptoru SURF a všech testovaných detektorů na daném datasetu. Jak je



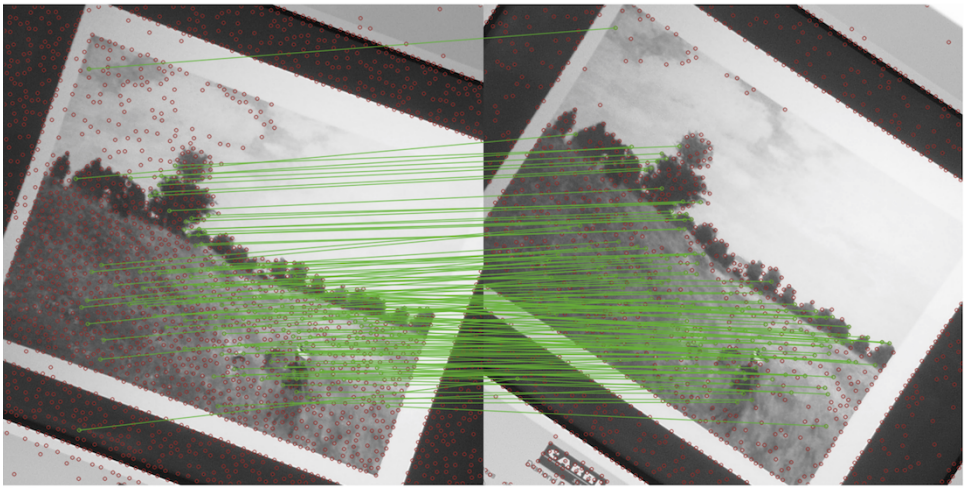
Obrázek 4.4: Ukázka transformace zoom ze subsetu Belledonne, detektor FAST, deskriptor ORB



Obrázek 4.5: Ukázka transformace zoom ze subsetu Ensimag, detektor i deskriptor SIFT

vidět v 4.1, v celkové výkonnosti vede detektor ORB. Při bližším pohledu vidíme, že exceluje zejména na subsetech blur (rozostření), light (změna světelných podmínek) a res (změna rozlišení). Z toho lze usoudit, že tento detektor založený na algoritmu FAST je vůdči těmto změnám parametrů obrazu velmi robustní. Za povšimnutí stojí, že jeho varianta - samostatná implementace algoritmu FAST v openCV má na všech subsetech asi poloviční hodnocení. Z toho je zřejmé, že se výkonnost detekčního algoritmu může drasticky změnit drobnými úpravami parametrů a vylepšeními aniž by se změnil jeho princip. Detektor SURF ve všech disciplínách překonal SIFT, přestože vznikl jako jeho aproximace.

Ve srovnání deskriptorů (tabulka 4.2) naopak ORB, založený na algoritmu BRIEF zaostává nad svou samostatnou implementací. Jako deskriptor má SIFT nad SURF převahu ve statických scénářích (subsety blur, light, res).



Obrázek 4.6: Ukázka transformace rotace ze subsetu Monet, detektor GFTT, deskriptor SIFT

Detektor	celkově[%]	zoom[%]	blur[%]	rot[%]	angle[%]	light[%]	res[%]
Harris	25.21	15.94	54.14	30.84	19.60	58.23	53.75
GFTT	24.18	16.83	49.85	30.33	18.95	55.30	45.88
SIFT	32.27	24.88	25.83	42.82	21.83	46.74	35.94
SURF	38.01	30.66	50.87	47.03	25.74	51.27	52.41
FAST	22.67	11.34	13.55	30.80	19.55	56.18	42.21
MSER	32.07	30.54	50.49	34.48	26.88	59.94	65.58
ORB	49.91	27.86	74.76	66.96	34.31	77.05	75.53

Tabulka 4.1: Celková výkonnost detektorů na datasetech

Deskriptor	celkově[%]	zoom[%]	blur[%]	rot[%]	angle[%]	light[%]	res[%]
BRIEF	13.55	9.39	14.14	17.77	9.68	19.78	22.63
SIFT	48.52	41.48	93.01	54.24	40.40	99.53	99.94
SURF	59.85	33.52	70.14	82.66	40.96	96.61	82.54
ORB	5.93	5.81	1.25	6.78	4.08	14.24	6.17

Tabulka 4.2: Celková výkonnost deskriptorů na datasetech

Ze srovnání kombinací (tabulky 4.4 a 4.3) je zřejmé, že všechny detektory mají nejlepší výsledky v kombinaci s deskriptory SIFT a SURF.

Při aplikaci v reálném čase na frekvenci 20Hz je na jeden celý cyklus uvažovaného systému k dispozici 0.05 vteřiny. Uvažujeme-li, že systém musí v každém cyklu provádět i jiné operace než detekci a popis příznaků, můžeme

Detektor	Deskriptor	celkově[%]	zoom[%]	rot[%]	angle[%]
Harris	BRIEF	4.05	6.87	2.64	4.20
Harris	SIFT	29.76	27.06	27.58	25.96
Harris	SURF	59.44	23.07	85.05	42.91
Harris	ORB	5.37	5.67	6.32	2.77
GFTT	BRIEF	6.43	7.51	6.78	8.46
GFTT	SIFT	27.78	29.19	23.44	27.98
GFTT	SURF	57.48	26.09	84.55	36.21
GFTT	ORB	5.04	4.54	6.54	3.16
SIFT	BRIEF	5.12	5.04	5.99	4.06
SIFT	SIFT	74.55	60.97	88.76	57.16
SIFT	SURF	44.17	27.33	70.25	23.61
SIFT	ORB	5.25	6.19	6.28	2.49
SURF	BRIEF	5.41	5.83	6.40	4.90
SURF	SIFT	72.50	61.26	87.39	49.20
SURF	SURF	69.15	51.04	87.98	45.67
SURF	ORB	4.96	4.52	6.35	3.21
FAST	BRIEF	4.68	3.95	5.91	2.56
FAST	SIFT	32.24	26.34	32.96	38.64
FAST	SURF	47.83	11.23	77.98	31.22
FAST	ORB	5.93	3.84	6.36	5.80
MSER	BRIEF	10.19	14.45	11.29	3.99
MSER	SIFT	38.91	50.45	33.41	41.05
MSER	SURF	69.33	45.26	83.92	53.92
MSER	ORB	9.86	12.01	9.32	8.58
ORB	BRIEF	58.42	21.80	85.56	38.69
ORB	SIFT	64.27	35.07	87.06	42.82
ORB	SURF	71.83	50.63	88.92	53.18
ORB	ORB	5.11	3.92	6.32	2.56

Tabulka 4.3: Celková výkonnost kombinací detektor -> deskriptor na dynamických datasetech

počítat s 0.025 vteřiny pro obě operace dohromady. Časy v tabulkách 4.5 a 4.6 představují dobu potřebnou pro nalezení příznaků v obou obrazech z testovaného páru, náročnost na jednom obraze bude tedy zhruba poloviční. Do této periody by se podle získaných dat žádná z kombinací zkoumaných metod nevešla. To je pravděpodobně způsobeno nedokonalým nastavením parametrů jednotlivých metod, vysokým rozlišením zpracovávaných obrazů a vysokým množstvím detekovaných příznaků (nebylo nijak omezeno), pro-

Detektor	Deskriptor	celkově[%]	blur[%]	light[%]	res[%]
Harris	BRIEF	4.05	2.23	11.70	3.10
Harris	SIFT	29.76	97.58	99.51	99.96
Harris	SURF	59.44	98.27	99.53	99.96
Harris	ORB	5.37	1.16	12.88	1.83
GFTT	BRIEF	6.43	1.70	12.68	2.20
GFTT	SIFT	27.78	97.42	99.49	99.82
GFTT	SURF	57.48	98.89	99.53	79.96
GFTT	ORB	5.04	1.40	9.49	1.54
SIFT	BRIEF	5.12	1.45	4.80	3.05
SIFT	SIFT	74.55	99.48	99.52	99.90
SIFT	SURF	44.17	0.98	79.18	37.37
SIFT	ORB	5.25	1.41	3.46	3.44
SURF	BRIEF	5.41	2.71	3.64	3.89
SURF	SIFT	72.50	99.64	99.51	99.99
SURF	SURF	69.15	99.62	99.51	99.80
SURF	ORB	4.96	1.53	2.40	5.96
FAST	BRIEF	4.68	2.28	3.60	2.25
FAST	SIFT	32.24	49.85	99.53	99.96
FAST	SURF	47.83	1.33	99.42	60.68
FAST	ORB	5.93	0.74	22.17	5.96
MSER	BRIEF	10.19	1.40	0.91	40.01
MSER	SIFT	38.91	99.80	99.69	99.99
MSER	SURF	69.33	99.64	99.59	99.99
MSER	ORB	9.86	1.12	39.58	22.33
ORB	BRIEF	58.42	99.76	99.52	100.00
ORB	SIFT	64.27	99.54	99.46	99.96
ORB	SURF	71.83	98.41	99.53	100.00
ORB	ORB	5.11	1.32	9.67	2.15

Tabulka 4.4: Celková výkonnost kombinací detektor -> deskriptor na statických datasetech

tože všechny porovnávané metody již byly nějakým způsobem v systémech pracujících v reálném čase nasazeny.

Z porovnání časů potřebných k detekci a popisu příznaků v tabulkách 4.5 a 4.6 lze vidět, že SIFT a SURF platí za svoji výkonnost o řád delším časem detekce před ostatními s výjimkou MSER a dokonce o dva řády delším časem výpočtu deskriptorů.

Detektor	průměrný čas detekce [s]
Harris	0.05
GFTT	0.05
SIFT	0.82
SURF	1.00
FAST	0.00
MSER	0.80
ORB	0.08

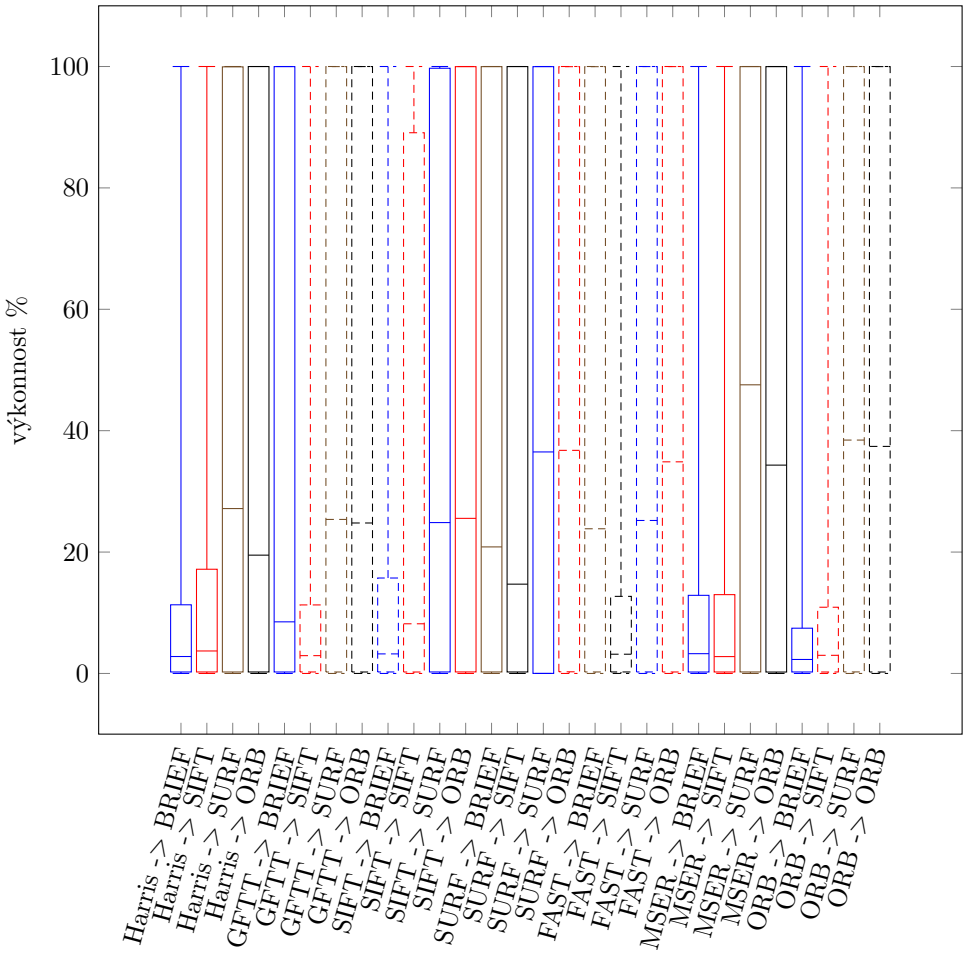
Tabulka 4.5: Průměrné časy detekce

Deskriptor	průměrný čas deskripce [s]
BRIEF	0.07
SIFT	8.23
SURF	2.97
ORB	0.07

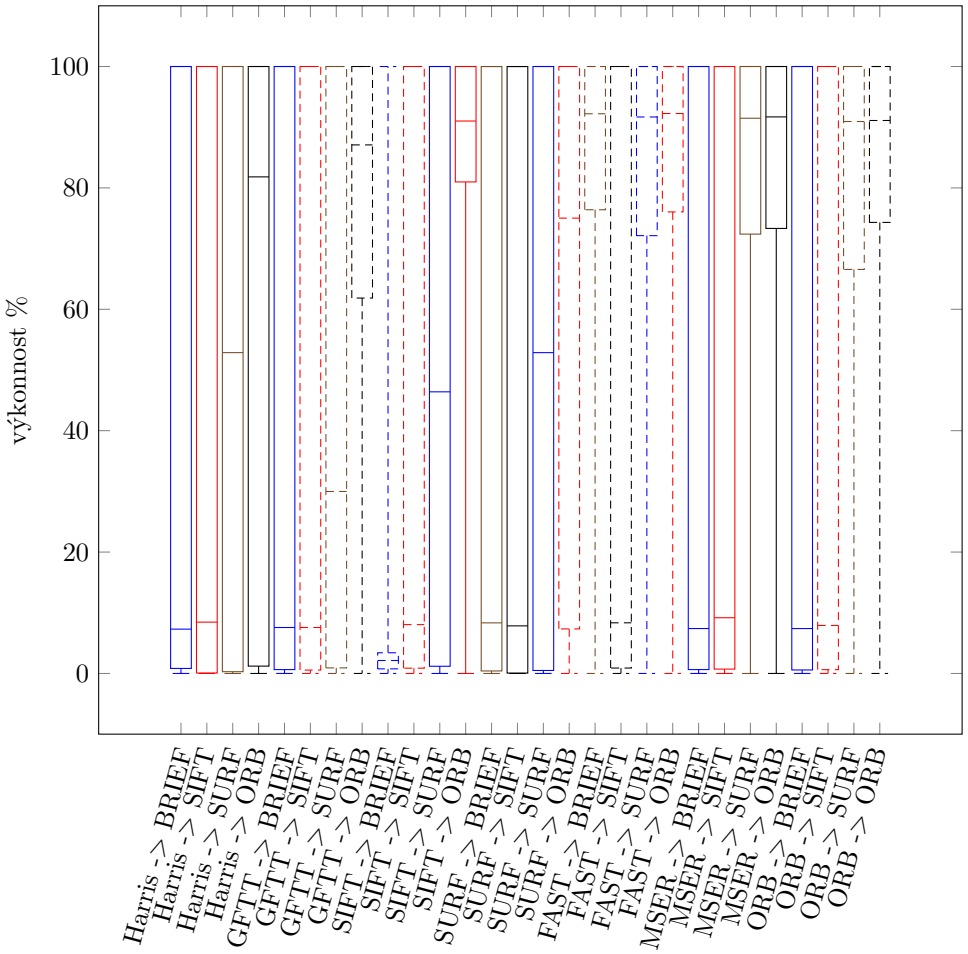
Tabulka 4.6: Průměrné časy deskripce

Dle tabulky 4.7 produkuje při daném nastavení největší množství příznaků detektor ORB. Lze ale také vidět, že množství detekovaných příznaků nemá přímou souvislost s kvalitou aproximace matice homografie.

Grafy 4.7 a 4.8 jsou boxploty zobrazující střední hodnoty, minima a maxima výkonností jednotlivých kombinací metod na subsetech Monet a Asterix. Vidíme, že Asterix byl pro všechny metody obecně náročnější. Potvrzuje se dominance SIFT a SURF, ale velmi slušných výsledků dosahují i body nalezené pomocí MSER a ORB.



Obrázek 4.7: Střední hodnota a standartní odchylka výkonnosti kombinací metod na datasetu Asterix (zoom)



Obrázek 4.8: Střední hodnota a standartní odchylka výkonnosti kombinací metod na datasetu Monet (rotace)

Detektor	Deskriptor	⌀ párů	⌀ použitých párů	⌀ skóre [%]
Harris	BRIEF	482.16	11.40	4.05
Harris	SIFT	554.30	84.87	29.76
Harris	SURF	554.30	143.08	59.44
Harris	ORB	481.38	19.59	5.37
GFTT	BRIEF	1264.33	46.27	6.43
GFTT	SIFT	1454.56	158.28	27.78
GFTT	SURF	1454.56	277.14	57.48
GFTT	ORB	1245.22	43.02	5.04
SIFT	BRIEF	3221.15	65.58	5.12
SIFT	SIFT	3603.22	1081.29	74.55
SIFT	SURF	3603.22	224.65	44.17
SIFT	ORB	3178.49	65.61	5.25
SURF	BRIEF	3340.22	66.86	5.41
SURF	SIFT	3532.28	867.44	72.50
SURF	SURF	3532.28	778.42	69.15
SURF	ORB	3295.36	65.12	4.96
FAST	BRIEF	881.80	21.88	4.68
FAST	SIFT	1010.68	178.34	32.24
FAST	SURF	1010.68	89.09	47.83
FAST	ORB	868.42	22.25	5.93
MSER	BRIEF	642.56	61.75	10.19
MSER	SIFT	748.09	215.44	38.91
MSER	SURF	748.09	258.98	69.33
MSER	ORB	629.85	61.63	9.86
ORB	BRIEF	7052.05	2479.68	58.42
ORB	SIFT	7052.05	1811.76	64.27
ORB	SURF	7052.05	2349.73	71.83
ORB	ORB	7052.05	128.17	5.11

Tabulka 4.7: Počty nalezených párů bodů

5 Závěr

V této práci byly teoreticky popsány metody detekce a popisu bodových příznaků v digitalizovaném obraze. Cílem teoretické části bylo poskytnou čtenáři přehled těchto metod spolu s vysvětlením principu jejic fungování. Tyto metody byly uvedeny od nejstarších a nejjednodušších, jako je Moravcův nebo Harrisův operátor, po modernější a komplexnější přístupy jako je SIFT, SURF nebo MSER. Pozornost byla věnována i moderním snahám o výkonovou optimalizaci této úlohy v algoritmech FAST, BRIEF a ORB. Dále byly zmíněny možnosti detekce objektů v obrazu pomocího se algoritmu Haar a jeho modernější alternativy metody histogramu orientovaných gradientů. V sekcích 3.12 a 3.13 byly zmíněny možnosti využití hran a objektů jakožto příznaků a příklady jejich nasazení v praxi. Popis metod obsahuje vždy algoritmus, vysvětlení jeho principu a případně popis využívaného matematického aparátu, jako je pyramida rozdílů gaussiánů (DoG) u metody SIFT, nebo integrální obraz a algoritmus AdaBoost u metody Haar. Teoretickou část uzavírájí algoritmy použité k další práci s nalezenými příznaky: hledání nejbližšího souseda a jedna z jeho možných aproximací Best bin first používané k přiřazování příznakových bodů. Nakonec je uvedena robustní regresní metoda RANSAC využívaná mimo jiné k odhadu vlastností geometrické transformace mezi dvěma obrazy.

Cílem praktické části práce bylo vybrané metody otestovat z hlediska množství nalezených příznaků, rychlosti jejich nalezení a popisu a kvality odhadu matice homografie. Jejím obsahem je popis využitého datasetu a testovaných prostorových transformací, vysvětlení pojmu homografie, algoritmus jejího hledání a výpočet měřítka, podle kterého je z ní určována kvalita jednotlivých metod. Praktickou část práce uzavírá popis softwarové implementace testovacího frameworku a okomentované výstupy z experimentů. Při souhrném testování byla zjištěna celková převaha metod SIFT a SURF nad ostatními potvrzující jejich robustností vůdči složitějším prostorovým transformacím jako je změna úhlu pozorovatele, nebo rotace.

V další práci by bylo možné se zaměřit na optimalizaci parametrů použitých metod na konkrétní datasety, neboť výkonnost těchto metod se v závislosti na parametrech výrazně mění a bylo by vhodné je kromě defaultního nastavení testovat i v nejlepším možném. Lze se zabývat i dosud nezmíněnými metodami jako je KAZE, AKAZE, atp. Poznatky o vlastnostech a fungování metod detekce příznaků je také možné využít při jejich praktické aplikaci v

jedné z oblastí zmíněných v úvodu, například při tvorbě systému simultánní lokalizace a mapování v reálném čase, nebo identifikačního systému v oblasti bezpečnosti.

Seznam obrázků

Literatura

[1] S. Y. Bao and S. Savarese. Semantic structure from motion. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2025–2032. IEEE, 2011.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[3] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[4] M. Berenda. Homografie a epipolární geometrie. *Trilobit [online] http://trilobit.fai.utb.cz/homografie-a-epipolarni-geometrie*, 2010.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[8] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] A. P. Gee and W. Mayol-Cuevas. Real-time model-based slam using line segments. In *Advances in Visual Computing*, pages 354–363. Springer, 2006.

[11] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[13] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[14] H. P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document, 1980.

[15] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012.

[16] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[18] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[19] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[20] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.