UNIVERZITY KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

REKONŠTRUKCIA OBJEKTOV S VYUŽITÍM VIACNÁSOBNÝCH POHĽADOV

Porovnávanie škálovo a afinne invariantných metód na hľadanie čŕt

2010

Igor Kolenič

UNIVERZITY KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

REKONŠTRUKCIA OBJEKTOV S VYUŽITÍM VIACNÁSOBNÝCH POHĽADOV

PODTITUL: POROVNÁVANIE ŠKÁLOVO A AFINNE INVARIANTNÝCH METÓD NA HĽADANIE ČŔT

Diplomová práca

Študijný odbor: 9.2.1 Informatika
Študijný program: Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: RNDr. JánLacko

Bratislava 2010

Igor Kolenič

Čestne prehlasujem, že predkladanú diplomovú prácu som vypracoval samostatne s použitím zdrojov uvedených v zozname na konci práce.

V Bratislave, Máj 2010 .....

Igor Kolenič

Ďakujem vedúcemu diplomovej práce RNDr. Jánovi Lackovi ako aj vedúcemu diplomového seminára Doc. RNDr. Andrejovi Ferkovi, PhD. za konzultácie a cenné rady počas práce. Zároveň ďakujem svojim blízkym a priateľom za psychickú podporu počas písania tejto práce.

Abstrakt

Názov: Rekonštrukcia objektov s využitím viacnásobných pohľadov

Podtitul: Porovnávanie škálovo a afinne invariantných metód na hľadanie čŕt

Autor: KOLENIČ, Igor .

Katedra: Katedra aplikovanej informatiky, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského, Bratislava

Vedúci diplomovej práce: RNDr. Ján Lacko

Počet strán: 62

Abstrakt: Cieľom tejto diplomovej práce je prezentovať prístupy – afinne a škálovo invariantné – k hľadaniu čŕt v obrázkoch pre ďalšie spracovanie, akým sú rozpoznávanie objektov, spájanie pre panorámy a pod., a implementovať porovnávací softvér a vybraté metódy na detekciu čŕt. Súčasťou programu je základná manipulácia s obrázkami pre získanie transformácií a deformácií obrázkov, ako sú rozmazanie, zmenšovanie a zväčšovanie, pridanie šumu, rotácie, zmena kontrastu a jasu a pod., na základe ktorých budú vyhodnocované a porovnávané metódy. Ako vybraté metódy boli použité SIFT, SURF a metóda na detekciu čŕt využitím rozkladu empirickým spôsobom.

Kľúčové slová: hľadanie čŕt, afinná invariancia, škálová invariancia

Abstract

Title: Object reconstruction using multiple images

Subtitle: Comparison scale and affine invariant methods for searching features

Author: KOLENIČ, Igor

Department: Department of applied informatics, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava.

Supervisor of master thesis: RNDr. Ján Lacko

Pages: 62

Abstract: Goal of this master thesis is to analyze approaches - scale and affine invariant - for features detection in images for further processing, as object recognition and reconstruction, panoramas stitching and more, and implement software for comparison of selected methods for features detection. There is basic handling with images included for obtaining needed transformations and deformations of images, as are blurring, scaling, rotation, noise adding, brightness and contrast change and others, to help evaluate and compare methods. Selected methods are SIFT, SURF and method Feature point detection utilizing the Empirical mode decomposition.

Keywords: searching features, affine invariance, scale invariance

Obsah

Úvod

V dnešnej dobe pri rozmáhajúcom sa trende informačných technológii (ďalej IT) a tiež rastúcej potrebe zábavnej IT, alebo tiež populárnej, sa používatelia snažia, čo najviac využiť silu počítačov na zautomatizovanie rôznych činností. Jednými z nich sú aj rozpoznávanie a rekonštrukcia objektov alebo tvorba panorám, ktoré využívajú pri predspracovaní metódy na hľadanie čŕt. Ich úspešnosť, kvalita, rýchlosť a použiteľnosť závisí od použitia vhodných metód pre použité typy obrázkov. Preto sme sa rozhodli pre ich porovnanie vzhľadom na rôzne kritéria.

Práca je delená do 4 kapitol.

V prvej sme sa zamerali na prehľad existujúcich metód pre hľadanie významných čŕt v obraze. Ide o ich vymenovanie, popis myšlienky algoritmu a ich použitia pri rekonštrukcii objektov.

V druhej kapitole sa nachádza motivácia pre výber konkrétnych metód, ktoré sme sa rozhodli porovnať. Každá z nich je osobitne detailne rozobratá.

V tretej kapitole sa nachádza softvérová špecifikácia, motivácia pre použitie programovacieho jazyka, operačného systému (ďalej OS), pre ktorý je aplikácia určená, ako aj jednotlivé štruktúry a návrh aplikácie, užívateľské rozhranie (ďalej GUI) a používateľská príručka k aplikácii.

V štvrtej kapitole sú uvedené výsledky z testovania vybraných metód a ich porovnanie.

1 Všeobecný prehľad problematiky

1.1 Rekonštrukcia objektov

Rekonštrukcia objektov je dnes pomerne zaujímavou a rýchlo sa rozvíjajúcou časťou v oblasti spracovania obrazu a počítačového videnia. Hlavným jej cieľom je získať z dostupných dát, ktorými môžu byť obrázky, video (sekvencia obrázkov), dáta z rôznych senzorov a špeciálne upravených zariadení a iných, priestorový model reálneho objektu zobrazeného v spomenutých dátach. V literatúre nájdeme mnoho prístupov k danej problematike, nakoľko vzhľadom na neobmedzenú variabilitu objektov a scén, v ktorých sa objekty nachádzajú, je existencia všeobecného jednotného prístupu veľmi otázna, ak nie nemožná.

Pri využití techniky rekonštrukcie objektov z viacnásobných pohľadov sa dá postup získania objektov rozdeliť na fázy:

1. detekcia čŕt
2. problém korešpondencie obrázkov - párovanie získaných bodov v obrázkoch
3. výpočet fundamentálnej matice reprezentujúcej epipolárnu geometriu medzi obrázkami
4. rekonštrukcia objektu

Tieto fázy značne závisia od zvoleného prístupu na základe dostupných informácii, ktorými môžu byť pozícia a orientácia fotoaparátov, ich vzdialenosť od objektu, ich vzájomná vzdialenosť, kalibračné nastavenia, ohniskové vzdialenosti a pod. Podľa spomenutých kritérií sú základné prístupy rozdelené na [Baumberg00]:

- **Calibrated Short Baseline Stereo**

  Dva fotoaparáty (kamery), ktorých konfigurácia je známa, sú umiestnené v relatívne blízkej vzdialenosti k sebe vzhľadom na ich vzdialenosť od pozorovaného objektu. V tomto prípade sa problém korešpondencie zjednodušuje spôsobom, že bod v prvom z obrázkov korešpondujúci bodu v druhom obrázku leží na známej epipolárnej čiare. Nevýhodou v tomto prípade väčšinou býva, že konfigurácia kamier je značne obmedzená.

- **Uncalibrated Short Baseline Stereo**

  Jeden fotoaparát (kamera) je posúvaný takým spôsobom, že posun a zmena orientácie je pomerne malá, ale ju nepoznáme. Typickým príkladom je video sekvencia. Dôsledkom toho je, že body zobrazené v jednom zábere sú na ďalšom snímku posunuté len o malú vzdialenosť. To uľahčuje problém korešpondencie spôsobom, že bod v nasledujúcom zábere korešpondujúci bodu v aktuálnom zábere leží v malom okolí pôvodného bodu. Nevýhodou v tomto prípade väčšinou býva fakt, že konfigurácia kamier je značne obmedzená (v prípade kamery jej pohyb).

- **Uncalibrated Wide Baseline Stereo**

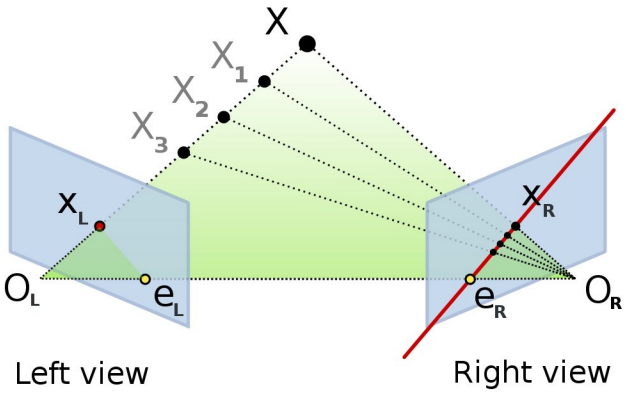
  Vzdialenosť fotoaparátov je v tomto prípade podstatná vzhľadom na pozorovanú scénu a neznáma, čoho dôsledkom je, že epipolárna geometria je neznáma tiež a nutnou a hlavnou úlohou je v tomto prípade jej výpočet na základe korešpondencie bodov. Pri tomto kroku je dôležité, aby v obidvoch obrázkoch boli nájdené body prislúchajúce totožnému bodu v pozorovanej scéne a následne ich párovanie. Na eliminovanie podstatného podielu nesprávne spárovaných bodov je možné použiť štatistické metódy ako napr. RANSAC [Fischler81], kedy ale s väčším počtom nezhôd rastie výpočtová náročnosť. Tento prístup však zlyháva v prípade veľkých rozdielov v pozíciách alebo parametroch fotoaparátov.

1.2 Epipolárna geometria a fundamentálna matica

Epipolárna geometria je vnútornou projektívnou geometriou medzi dvoma pohľadmi. Nezávisí na štruktúre pozorovanej scény, ale len na vnútorných vlastnostiach fotoaparátov a ich vzájomnej pozícii. [Hartley04]

Fundamentálna matica $\boldsymbol{F}$ je potom algebraická reprezentácia epipolárnej geometrie. $\boldsymbol{F}$ je homogénna matica 3x3 s hodnosťou 2, pre ktorú platí, že ak x_L je bodom v prvom obrázku a x_R a je korešpondujúcim bodom x_L v druhom obrázku, tak

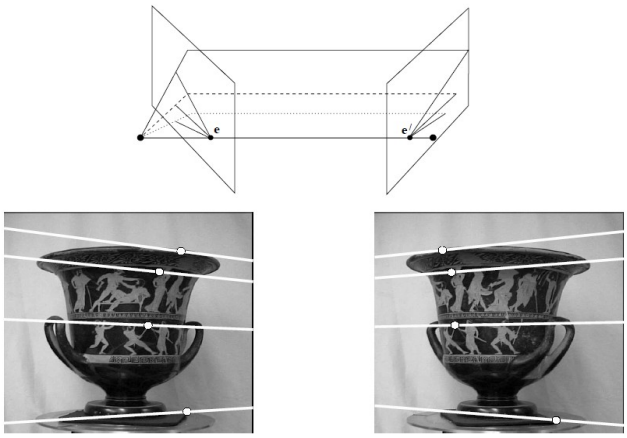
$$x_R^T F x_L = 0 \qquad (1.1)$$



Obrázok 1.1: **Epipolárna geometria**

O_L, O_R sú ohniská fotoaparátov, bod X_L je priemetom bodu X na ľavý pohľad a obdobne X_R na pravý pohľad, e_R, e_L sú epipóly - priesečníky zodpovedajúcej priemetne a základne (spojnica ohnísk fotoaparátov), $e_R X_R$ - epipolárna čiara zodpovedajúca X_L (prevzaté z [WikiEG])

Transformácia Fx_L definuje epipolárnu čiaru $e_R x_R$ v druhom obrázku zodpovedajúcu bodu x_L (viď obrázok 1.1). Na tejto čiare ležia všetky priemety bodov reálnej scény do druhého obrázku, ktoré ležia na spojnici bodu X a O_L a teda v prvom obrázku sa zobrazujú do jedného bodu a to do X_L. Fundamentálna matica má 7 stupňov voľnosti a det(F) = 0, keďže má hodnosť 2. Získaná môže byť z matíc zobrazenia fotoaparátov alebo zo známych korešpondujúcich bodov v obrázkoch, pričom je potreba aspoň 7 párov bodov[Hartley04] . Ďalšie vlastnosti ako aj výpočet je veľmi dobre popísaný v [Hartley04].



Obrázok 1.2: **Korešpondujúce body a epipolárne čiary** (prebraté z [Hartley04])

Spodný rad: korešpondujúce body a príslušné epipolárne čiary.
Horný rad: náčrt vzájomnej polohy obrázkov a epipolárnych rovín

1.3 Škálový priestor (Scale-space)

1.3.1 Motivácia

Teória škálového priestoru bola vyvinutá v okrem oblasti spracovania obrazu aj v iných oblastiach. Pri práci s obrazom pomáha identifikovať jednotlivé časti, objekty zobrazované na obrázkoch. Rovnaký objekt z reálneho sveta môže pozorovaný z rôznych vzdialeností, kedy objekt zosnímaný pri menšej vzdialenosti k pozorovateľovi sa javí ako väčší ako keď ho zosnímame z väčšej vzdialenosti. Opačne, pri pozorovaní danej scény v rôznych mierkach vystupujú do popredia iné úrovne detailov - abstrakcii. Napríklad o detaile listu stromu ma zmysel sa baviť pri pozorovaní v mierke centimetrov až pár metrov, kde naopak pri vzdialenostiach kilometer alebo nanometer je to zbytočné, a relevantnejšie je sa baviť o lese, resp. o molekulách [Lindeberg09].
Pri spracovaní obrazu však nemáme, aspoň tak predpokladáme, žiadne informácie o úrovni abstrakcie objektov na obrázku, preto vhodným riešením je práca v rôznych škálach súčasne, tzv. multi-scale reprezentácia, kedy štruktúry na pre jemnejšie škály sú postupne potláčané [Lindeberg94]. Na to sa využíva postupné rozmazávanie obrázka (Obrázok 1.3).
Výsledkom mnohých prístupov je, že konvolúcia Gaussovým jadrom a jeho deriváciami poskytuje zovšeobecnenú triedu obrazových operátorov s jedinečnými vlastnosťami pre tento účel. Podmienkami, ktoré špecifikujú unikátnosť sú linearita a invariancie priestorového posunu, kedy pri postupe z menšej škály k väčšej nesmú vzniknúť nové štruktúry [Lindeber98].

1.3.2 Definícia

Definícia škálového priestoru je platná pre N-dimenzionálne zobrazenia, avšak kvôli

jednoduchosti a keďže vo väčšine prípadov sa jedná o dvojrozmerné obrázky uvádzame dvojrozmernú verziu.

Nech f: $\mathbf{R}^2$->$\mathbf{R}$ reprezentuje dvojrozmerný obrázok, potom

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (1.2)$$

$$L(x, y; \sigma) = (g(., .; \sigma) * f)(x, y) \quad (1.3)$$

$L(x, y; \sigma)$ reprezentuje obrázok v škále σ, kde operácia * je konvolúciou obrázka f gausovým jadrom. $L(x, y; 0) = f(x, y)$

Obrázok 1.3. **Multi-scale reprezentácia obrázka**

(vľavo hore) Pôvodný čiernobiely obrázok - škála 0, (vpravo-hore) - (vpravo dole) postupná reprezentácia pre škálu 1, 8, 64

1.4 Detekcia čŕt

1.4.1 Detekcia čŕt všeobecne

Pojem črta (alebo tiež bod záujmu "point of interest") nie presne definovaný a je skôr používaný na označenie časti obrázku, resp. objektu v ňom, ktorá je istým spôsobom zaujímavou pre ďalší proces. Takýmito objektami môžu byť rohy, hrany, údolia, križovatky, škvrny alebo celé oblasti a pod. Významnými sú predovšetkým tým, že reprezentujú časť obrázka aj po rôznych zmenách obrázka . Podľa [Tuytelaars08] by dobrá črta mala spĺňať nasledujúce kritéria:

- Opakovateľnosť (repeatability) - pri dvoch obrázkoch zobrazujúcich ten istý objekt s rôznymi pozorovacími podmienkami, by na časti objektu zobrazenej v obidvoch obrázkoch malo byť detekované vysoké percento zhodných čŕt v obidvoch obrázkoch. Toto môže byť vnímané dvoma spôsobmi a to:
    - Invariancia (invariance) - kladie sa dôraz, aby pri väčších deformáciách, boli metódy neovplyvnené matematickými transformáciami zodpovedajúcimi príslušným deformáciám, pokiaľ je to možné.
    - Robustnosť (robustness) - pri relatívne malých deformáciách, ako napríklad šum, kompresia, rozmazanie a pod., je detektor na ne menej senzitívny, t.j. že požadovaná vlastnosť sa môže znížiť, ale nie priveľmi
- Odlíšiteľnosť (distinctiveness/informativeness) - okolie črty sa vyznačuje väčšou rôznosťou, na základe ktorej sa črty odlíšiť a neskôr párovať
- Lokálnosť (locality) - črty by mali byť lokálne, aby sa znížila pravdepodobnosť ich vzájomného pohltenia a tiež aby sa dali rozoznať
- Početnosť (quantity) - počet nájdených čŕt by mal byť dostatočne veľký, aby aj na malých objektoch bol detekovaný primeraný počet. Avšak optimálny počet závisí na potrebách aplikácie
- Presnosť (accuracy) - črty by mali byť presné lokalizované tak ako pozícii v obrázku takže s ohľadom na škálu
- Efektívnosť (efficiency) - detekovanie prihliada na časovo náročné aplikácie

1.4.2 Známe metódy

V literatúre je predostretých množstvo rôznych prístupov k tejto problematike, ktoré sa líšia ako princípom, tak aj typom čŕt, ktoré detekujú. Veľmi využívanými prístupmi sú tie, ktoré používajú derivácie šedotónového obrazu. Jedným z najznámejších je **Harrisov detektor** [Harris88], je rozšírením , využívajúci autokorelačnú maticu, alebo tiež maticu druhého momentu

$$M(\boldsymbol{x}) = g() * \begin{bmatrix} I_x^2(\boldsymbol{x}) & I_x(\boldsymbol{x}) I_y(\boldsymbol{x}) \\ I_x(\boldsymbol{x}) I_y(\boldsymbol{x}) & I_y^2(\boldsymbol{x}) \end{bmatrix} \qquad (1.2)$$

kde *g()* je Gaussové jadro a operácia * je konvolúcia jednotlivých zložiek matice. Matica reprezentuje rozloženie gradientu v okolí bodu $\boldsymbol{x}$ a jej vlastné hodnoty λ_1 a λ_2 zodpovedajú hlavným zmenám hodnôt - krivosti v dvoch kolmých smeroch. Ak je jedna z nich podstatne väčšia ako druhá, bod je na hrane, ak sú obidve veľké, bod zodpovedá rohu. Harris a Stephens však nepočítajú vlastné hodnoty, ale krivosť v bode odhadujú pomocou

$$R(\boldsymbol{x}) = \lambda_1 \lambda_2 - k(lamdba_1 + \lambda_2)^2 = det(M(\boldsymbol{x})) - k\, trace(M(\boldsymbol{x}))^2 \qquad (1.3)$$

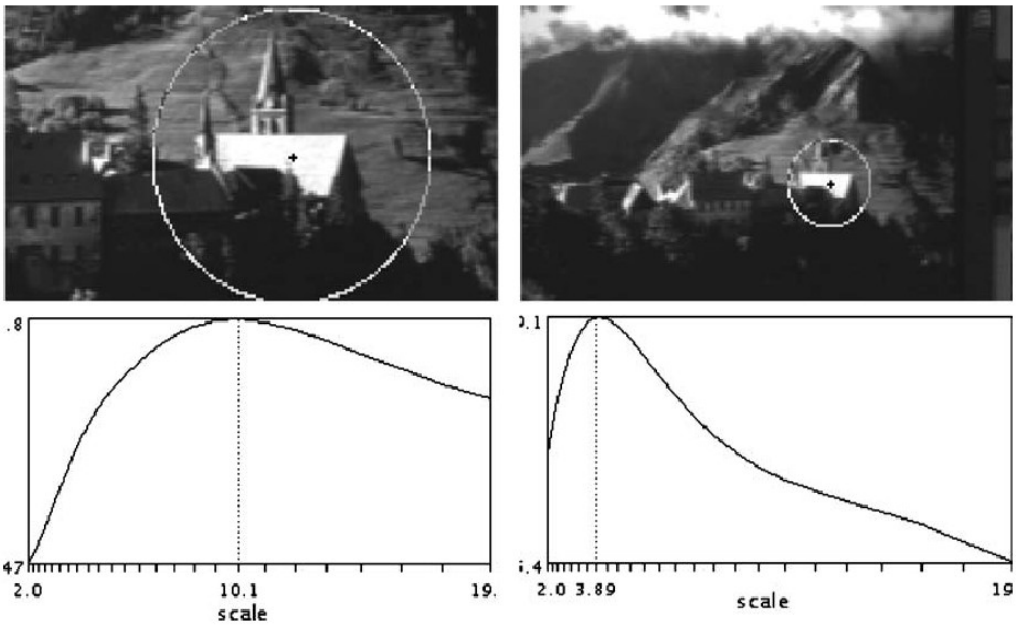
kde *k* je konštanta odhadnutá z pokusov a v literatúre sa stretávame s hodnotou medzi *0,04* a *0,15*. Črtami sú potom body, pre ktoré je R je väčší ako vopred stanovený prach a zároveň maximum v ich okolí. Harrisov detektor je síce robustný pre rotáciu a zmenu osvetlenia, ale taktiež je senzitívny na zmenu škály a podľa veľkosti použitého váhovacieho jadra aj na šum. Shi a Tomasi [Shi94] ukázali, že pri predpoklade afinných zmien v obraze je vhodnejšie použiť mierku R = min(λ_1 a λ_2). Nevýhody Harrisovho detektora, čo sa týka citlivosti na zmenu škály, boli potlačené **multi-scale Harris detektorom** [Baumberg00], ktorý vyhľadáva črty v niekoľkých škálach v reprezentácii škálového priestoru. Autokorelačná matica preto obsahuje ďalšie 2 parametre a to σ_D (derivačná škála) a σ_I (integračná škála), ktoré zvyčajne sú vo vzťahu $\sigma_I = k\sigma_D$, kedy *k* býva v literatúre z intervalu $\langle \sqrt{2}, 2 \rangle$ [Wiki CD]

$$M(\boldsymbol{x}, \sigma_D, \sigma_I) = g(\sigma_I) * \begin{bmatrix} L_x^2(\boldsymbol{x}, \sigma_D) & L_x(\boldsymbol{x}, \sigma_D) L_y(\boldsymbol{x}, \sigma_D) \\ L_x(\boldsymbol{x}, \sigma_D) L_y(\boldsymbol{x}, \sigma_D) & L_y^2(\boldsymbol{x}, \sigma_D) \end{bmatrix} \qquad (1.4)$$

kde L(**x**, σ) je reprezentácia *I* v škálovom priestore zodpovedajúca škále σ a L_a je parciálna derivácia *L* v smere *a*. Mikolajczyk [Mikolajczyk01] ďalej prichádza **Harris-Laplace** s rozšírením tejto metódy o automatickú detekciu charakteristickej škály [Lindeberg98] a vyberá body, ktoré sú zároveň maximom $M(x, \sigma_D, \sigma_I)$ a zároveň sú lokálnym extrémom škálovo normalizovaného Laplacovho operátora [Lindeberg98] medzi jednotlivými škálami

$$\nabla^2_{norm} L(\boldsymbol{x}, \sigma) = \sigma^2 \nabla^2 L(\boldsymbol{x}, \sigma) = \sigma^2 (L_{xx}(\boldsymbol{x}, \sigma) + L_{yy}(\boldsymbol{x}, \sigma)) \qquad (1.5)$$

V tomto zmysle sa detektor stáva škálovo invariantným.



Obrázok (1.4)**: Príklad výberu charakteristickej škály**

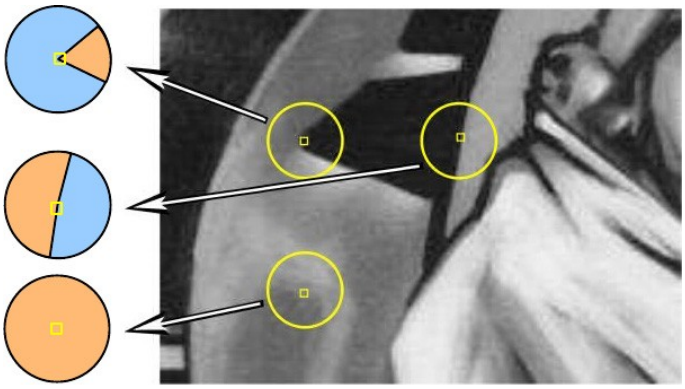
Vrchný rad ukazuje obrázka zosnímané s rozdielnou ohniskovou vzdialenosťou. Spodný rad ukazuje odozvu $\nabla^2_{norm} L(\boldsymbol{x}, \sigma)$ cez rozsah škál. Charakteristické škály 10,1 a 3,89 sú vybraté pre ľavý a pravý obrázok. Pomer škál zodpovedá zmene škály medzi obrázkami. Kružnica zobrazená okolo bodov v hornom rade je 3 násobkom charakteristickej škály. (prebraté z [Mikolajczyk04])

Mikolajczyk a Schmid [Mikolajczyk04] neskôr prispôsobili Harris-Laplace ako afinne invariantnú metódu **Harris-Affine** pomocou iteratívnej normalizácie eliptických afinných okolí bodov [Lindeberg98] získaných Harris-Laplace detektorom, ktoré sú odhadnuté pomocou veľkostí vlastných hodnôt autokorelačnej matice v danom bode. Postup detekcie je potom nasledovný:

1. detekcia prvotných bodov (regiónov) pomocou Harris-Laplace detektora
2. odhad afinnej formy pomocou autokorelačnej matice
3. normalizácia afinnej formy (elipsy) do kružnice
4. znovu detekcia novej pozície a škály v normalizovanom obrázku
5. pokračuj bodom 2 ak vlastné hodnoty autokorelačnej matice pre získaný bod nie sú rovnaké alebo skonči pokiaľ procedúra nezkonveruguje vo vopred stanovenom počte iterácii

Iným prístupom hranového detektora ako Harrisov je **SUSAN** (Smallest Univalue Segment Assimilating Nucleus) [Smith97]. Je založený na porovnávaní priamo jasových hodnôt bodov a s bodmi v malej oblasti okolo nich. Centrom oblasti je bod nazývaný *nucleus* a jeho hodnota je porovnávaná s bodmi v kruhovej oblasti spôsobom, či daný bod má alebo nemá podobnú hodnotu ako *nucleus*. Počet podobných bodov potom charakterizuje okolie bodu a nazýva sa USAN (Univalue Segment Assimilating Nucleus). Črty sú potom detekované spôsobom, že ak je počet "podobných" bodov v oblasti väčší ako stanovené minimum a menší ako preddefinovaný prah. Podľa veľkosti prahu sa dajú detekovať rohy alebo hrany, keďže počet podobných bodov v oblasti v bode ležiacom na hrane, klesá približne na polovicu a pri rohoch na štvrtinu [Tuytelaars08]. Oproti Harrisovmu poskytuje väčšiu robustnosť voči šumu, ale podobne ako on je zlyháva pri zmene škály.

Laganiere a Vincent [Laganiere02] kombinujú SUSAN a Harrisov detektor. Používajú klinový model, ktorý charakterizuje rohy podľa ich orientácie a uhlovej veľkosti. Touto kombináciou získavajú dobrý kompromis medzi presnosťou SUSAN a opakovateľnosťou Harrisovho detektora. Aj on však zlyháva pri zmenách škály.



Obrázok 1.5: Rohy v SUSAN sú detekované segmentáciou kruhového okolie "podobných" (oranžová časť) a "nepodobných" (modrá časť) regiónov. Rohy sú umiestnené tam, kde relatívna plocha "podobných" bodov (USAN) dosahuje lokálne minimum pod určitým prahom. (prebraté z [Tuytelaars08])

1.4.3 Škálovo invariantné metódy

Jedným z prístupov ako odstrániť tento nedostatok vo všeobecnosti, je detekovať črty v celej reprezentácii škálového priestoru (viď sekciu 1.3). Ako uvádzajú Tuytelaars a Mikolajczyk vo svojom prehľade [Tuytelaars08], mnohé navrhnuté metódy hľadajú maximum v 3D reprezentácii obrázku (x, y, škála), na základe myšlienky [Crowley84], kde škálový priestor bol vybudovaný s low-pass filtrami a črty boli detekované ako lokálne maximá v okolí zodpovedajúcemu 3D kocke a ich absolútne hodnoty boli vyššie ako stanovený prah. Tieto prístupy sa líšia hlavne v spôsobe výstavby reprezentácie škálového priestoru, teda použitím odlišných derivácií.

Lindeberg [Lindeberg] využíva škálovo normalizovaný operátor Laplace-of-Gaussian (LoG), ktorý je kruhovo symetrický a preto aj použitá metóda je prirodzene rotačne invariantná. Lowe vo svojej metóde **SIFT** [Lowe99],[Lowe04] (bližšie popísaná v sekcie 2.2) nahrádza LoG jeho aproximáciou pomocou rozdielu dvoch po sebe nasledujúcich obrázkov rozmazaných gaussovým filtrom - DoG (Difference-of-gaussians). Táto aproximácia poskytuje podobné výsledky ako LoG a je výpočtovo výhodnejšia. V [Lowe04] sú naviac detekované body ďalej interpolované pomocou prístupu [Brown02] pre získanie presnejšej pozície v obrázku ako aj v škálovom priestore. Ako už bolo spomenuté v predchádzajúcom bloku k týmto prístupom patrí aj

Multi-scale harris detektor [Baumeberg00] a **Harris-Laplace** [Mikolajczyk01]. Mikolajczyk a Schmid [Mikolajczyk04] navrhli ešte podobnú metódu ako Harris-Laplace a to **Hessian-Laplace**, ktorej princíp je rovnaký s tým rozdielom, že prvotné vyhľadávanie nerobia pomocou Harrisovho prístupu, ale použitím determinantu Hessianovej matice, ktorá detekuje škvrny, a na rozdiel od autokorelačnej matice pracuje s druhými deriváciami intenzity obrazu. Inou škálovo invariantnou metódou je **SURF** (Speeded Up Robust Features) navrhnutá Bayom a spol. v [Bay06], ktorá síce vyhľadáva črty v škálovom priestore, ale na budovanie nevyužíva priamo derivácie, ale ich aproximáciu pomocou blokových filtrov a integrálneho obrazu [Viola01] (bližšie popísaná v sekcie 2.4). Taktiež nerozmazáva postupne obrázky gaussovým filtrom, ale blokové filtre aplikuje na pôvodný obrázok. Jej cieľom je zefektívniť výpočet aj za cenu malej straty presnosti.

1.4.4 Afinne invariantné metódy

Škálovo invariantné metódy však zlyhávajú pri väčších zmenách obrázka v rozdielnych v kolmých smeroch, ako napríklad zmenšenie, skosenie, zošikmenie, pretože tie ovplyvňujú nielen pozíciu a škálu, ale aj tvar lokálnych štruktúr. Afinne invariantný detektor potom môžeme považovať za zovšeobecnenie škálovo invariantného [Tuytelaars08].

Medzi takéto metódy patria **Harris-Affine** a **Hessian-Affine** [Mikolajczyk04], ktoré sú rozšírením už spomenutých škálovo invariantných metód Harris-Laplace a Hessian-Laplace o iteratívny odhad afinných eliptických regiónov. Okrem prvotného výberu bodov v škálom priestore majú spoločný ostatný postup "rovnaký", ktorý bol spomenutý v sekcii 1.4.1. Tieto metódy sú navzájom doplnkové v zmysle, že vracajú iné typy čŕt v obrázku - Harris - rohy, Hessian - škvrny, aj keď Hessian detekuje aj rohové štruktúry v menších škálach.

Ako veľmi robustná afinne invariantná sa ukázala metóda **MSER** (Maximally Stable Extremal Regions) [Matas02]. Nedetekuje ani body ani škvrny ako také ale oblasti rôznych veľkostí a tvarov. V tejto metóde definovali regióny podľa extrémalnych vlastností intenzity v regióne a na jeho vonkajšej hranici. Myšlienka je nasledujúca. Nech

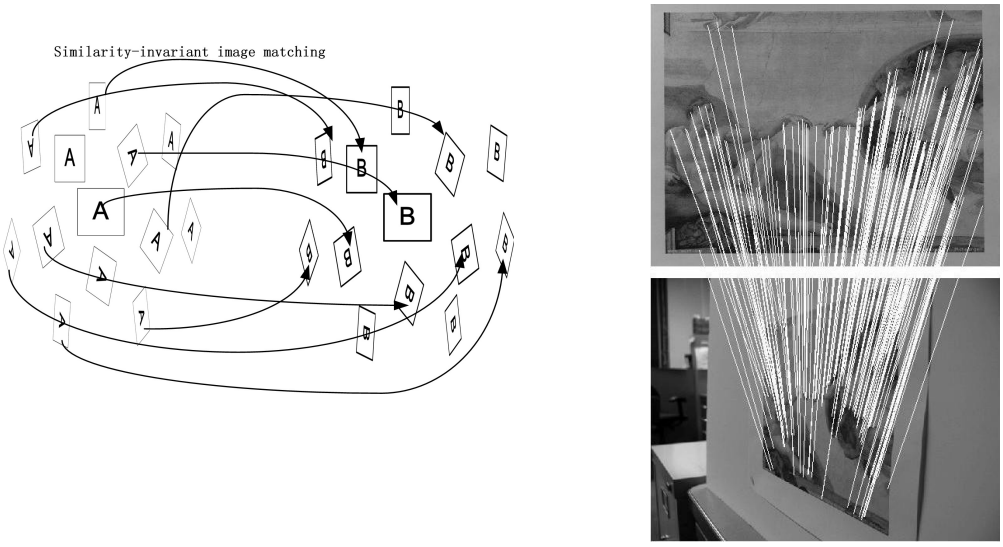
$S = \{I_0, I_1, .., I_{255}\}$, kde I_i je prahovaný obrázok prahom i, sú postupne všetky možnosti prahovania šedotónového obrázku I, ktorých body sú označené ako čierne, ak sú pod zvoleným prahom alebo ako biele, ak sú rovné alebo väčšie. Ak zobrazíme tieto obrázky ako film, potom prvý obrázok je biely a postupne sa na ňom objavujú čierne body, ktoré zodpovedajú lokálnemu minimu v intenzite a tie ďalej rastú. V istom momente sa oblasti zodpovedajúce lokálnemu miminu spoja. Nakoniec posledný obrázok je čierny. Množina všetkých spojených komponentov zo všetkých prahovaných obrázkov je množinou maximálnych regiónov (minimálne regióny sa dajú získať rovnakým postupom na invertovanom obrázku I). Vo veľa obrázkoch je lokálna binarizácia v istých regiónoch stabilnou počas prechodu viacerými obrázkami a tieto regióny sú centrom záujmu a podľa toho sa maximálne stabilné extremálne regióny. Tieto regióny majú nasledujúce vlastnosti:

- invarianciu voči afinných transformáciám a zmene intenzít
- konvarianciu voči transformácii zachovávajúcu susednosť $T : D \to D$ v obrazovej doméne
- stabilitu, nakoľko sú vybraté len regióny, ktorých príspevok je virtuálne nezmenený vo rozsahu prahov
- detekcia vo viacerých škálach. Aj keď nie je použité rozmazanie, detekované sú aj veľmi malé aj veľmi veľké štruktúry
- množina všetkých extremálnych regiónov je vypočítavané v zložitosti $O(n \log \log n)$, kde n je počet bodov obrázku I

Khan a spol. [Khan08] prišli s afinne invariantnou metódou, ktorá využíva morfologické operácie ako hranový detektor pre získanie segmentov, z ktorých sú potom získavané vstupné signály pre EMD rozklad [Huang98] a konkrétne prvú IMF. Body, v ktorých má IMF funkcia vyššiu frekvenciu sú potom kandidátmi na body záujmu (bližšie popísaná v sekcie 2.3).

Nový prístup **ASIFT** (Affine Sift) bol navrhnutý nedávno Morelom a Yuom [Yu09], ktorý nie je "detektorom" v pravom zmysle slova, ale prekladá spôsob ako porovnávať 2 obrázky z veľmi veľkými afinnými transformáciami a hľadať medzi nimi korešpondujúce body. Hlavnou myšlienkou je nie najprv detektovať body a potom ich afinne

normalizovať, ale práve naopak, z každého obrázku sa vytvori konečný malý počet otočených a naklonených obrázkov, ktorými simulujú inú pozíciu fotoaparátu, a následne v obrázkoch získaných z prvého a druhého detekujú body pomocou SIFT a tieto porovnávajú.



Obrázok 1.6: **ASIFT**

Vľavo: myšlienka metódy - simulácie zmeny pohľadov kamery na snímaný objekt v jednotlivých obrázok a ich následné porovnávanie

Vpravo: výsledné korešpondujúce body v obrázkoch zosnímaných z rôznych pozícii s veľkou zmenou pohľadu (prebraté z [Yu09])

2 Metódy na hľadanie čŕt použité v porovnávaní

2.1 Motivácia

V literatúre sa vyskytujú mnohé porovnania rôzne vybratých z vyššie spomenutých ako aj iných metód [Mikolajczyk05],[Schmid98],[Schmid00],[Yu09]. Takisto účel a cieľ porovnania je variabilný.

Pre testovanie sme sa rozhodli pre metódy SIFT [Lowe99],[Lowe04], SURF [Bay06], a metódu navrhnutú Khanom a spol. [Khan08], vzhľadom na odlišný prístup daných metód a tiež ich efektívnosť ako aj vhodnosť pre rôzne typy aplikácií.

2.2 SIFT

SIFT (Scale invariant feature transform) je metóda navrhnutá **Davidom G. Lowom** [Lowe99],[Lowe04]. Patrí dnes medzi najvyužívanejšie metódy na hľadanie čŕt v obraze. Využíva Difference-of-Gaussians (DoG) ako blízku aproximáciu normalizovaného Laplacian-of-Gaussian - $\sigma^2 \nabla^2 G$ [Lindeberg04].

Samotný SIFT má dve základné funkcie, a to nájdenie čŕt (detektor) a pre nájdené body vytvorenie príznaku (deskriptor) pre ďalšiu prácu pri párovaní.

Detekcia:

Detekovanie čŕt touto metódou by sa dalo rozdeliť na 4 fázy, a to:

1. Nájdenie extrémov v škálovom priestore
2. Verifikácia, či bod spĺňa vopred určené podmienky
3. Výpočet orientácie a dĺžky jej vektora
4. Priradenie deskriptora

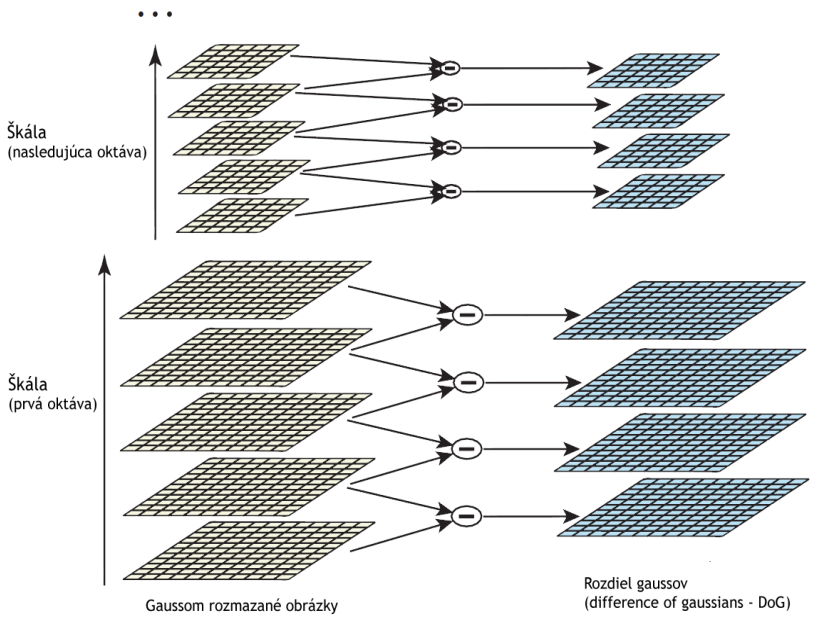
Tieto fázy sú potom v implementácii algoritmu previazané, kvôli efektivite.

2.2.1 Hľadanie extrémov v škálovom priestor

Ako už bolo spomenuté, Lowe pri vyhľadávaní extrémov využíva rozdiel D dvoch obrázkov rozmazaných gaussom zodpovedajúcim škálam σ a kσ.

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))*I(x,y)$$
$$=L(x,y,k\sigma)-L(x,y,\sigma) \qquad (2.2.1)$$

V prvom kroku, predpokladá, že vstupný obrázok má rozmazanie $\sigma = 0.5$, preto mu zdvojnásobuje veľkosť bilineárnou interpoláciou, čiže nový obrázok ma rozmazanie $\sigma = 1$ relatívne oproti pôvodnému. Následne buduje jednotlivé levely škálového priestoru postupným rozmazávaním, teda konvolúciou gaussovým jadrom, odlíšenými o konštantný násobok *k*. Škálový priestor rozdeľuje na oktávy. Tvorba oktávy pozostáva z vopred stanoveného počtu vrstiev *s*, ktoré tvoria rozmazané obrázky. Keďže každá nasledujúca oktáva má dvojnásobné σ oproti predchádzajúcej a podľa [Lowe04] je optimálny počet levelov pre jednu oktávu rovný $s = 3$, $k = 2^{1/s}$. Susedné vrstvy potom od seba odpočíta pre získanie $D(x, y; \sigma)$. Kompletná oktáva však zahŕňa $s+3$ rozmazaných obrázkov nutných pre vytvorenie $s+2$ DoG obrázkov potrebných pre hľadanie extrémov. Po vytvorení oktávy zoberie vrstvu prislúchajúcu k dvojnásobnej vstupnej škále σ, prevzorkuje ho spôsobom že, vynechá všetky nepárne stĺpce a riadky a následne ho používa na budovanie ďalšej oktávy, kedy presnosť vzorkovania relatívna voči σ sa neodlišuje od začiatku predošlej oktávy, ale sa tým omnoho zrýchli výpočet [Lowe04].



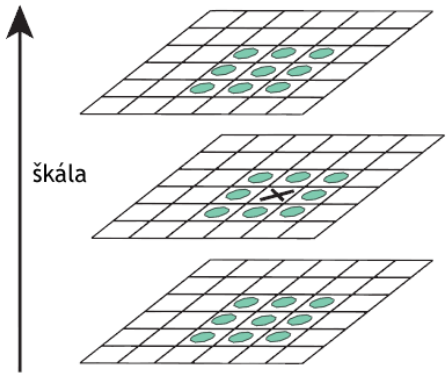
Obrázok 2.2.1: **Tvorba škálového priestoru pre *s* = 2**

Ľavý stĺpec : postupné rozmazávanie konvolúciou gaussovým jadrom - $L(x,y;k\sigma)$

Pravý stĺpec: Rozdiel dvoch susedných vrstiev - $D(x,y;\sigma_i) = L(x,y;k\sigma_i) - L(x,y;\sigma_i)$

(prevzaté z [Lowe04])

Následne po vybudovaní oktáv hľadá extrémy pre $D(x, y; \sigma)$ tým spôsobom, že pre každý bod DoG obrázka zistí, či je extrémom v jeho 3x3 okolí (8 bodov). Ak áno, zisťuje či je rovnakým extrémom aj v 3x3 okolí (9 bodov) v DoG o úroveň hore aj dole, t.j. porovnanie s bodu s 26 susednými bodmi v susedných úrovniach (viď obrázok 2.2.2). Nájdené body sa stávajú kandidátmi, ktoré sú ďalej filtrované.



Obrázok 2.2.2: **Hľadanie extrému** (prebraté z [Lowe04])

Kandidátom na bod záujmu sa stáva bod, ktorý je extrémom spomedzi 26 susedných bodov a to 8 vo svojom okolí 3x3 v rovnakej škále ako aj 9 v najbližšej škále nad a pod.

2.2.2 Verifikácia kandidátov

2.2.2.1 Určenie presnej polohy

[Lowe99] bral do úvahy súradnice kandidátov na akých boli nájdení, avšak [Lowe04] na základe poznatkov [Brown02] určuje presnejšiu pozíciu nájdeného extrému interpoláciou. Využíva Taylorov rozvoj funkcie $D(x, y; \sigma)$.

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\frac{\partial^2 D^T}{\partial \boldsymbol{x}^2}\boldsymbol{x} \qquad (2.2.2)$$

kde D a jej derivácie sú vypočítavané v bode kandidáta pomocou rozdielu susedných bodov [Brown02] a $\boldsymbol{x}=(x, y, \sigma)^T$ je odchýlka od neho. Pozícia extrému $\hat{\boldsymbol{x}}$ je potom následne vyrátaná z rovnice (2.2.2) položenej rovnej 0.

$$\hat{\boldsymbol{x}}=-\frac{\partial^2 D^{-1}}{\partial \boldsymbol{x}^2}\frac{\partial D}{\partial \boldsymbol{x}} \tag{2.2.3}$$

Ak $\hat{\boldsymbol{x}}$ je väčšie ako 0,5 alebo menšie ako -0,5 v jednom z rozmerov, znamená to, že extrém leží bližšie k inému bodu a preto interpolácia sa opakuje pre nový bod, ku ktorému je extrém bližšie [Lowe04]. Získaná odchýlka je potom pripočítaná k pozícii interpolovanému bodu.

2.2.2.2 Odmietnutie bodov s nízkym kontrastom

[Lowe04] ďalej kontroluje hodnotu $|D(\hat{\boldsymbol{x}})|> 0{,}03$, ktorú získal z rovnice

$$D(\hat{\boldsymbol{x}})=D+\frac{1}{2}\frac{\partial D^T}{\partial \boldsymbol{x}}\hat{\boldsymbol{x}} \tag{2.2.4}$$

kde pracuje s hodnotami bodov obrázkov v intervale <0,1>. Ak je hodnota menšia, bod odmietne.

2.2.2.3 Odmietnutie bodov s nízkym zakrivením

Následne prebieha kontrola, či body nemajú veľký pomer medzi krivosťou v smere hrany, na ktorej sú detekované a v kolmom smere. Takéto body sú nestabilné, pretože je ich ťažké identifikovať. Hlavné zakrivenia sú pomerné k vlastným hodnotám Hessianovej matice

$$\boldsymbol{H}=\begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.2.5}$$

, ktoré ale netreba priamo vypočítať, keďže Lowa zaujíma len ich vzájomný pomer. Na výpočet jednotlivých derivácii používa rozdiel susedných bodov. Nech α a β sú vlastné hodnoty **H**, pričom α nech je väčšia. Potom pomer medzi nimi je $r=\frac{\alpha}{\beta}$ a použitím prístupu [Harris88]

$$Trace(\boldsymbol{H})=D_{xx}+D_{yy}=\alpha+\beta \tag{2.2.6}$$

$$Det(\boldsymbol{H}) = D_{xx} D_{yy} - D_{xy}^2 = \alpha \beta \tag{2.2.7}$$

získava

$$\frac{(Trace(\boldsymbol{H}))^2}{Det(\boldsymbol{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{2.2.8}$$

ktorú porovnáva s vopred určeným prahom, teda odmieta body, ktoré nevyhovujú nerovnici

$$\frac{Trace(\boldsymbol{H})^2}{Det(\boldsymbol{H})} < \frac{(r_{thr}+1)^2}{r_{thr}} \tag{2.2.9}$$

kde r_{the} je prah a podľa [Lowe04] vhodná konštanta je $r_{thr} = 10$.

2.2.3 Výpočet orientácie a dĺžky gradientu

Bodom, ktoré vyhovovali v predchádzajúcich testoch sa priradí veľkosť m a smer θ orientácie, ktoré sú získané z Gaussom rozmazaného obrázku so škálou najbližšie k škále detekovaného bodu a to nasledujúcim spôsobom. Pre všetky body $L(x,y;\sigma)$ a všetky škály sa predpočítajú hodnoty m a θ.

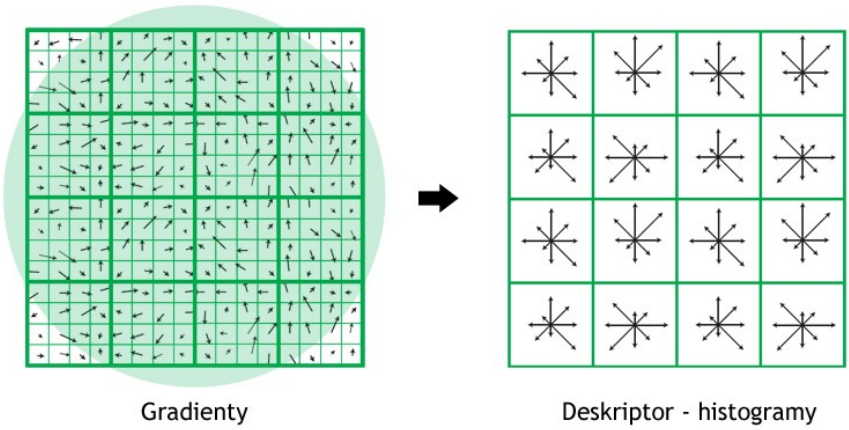
$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{2.2.10}$$

$$\theta(x, y) = \arctan\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \tag{2.2.11}$$

Pre každú nájdenú črtu je vytvorený histogram orientácii, ktorý ma 36 pozícii, každý zodpovedajúci rozsahu 10°. Do histogramu prispieva každý bod v okolí črty svojou veľkosťou orientácie, pričom tá je váhovaná Gaussovým oknom so σ, ktorá je 1,5 násobkom škály nájdeného bodu. Črte je priradená tá orientácia, ktorá ma v histograme maximum. Pokiaľ sa v histograme nachádzajú pozície, ktorých hodnota je nad 80% maxima, pre každú z týchto hodnôt je vytvorená kópia bodu, ktorému sa priradí orientácia na danej pozícii. Kvôli lepšej presnosti je priraďovaná orientácia a veľkosť získaná interpoláciou paraboly prechádzajúcej vrcholom a najbližšími hodnotami. [Lowe04]

2.2.4 Výpočet príznaku - deskriptora

Príznak - 128 rozmerný vektor - je vypočítavaný zo 16x16 okolia črty ako 16 histogramov (4x4) orientácii s 8 pozíciami. Do jedného histogramu prispievajú body z podokna o veľkosti 4x4, ktoré sú predtým váhované Gaussovým oknom so σ = polovici veľkosti okna (viď obrázok 2.2.3). Vektor je následne normalizovaný na jednotkovú dĺžku, ďalej hodnoty väčšie ako 0,2 sú nahradené hodnotou 0,2, aby sa zabránilo vplyvu gradientov s veľkou dĺžkou [Lowe04] a opäť je vektor normalizovaný na jednotkovú dĺžku.



Obrázok 2.2.3: **Tvorba deskriptora** (upravený z Lowe04)
Vľavo sú znázornené gradienty v obrázku, rozdelenie na 16 podokien ako aj vplyv gausovho jadra. Vpravo sú získané histogramy pre jednotlivé podokná.

2.3 Detekcia čŕt využitím rozkladu empirickým spôsobom (Empirical Mode Decomposition - EMD)

J. F. Khan spolu s partnermi [Khan08] prináša, ako uvádza, široko afinne invariantnú metódu na hľadanie čŕt. Neprehľadáva ako Lowe škálový priestor, ale body sa snaží vyhľadávať na hranách, ktoré získal morfologickými transformáciami [Serra82] podľa vzoru [Chin93], kde na lokalizáciu alebo rozhodnutie, či bod je zaujímavý využíva rozklad EMD [Huang98], konkrétne jej 1 vnútornú funkciou (intrisinc mode function - IMF).

Priebeh funkcie sa dá rozdeliť na 3 fázy

1. Detekovanie hrán

2. EMD rozklad segmentov na 1. IMF funkciu

3. Výber kandidáta

2.3.1 Detekovanie hrán

Najprv obrázok jemne rozmaže, použitím transformácií otvorenie-uzavretie a uzavretie-otvorenie, tak že výsledný obrázok je priemerom hodnôt týchto transformácií.

$$I_b = \frac{{}_B({}_BI_O)_C + {}_B({}_BI_C)_O}{2} \tag{2.3.1}$$

kde B je 3x3 8-spojitý štrukturálny element a ${}_BX_O$ a ${}_BX_C$ *sú* transformácie otvorenia a uzavretia pomocou B na obrázku X. Ďalej je použitý operátor gradientu [Hasan00] ako rozdiel dilatácie a erózie.

$$I_{gr} = {}_B(I_b)_d - {}_B(I_b)_e \tag{2.3.2}$$

Obrázok I_{gr} je následne prahovaný

$$I_e = \begin{cases} 1 & I_{gr} > y \\ 0 & inak \end{cases} \tag{2.3.3}$$

$$y = \frac{\sum (I_{gr} . I_c)}{\sum I_c} \tag{2.3.4}$$

kde I_c je $max(g_1 ** I_{gr}\ ,\ g_2\ **\ I_{gr})$, operácia $g**X$ konvolúcia obrázka X jadrom g, $g_1 = [-1\ 0\ 1]$, $g_2 = g_1^T$ a operácia . je násobenie medzi totožnými bodmi. Binárnu mapu I_e ďalej upravuje tak, že ztenšuje hrany na šírku 1 pixlu eróziou vo vertikálnom a horizontálnom smere. Výsledným obrázkom je maximum z týchto obrázkov, teda

$I_{te} = max(I_{ve}, I_{he})$, kde

$$ak\ I_e \neq 0 \quad potom \begin{cases} I_{he}(x_i, y_j) = I_e(x_i, y_j) \\ I_{he}(x_i, y_k) = 0 \qquad k \in_{k \neq j} \{j - w_h, j + w_h\} \end{cases} \tag{2.3.5}$$

$$ak\ I_e \neq 0 \quad potom \begin{cases} I_{ve}(x_i, y_j) = I_e(x_i, y_j) \\ I_{ve}(x_k, y_j) = 0 \qquad k \in_{k \neq i} \{i - w_h, i + w_h\} \end{cases} \tag{2.3.6}$$

Takto získaný obrázok ešte môže obsahovať veľmi krátke hranice, preto tie časti ktoré majú menšiu dĺžku ako N, sú odstránené.

2.3.2 EMD rozklad a prvá IMF

Výber kandidátov prebieha na jednotlivých hraniciach, resp. ich častí. Nech P reprezentuje jeden taký segment $P = \{p_i = (x_i, y_i);\ i = 1, 2, 3 \cdots n\}$, p_{i+1} je susedný s p_i. Hranice sú transformované na 1 rozmerný signál - *1D θ-P*, ktorý tvoria zmeny uhlov pri prechode hranicou, ktorý je následne vstupom pre EMD rozklad.

2.3.2.1 Vstupný signál pre EMD rozklad

Jednotlivé hodnoty uhlov $\theta(p_i)$ sú vypočítané pomocou kovariančnej matice $M(p_i)$ a jej vlastných hodnôt

$$M(p_i) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \tag{2.3.7}$$

$$m_{11} = \left[\frac{1}{2s+1} \sum_{j=i-s}^{i+s} x_j^2 \right] - \bar{x}_i^2 \tag{2.3.8}$$

$$m_{22} = \left[\frac{1}{2s+1} \sum_{j=i-s}^{i+s} y_j^2 \right] - \bar{y}_i^2 \tag{2.3.9}$$

$$m_{12} = m_{21} = \left[\frac{1}{2s+1} \sum_{j=i-s}^{i+s} x_j y_j \right] - \bar{x}_i \bar{y}_i \tag{2.3.10}$$

$$\bar{x}_i = \frac{1}{2s+1} \sum_{j=i-s}^{i+s} x_j \tag{2.3.11}$$

$$\bar{y}_i = \frac{1}{2s+1} \sum_{j=i-s}^{i+s} y_j \tag{2.3.12}$$

$$\theta(p_i) = \begin{cases} \left| \arctan\left(\frac{\lambda_1 - m_{11}}{m_{12}} \right) \right| & ak\ m_{12} \neq 0 \\ \frac{\pi}{2} & inak \end{cases} \tag{2.3.13}$$

kde λ_1 je vlastná hodnota $M(p_i)$ a s je nejaké celé číslo [Khan08], ktoré definuje výsek z P okolo bodu p_i. Absolútna hodnota je použitá preto, aby sa zabránilo príliš veľkému kolísaniu hodnôt pre susedné body spôsobené kvantovými efektami. [Khan08] podľa [Quddus99], [Lee95].

2.3.2.2 EMD rozklad [Huang98]

Výsledkom EMD rozkladu sú postupné IMF signály. Podľa [Huang98] IMF je funkcia, ktorá spĺňa 2 podmienky: (1) na celom obore sa počet extrémov líši od počtu prechodov cez 0 maximálne o 1 a (2) v každom bode je stredová hodnota medzi obalom definovaného maximami a obalom definovaného minimami rovná 0. Autorov [Khan08] však zaujíma len 1. z rozkladu. Nech $c_i(t)$ je vstupný signál potom $imf_i(t)$ sa získava iteráciou

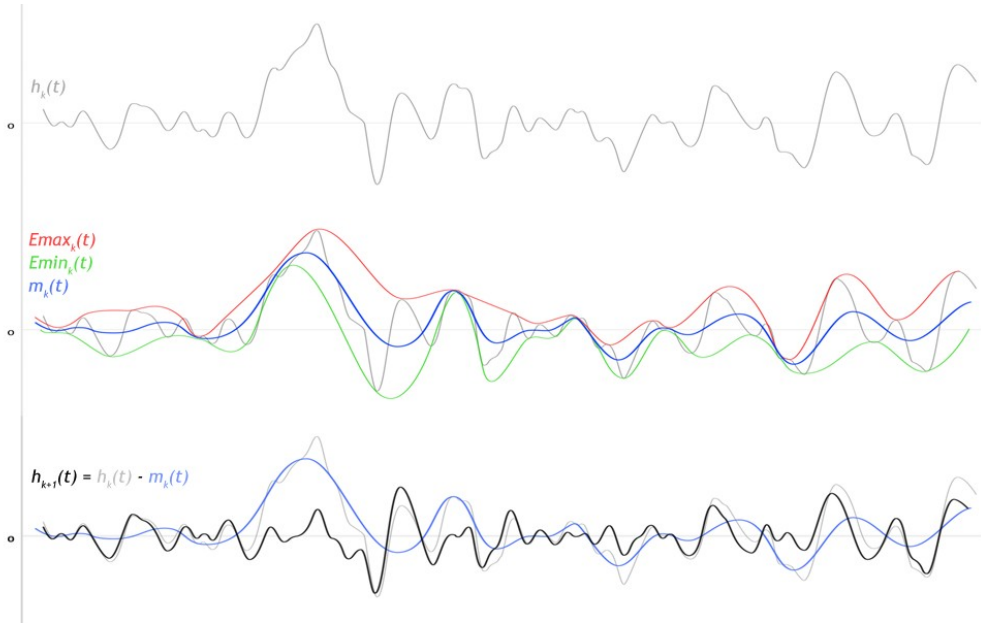
$$m_k(t) = \frac{Emax_k(t) + Emin_k(t)}{2} \tag{2.3.14}$$

$$h_{k+1}(t) = h_k(t) - m_k(t) \tag{2.3.15}$$

kde $h_0(t) = c_i(t)$, $Emax_k(t)$ je obalom definovaným kubickou interpoláciou maxím $h_k(t)$ a $Emin_k(t)$ je obalom definovaným kubickou interpoláciou miním $h_k(t)$. Iterácie pokračujú, kým štandardná odchýlka SD

$$SD = \sum_{t=0}^{T} \left[\frac{(h_k(t) - h_{k-1}(t))^2}{h_{k-1}^2(t)} \right] \tag{2.3.16}$$

nie pod určitou hranicou. Typickou hodnotou je 0,2 - 0,3 [Huang98]. Konečná $h_k(t)$ je potom $imf_i(t) = h_{max}(t)$. Ďalšia $imf_{i+1}(t)$ sa získa obdobne, ale vstupným signálom je $c_{i+1}(t) = c_i(t) - imf_i(t)$.



Obrázok 2.3.1: **Jeden krok iterácie pre výpočet $imf_i(t)$.**

Hore: Vstupný signál $h_k(t)$

Stred: Obaly extrémov vytvorené kubickou interpoláciou: $Emax_k(t)$ - maxím $h_k(t)$ (červená), $Emin_k(t)$ - miním $h_k(t)$ (zelená) a ich medián $m_k(t)$

Dole: Vstupný signál $h_k(t)$ (šedá), medián $m_k(t)$ (modrá) a výsledný signál kroku iterácie $h_{k+1}(t) = h_k(t) - m_k(t)$ (čierna) (farebne upravený obrázok z [Huang98])

2.3.3 Výber kandidátov

Selekcia bodov záujmu prebieha na získanej IMF funkcii z *1D θ-P* signálu prislúchajúceho segmentu $P = \{pi = (xi, yi); i=1,2,...,n\}$, $IMF(p_i)$ a $\theta(p_i)$ sú jednotlivými hodnotami pre dané body segmentu. Prvotný výber kandidátov je na základe frekvencie IMF funkcie: Prvá IMF ukazuje odlišne vyššiu frekvenciu na pravých bodoch záujmu ako na rovných čiarach [Khan08]. Vybraté sú body, ktoré majú vo svojom okolí (Wz) aspoň toľko prechodov cez 0 ako je prah, ktorý je v [Khan08] stanovený na 1/3 všetkých prechodov cez 0 danej IMF. Zo získaných bodov sú ďalej body, ktoré majú vo svojom okolí (Ws) body s rovnakým počtom prechodov cez 0, ale s vyššou hustotou, odstránené zo zoznamu kandidátov. Zvyšní kandidáti sú testovaní, či sú extrémami v pôvodnom obrázku vo svojom 3x3 okolí. Nakoniec sú odmietnuté také body, aby vzdialenosť medzi jednotlivými kandidátmi bola aspoň taká ako vopred stanovená minimálna vzdialenosť, ktorá je v [Khan08] 5 pixlov.

2.4 SURF: Speeded Up Robust Features [Bay06],[Bay08]

Bay, Tuytelaars, Van Gool prišli v roku 2006 s škálovo a rotačne invariantnou metódou, ktorá je založená na princípe Hessianovej matice, ale na výpočet derivácií používa jednoduchú aproximáciu založenú na integrálnom obraze [Viola01], čo znižuje čas výpočtu [Bay06], [Bay08]. Súčasťou SURF je aj vlastný deskriptor, ktorý popisuje distribúciu Haar-vlnkových odoziev v okolí bodu záujmu, ktorý ale nie je popísaný v tejto práci, a pre bližší popis odkazujeme čitateľa na [Bay06], [Bay08], [Evans09].

2.4.1 Integrálne obrazy

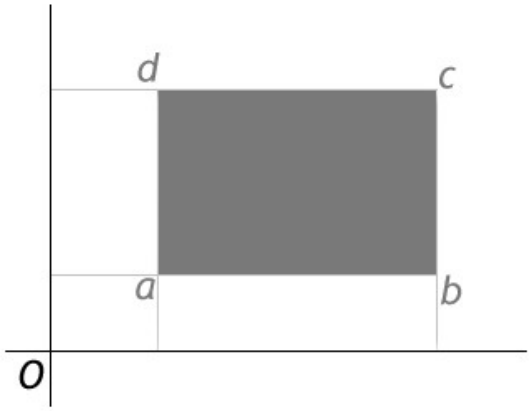
Integrálny obraz I_Σ obrazu I , ako je popísaný v [Viola01], je obraz, v ktorom každý bod obsahuje súčet obrazových bodov v obdĺžnikovej oblasti v I definovanej daným bodom a počiatkom.

$$I_{\sum}(x, y)=\sum_{i=0}^{x}\sum_{j=0}^{y} I(i, j) \qquad (2.4.1)$$

Dotaz na súčet bodov v obdĺžnikovej oblasti má konštantnú zložitosť bez ohľadu na jej veľkosť a je daný predpisom

$$\sum rect_{abcd}=I_{\sum}(\boldsymbol{c})+I_{\sum}(\boldsymbol{a})-(I_{\sum}(\boldsymbol{b})+I_{\sum}(\boldsymbol{d})) \qquad (2.4.2)$$

kde **a**, **b**, **c**, **d** sú rohové body oblasti a **a** je najbližšie k počiatku (viď obr. 2.4.1)



Obrázok 2.4.1: **Obdĺžniková oblasť záujmu**

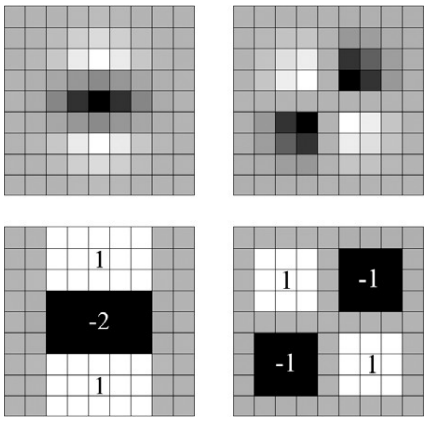
Pri použití integrálneho obrazu I_Σ na výpočet súčtu bodov v obdĺžnikovej oblasti danej bodmi *a, b, c, d* v *I* stačia štyri operácie $\Sigma = I_\Sigma(c) - I_\Sigma(b) - (d) + I_\Sigma(a)$.

2.4.2 Rýchly Hessian (Fast-Hessian)

Detektor metódy SURF je založený na determinante Hessianovej matice v škálovom priestore, kedy ak $\boldsymbol{x} = (x,y)$ je bod *I*, tak Hessianova matica $\boldsymbol{H}(\boldsymbol{x}, \sigma)$ pre bod **x** v škále σ je definovaná

$$\boldsymbol{H}(\boldsymbol{x}, \sigma)=\begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix} \qquad (2.4.3)$$

kde $L_{xx}(\boldsymbol{x}, \sigma)$, $L_{xy}(\boldsymbol{x}, \sigma)$, $L_{yy}(\boldsymbol{x}, \sigma)$ sú konvolúcie príslušnými druhými deriváciami Gaussového $g(\sigma)$ jadra obrázku I v bode $\boldsymbol{x}$, známych ako Laplacian-of-Gaussian. Konvolúcie spomenutými jadrami však Bay a spol. [Bay06] nahradili blokovými filtrami (box filters) Dxx, Dyy, Dxy (viď obrázok 2.4.2) ako aproximácie druhých derivácii Gaussového jadra. Pri použití integrálneho obrazu potom trvá aplikovanie blokových filtrov lineárny čas vzhľadom na počet obrazových bodov nezávisle od veľkosti filtra, kdežto pri Laplacian-of-Gaussian je zložitosť (počet bodov I)x(počet bodov jadra).



Obrázok 2.4.2: **Aproximácie druhých derivácií Gaussového jadra 9x9 blokovými filtrami**

Horný rad: Diskrétne parciálne derivácie Gaussového jadra v yy-smere a xy-smere

Dolný rad: Príslušné aproximácie s váhami pre jednotlivé bloky (šedé body sú rovné 0)

(prebraté z [Bay06])

Aproximácie Dxx, Dyy, Dxy sú potom vypočítané tak, že zodpovedajúce obdĺžnikové oblasti z I_Σ (biele a čierne) sú násobené príslušnou váhou (viď obrázok 2.4.2). Následne determinant je daný predpisom

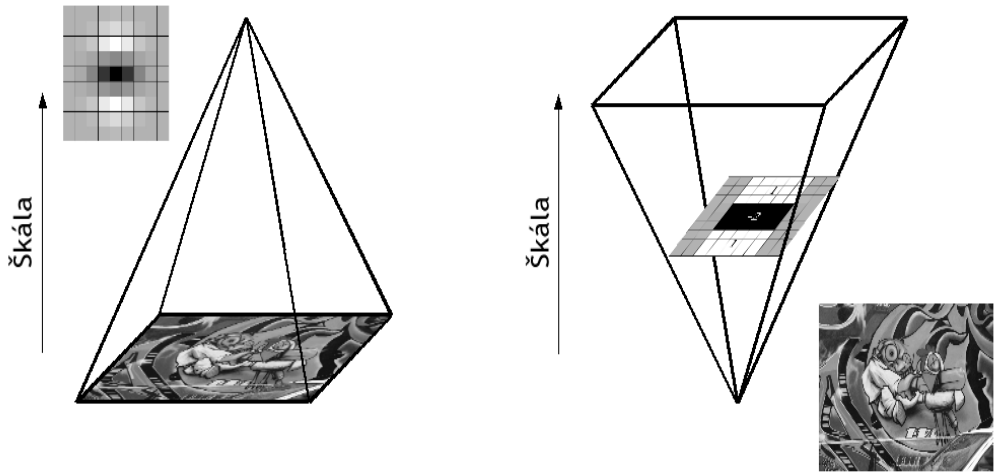
$$\det(\boldsymbol{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \qquad (2.4.4)$$

kvôli vyváženiu relatívnych váh filtra a Hessianovho determinantu [Bay06].

2.4.3 Budovanie škálového priestoru

Na rozdiel od klasického prístupu budovania škálového priestoru reprezentovaného pyramídou s vrstvami získanými postupným rozmazávaním Gaussom a následným

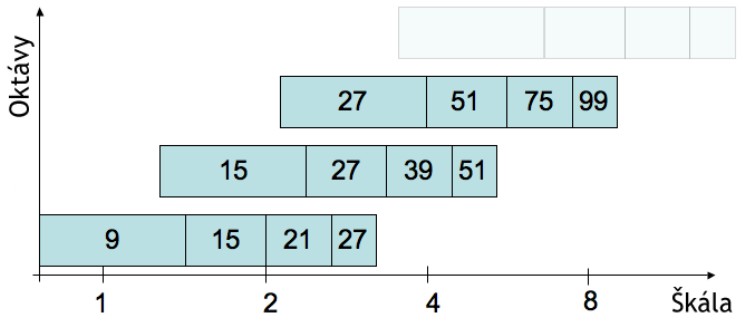
podvzorkovaním pre vytvorenie ďalšieho levelu, [Bay06] buduje škálový priestor spôsobom, že zväčšuje blokový filter a aplikuje ho na pôvodný obrázok.



Obrázok 2.4.3: **Rozdiel v spôsobe budovania škálového priestoru**

Kým štandardne sa škálový priestor buduje zmenšovaním obrázka (vľavo) Bay a spol. ho budujú použitím integrálneho obrazu a zväčšovaním blokového filtra (vpravo) (prevzaté z [Bay08])

Inicializačným filtrom je filter s veľkosťou 9x9 (viď obrázok 2.4.2), zodpovedajúci škále $\sigma = 1.2$. Následne aplikuje filtre o veľkosti 15x15, 21x21, 27x27 a pod. Podobne ako SIFT [Lowe99],[Lowe04] rozdeľuje škálový priestor na oktávy, kde prechod medzi oktávami je 2σ a kde rozdiel medzi veľkosťami filtrov sa s narastajúcou oktávou zdvojnásobuje. (Obrázok 2.4.4). Škála jednotlivých filtrov je vypočítavaná ako *(veľkosť filtra)x(1,2/9)*.



Obrázok 2.4.4: **Veľkosti blokových filtrov**

Grafická reprezentácia veľkosti filtrov pre 3 rôzne oktávy. Logaritmická vodorovná os

reprezentuje škály. Oktávy sa prekrývajú aby sa zabezpečilo pokrytie všetkých možných škál bez švov. (prebraté z [Bay08])

2.4.4 Vyhľadávanie bodov

Obdobne ako [Lowe99], [Lowe04] kandidátom sa stáva bod, ktorý je maximom (u v SIFT je to extrém) vo svojom 3x3 okolí a rovnako aj v zodpovedajúcom 3x3 okolí vo vrstve nad a pod (viď obrázok 2.2.2). Následne pre získanie presnejšej polohy a škály získaných kandidátov Bay a spol. prevádza interpoláciu obdobne ako Lowe v SIFT [Lowe99], [Lowe04] na základe metódy navrhnutej v [Brown02] s tým, že na rozvoj taylorovho rozvoja je použitý determinant Hessianovej matice. Čiže ak $\boldsymbol{x} = (x, y, \sigma)$ a $H(\boldsymbol{x}) = det(\boldsymbol{H}(\boldsymbol{x}))$ potom

$$H(\boldsymbol{x}) = H + \frac{\partial H^T}{\partial \boldsymbol{x}} \boldsymbol{x} + \frac{1}{2} \boldsymbol{x}^T \frac{\partial^2 H^T}{\partial \boldsymbol{x}^2} \boldsymbol{x} \qquad (2.4.5)$$

a odchýlka $\hat{\boldsymbol{x}} = (x, y, \sigma)$ je získaná z lineárneho systému

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 H^{-1}}{\partial \boldsymbol{x}^2} \frac{\partial H}{\partial \boldsymbol{x}} \qquad (2.4.6)$$

Podobne ako v kapitole 2.2.2.1, ak odchýlka v niektorom z rozmerov je väčšia ako 0,5 interpolácia pokračuje pre iný bod, ku ktorému je odchýlka bližšie. Body ktoré nekonvergujú počas vopred stanoveného počtu krokov sú odmietnuté.

3 Implementácia

3.1 Technológie

Aplikáciu budeme robiť v C# a technológii .NET. Na použitie jednotlivých základných grafických operácii s obrázkami sme sa rozhodli pre OpenCV [OpenCV] knižnicu od spoločnosti Intel na základe porovnania [Kulkova07], resp. pretože OpenCV je pre C++, tak pre jej wrapper do C# .NET technológie, a to konkrétne EmguCV [EmguCV], ktorá je distribuovaná pod GPL licenciou a medzi ostatnými wrapermi ako jediná implementovala OpenCV verziu 2.

OpenCV je veľmi využívaná voľne dostupná knižnica pre akademické aj komerčne využitie určená pre oblasť real-time počítačového videnia a obsahujúca viac než 500 optimalizovaných algoritmov.

Počas vývoja sme však zistili nedostatočné spracovanie OpenCV v EmguCV, konkrétne chýbajúce metódy na výpočet vlastných hodnôt a riešenie lineárneho systému, preto v niektorých častiach sme použili implementáciu matíc a práce s nimi [DotNetMatrix], ktorá je distribuovaná ako voľná.

3.2 Štruktúra aplikácie

Pre neskoršie rozšírenie aplikácie o ďalšie metódy sme sa rozhodli oddeliť aplikačnú časť s GUI, ktorá bude mať na starosti porovnávanie a zobrazovanie výsledkov od implementácie samotných metód na detekciu čŕt. Metódy sú importované z dll knižníc pri spustení aplikácie uložených v špecifickom adresári v **features_dll**, ktorý je umiestnený v rovnakom adresári ako exe súbor aplikácie. Pri neskoršej potrebe porovnať inú metódu na detekciu čŕt potom stačí len implementovať samotnú metódu podľa nižšie uvedených kritérií a výslednú dll knižnicu s novou metódou len skopírovať do určeného adresára bez nutnosti zásahu do aplikácie a jej rekompilácie.

3.2.1 Trieda CDetector a namespace FeaturesDetectors

Každá metóda je implementovaná súčasť namespacu FeaturesDetectors, ktorý obsahuje základné triedy, ktoré sú spoločné pre všetky metódy alebo od ktorých jednotlivé triedy metód potom dedia svoju hlavnú triedu, alebo triedy, ktoré využivajú vo svojich funkciách alebo štruktúrach.

Namespace FeaturesDetectors zahrňuje nasledujúce triedy:

trieda **CDetector** - hlavná trieda pre detekciu, od ktorej implementované metódy dedia svoju hlavnú triedu. Obsahuje spoločné virtuálne vlastnosti a funkcie, ktoré každá metóda musí implementovať a to:

vlastnosť **detectorName** - slúži ako konštanta na identifikáciu metódy

```
public virtual string detectorName { get { return ""; } }
```

vlastnosť **options** - obsahuje špecifické nastavenie pre metódu. Je do nej priradená podtrieda triedy CDetectorOptions vlastná pre konkrétnu metódu

```
public virtual CDetectorOptions options { get; protected set; }
```

funkciu **ShowOptionsDialog** - ktorá slúži na obsluhu samostatného okna s špecifickými nastaveniami pre konkrétnu metódu, ktoré sa ukladajú do **options**. Túto obsluhu si riadi každá metóda zvlásť ako aj implementáciu dialógového okna.

```
virtual public DialogResult ShowOptionsDialog() ;
```

funkciu **GetFeatures** - ktorá je hlavnou detekovacou funkciou a vracia zoznam nájdených čŕt

```
virtual public Feature[] GetFeatures(Image<Gray, byte> image);
```

a spoločnú procedúru **DrawFeatures** - ktorá zabezpečuje vykresľovanie čŕt do obrázka

```
public void DrawFeatures(Image<Bgr, byte> image, Feature[] features,
    DrawType drawPointType = rawType.ARROW);
```

trieda **Feature** - spoločná trieda pre ukladanie pre bodov záujmu a ich vlastností. V prípade, že niektorá z metód potrebuje doplniť ďalšie vlastnosti, používa jej podtriedu

```
public class Feature {
    public PointF location = new PointF();
    public double scale = 0;
    public double direction = 0;
    public Point position = new Point();
```

```
        public float[] descriptor = null;
        public double magnitude = 0;
    }
```

trieda **CDetectorOptions** - trieda bez vlastností. Každá metóda implementuje vlastnú podtriedu s pre ňu špecifickými vlastnosťami**.** O obsluhu týchto vlastností sa stará dialógové okno implementované v každej metóde samostatne.

Hlavnou triedou každej metódy je potom trieda nazvaná **CdetectorFunc** zdedená od CDetector, aby sme zabezpečili automatické nahratie triedy pri načítaní dll súboru, ktorý obsahuje implementáciu danej metódy.

3.2.2 Hlavná aplikácia

Samotná aplikácia je robená ako MDI (multi document interface) aplikácia, čo znamená, že hlavné okno aplikácie môže obsahovať ľubovoľný počet podokien. Tieto podokná sú inštancie triedy **MDIChildForm** a sú určené na uchovávanie načítaných alebo zmenených obrázkov, ako aj na zobrazovanie nájdených čŕt a informácii o priebehu detekcie vybratou metódou. Je to hlavne z dôvodu, aby sa umožnilo rýchlo a prehľadne porovnávať nájdené body, ich kvantitu a umiestnenie na konkrétnych obrázkoch na základe úsudku užívateľa, bez nutnosti generovať obrázky s črtami a porovnávať ich v externom programe. Hlavné okno je inštancia triedy **MainForm**.

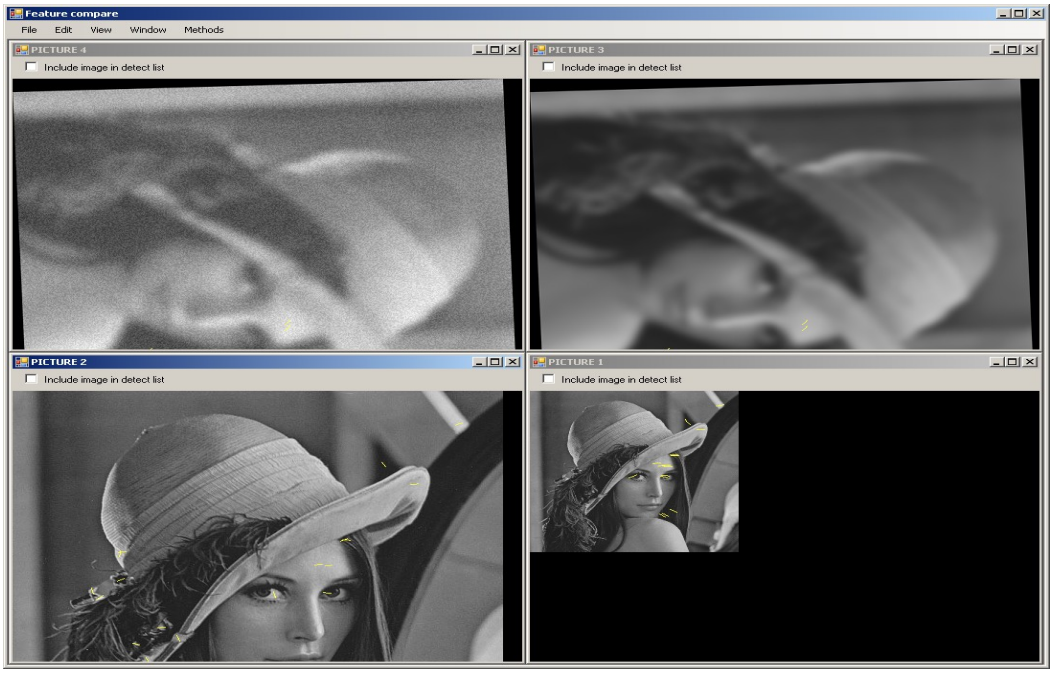
3.2.2.1 Návrh používateľského rozhrania - GUI

Menu

- **File**
  - **Load Image** - Invokuje dialógové okno pre výber súboru obrázka. Podporovanými typmi obrázkov sú základné typy, a to. .jpeg, .gif, .png, .bmp
  - **Save results** - Pre všetky MDI podokná uloží aktuálne obrázky a detekované výsledky
  - **Exit** - Ukončenie aplikácie
- **Edit** - Sekcia týkajúca sa zmien obrázkov

- **Change Image** - Invokuje dialógové okno, ktoré umožňuje meniť obrázok v aktívnom okne rôznymi transformáciami.
- **Clone Image** - Duplikuje aktívne okno s obrázkom

- **View** - Usporiadanie okien pre lepšiu prehľadnosť
  - **Cascade** - Kaskádovo
  - **Horizontal** - Vodorovne
  - **Vertical** - Zvislo
  - **Display features as** - Ponuka možnosti spôsobu zobrazenia nájdených čŕt
- **Windows** - Ponuka všetkých okien s obrázkami, ktoré boli alebo otvorené zo súboru alebo duplikované z už existujúceho okna.
- **Methods** - Ponuka automaticky nahratých metód pri štarte aplikácie na vyhľadávanie čŕt. Každá metóda ďalej obsahuje podmenu s položkami
  - **Run** - Spustí detekciu metódy na obrázku v aktívnom okne
  - **Run for checked** - Spustí detekciu metódy na všetkých oknách, ktoré sú vybraté v zozname a to spôsobom, že v danom okne je zaškrtnuté políčko **Include image in detect list**
  - **Options** - Invokuje okno s nastaveniami pre danú metódu



Obrázok 3.1: **Hlavné okno aplikácie s MDI podoknami**

Na obrázku je náhľad aplikácie s načítaným obrázkom Lena (vľavo dole) a jeho 3 modifikácie: rotácia, rozmazanie, šum (vľavo hore), rotácia, rozmazanie (vpravo hore), zmena veľkosti (vpravo dole) a zobrazené nájdené črty metódou SURF s parametrom hessianThreshold = 5000 .

3.2.2.2 Inicializácia aplikácie - inicializovanie hlavného okna a načítanie implementovaných metód na detekciu čŕt

```
public MainForm()
  {
   InitializeComponent();
   imgChangeDg = new ImageChangeDialog();
   featureMethods = new List<CDetector>();
   string[] files = Directory.GetFiles(Application.StartupPath + @
   "\features_dll\", "*.dll");

   foreach ( string f in files ) {
      LoadDll(Application.StartupPath + @"\features_dll\" +
   Path.GetFileName(f));
   }
  }

private void LoadDll(string fileName) {
   try {
      Assembly u = Assembly.LoadFile(fileName);
      Type[] tp = u.GetTypes();
      Type type = u.GetType("FeatureDetectors.CDetectorFunc");
      if (type != null) {
         CDetector featureDetector = (CDetector)
         Activator.CreateInstance(type);

         featureMethods.Add(featureDetector);
         int index = featureMethods.IndexOf(featureDetector);

         ToolStripMenuItem featureMenu = (ToolStripMenuItem)
         menuMethods.DropDownItems.Add(featureDetector.detectorName);
         featureMenu.Checked = true;
         featureMenu.Tag = index;
         featureMenu.CheckOnClick = true;
         ToolStripMenuItem featureOptMenu1 = (ToolStripMenuItem)
         featureMenu.DropDownItems.Add("Run");
         featureOptMenu1.Click += new EventHandler(OnFeatureRunClick);
         ToolStripMenuItem featureOptMenu2 = (ToolStripMenuItem)
         featureMenu.DropDownItems.Add("Run for checked");
         featureOptMenu2.Click += new EventHandler(OnFeatureRunAllClick);
         ToolStripMenuItem featureOptMenu3 = (ToolStripMenuItem)
         featureMenu.DropDownItems.Add("Options");
         featureOptMenu3.Click += new EventHandler(OnFeatureOptionsClick);
      }
   }
   catch {
      MessageBox.Show("Failed loading methods", "Error",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
   }
}
```

Načítanie obrázku - podporované formáty obrázkov sú bežne najpoužívanejšie a to sú *.jpg, *.png, *.tif, *.bmp, *.gif . O načítanie obrázku sa stará trieda **MDIChildForm** pri jej samotnom vytváraní, ktorej konštruktoru je posielaný parameter s cestou k súboru, po vybratí v dialógovom okne. Keďže vo väčšine prípadov detekcia čŕt prebieha na

šedotónových obrázkoch, a aj vybraté metódy patria do tejto skupiny, otváraný obrázok je automaticky konvertovaný do šedotónového pomocou Emgu.CV.Image konštruktora.

```
private void opDgOpen_FileOk(object sender, CancelEventArgs e) {
   if ( opDgOpen.FileName != "" ) {
      MDIChildForm newMDIForm = new MDIChildForm(this,
      opDgOpen.FileName);
   }
}
public MDIChildForm(Form MDIParent, string fileName) {
   InitializeComponent();
   loadImage(fileName);
   Init(MDIParent);
}


public void loadImage(string fileName) {
   image = new Emgu.CV.Image<Gray, Byte>(fileName);
   imageID = GenerateImageId(image);
   reloadImage();
}
```

3.2.2.3 Transformácia obrázku

Na zmenu obrázka slúži dialógové okno triedy **ImageChangeDialog**. Transformácie sú aplikované na obrázok v aktívnom MDI okne a nový vygenerovaný obrázok môže alebo nahrať pôvodný alebo vytvoriť nové MDI okno.

K dispozícii sú nasledovné možnosti zmeny obrázka. Nakoľko nejde o plnohodnotný grafický editor, jednotlivé možnosti zmien sú adaptované na potreby porovnávania.

I_N, I, I_S sú inštancie triedy Emgu.CV.Image<Gray, byte> kvôli lepšej prehľadnosti. Hodnoty jednotlivých trackbarov sú uvádzané v {}.

zmena jasu - pripočítanie rovnakej hodnoty ku všetkým obrazovým bodom

$I_N = I + k$ $\quad k = -100 < \{Brightness\}$ $\quad <-100,100>$

zmena kontrastu -

$I_N = I^*k + (1 - k) / 2$ $\quad k = \{Contrast\}/100+1$ $\quad <-100,100>$

rozmazanie - konvolúcia gausovým jadrom s veľkosťou k

$I_N = g(\sigma)^{**}I$ $\quad k = 2\{Blur\}/2 +1$ $\quad <0,100>$

$\sigma = 0.3(k/2 - 1) + 0.8$

otočenie -

$I_N = I.Rotate(k, black, false)$ $k = \{Rotation\}$ <-180, 180>

zmena veľkosti - nepotrebujeme enormnú zmenu veľkosti, ale skôr jemnejšiu škálu

- transformácia je pomocou bilineárnej interpolácie

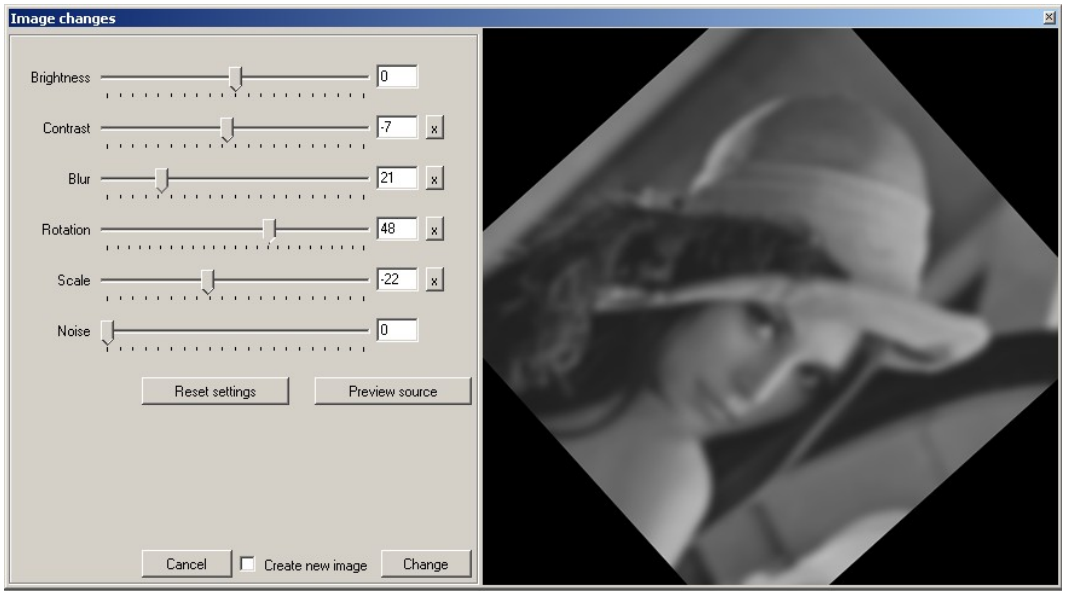
$$k = \begin{cases} \{scale\}/100 * 3 + 1 & \{scale\} > 0 \\ 1 - \{scale\}/110 & \{scale\} < 0 \\ 1 & \{scale\} = 0 \end{cases}$$

$I_N = I.Resize(I.Width*k, I.Height*k)$

pridanie šumu - šum je generovaný uniformne pomocou funkcie triedy Emgu.CV.Image

$I_S = I.SetRandUniform(new\ MCvScalar(0), new\ MCvScalar(255));$

$I_N = I + I_S*k$ $k = \{Noise\}/100$ <0, 100>



Obrázok 3.2: **Dialógové okno na zmenu obrázka**

Upravovaný obrázok je z aktívneho MDI okna a sú na neho aplikované transformácie podľa hodnôt jednotlivých trackbarov súčasne. Na porovnanie pôvodného a aktuálneho obrázka slúži tlačidlo PREVIEW SOURCE. Zaškrtnutím políčka CREATE NEW IMAGE sa pri potvrdení zmien pre transformovaný obrázok vytvorí nové MDI okno namiesto aktualizácie obrázku v aktívnom okne.

3.2.3 Implementácia metód

3.2.3.1 SIFT

Podrobný popis algoritmu je uvedený v sekcii 2.2, preto len popíšeme použité štruktúry.

trieda **SIFTOptions** trieda nastavení špecifických nastavení pre SIFT s predvolenými nastaveniami podľa [Lowe04].

```
public class SIFTOptions : CDetectorOptions {
        public int scaleLevels              = 3;
        public int maxOctaves               = 5;
        public int minImageSize             = 32;
        public double curvatureRatio        = 10;
        public double interpolateThreshold  = 0.03;
        public double peakThreshold         = 0.005;
        public bool    doubleImage          = true;
}
```

trieda **OctaveLayer** - reprezentuje vrstvu z pyramídy škálového priestoru ako aj príslušný DoG obrázok.

```
public class OctaveLayer {
      public Image<Gray, float> DoG { get; private set; }
      public Image<Gray, float> Gaus { get; private set; }
      ...
      public int level { get; private set; }
      public double scale { get; private set; }
      ...
```

Inicializácia triedy s priradením parametrov

```
      public OctaveLayer( Image<Gray, float> img, Image<Gray, float> dog, int
         _level, double _scale );
```

funkcia **FindFeatures** - metóda na vyhľadanie čŕt na danej vrstve. Pre všetky body DoG obrázku obsahuje postupne vylučovanie na kontrolu na extrém, krivosť, spresnenie polohy a škály črty a pri splnení kritérií pridanie bodu do zoznamu.

```
      public int FindFeatures(List<OctaveLayer> layers, SIFTOptions options);
```

funkcia **FilterHesian** - vracia hodnotu ľavej časti nerovnice 2.2.9 v danom bode

```
      public double FilterHessian(ref int x, ref int y) ;
```

funkcia **InterpolateFeature** - vypočítava vektor $\hat{\boldsymbol{x}}$ z rovnice 2.2.3 a vracia v premennej offset a vracia hodnotu rovnice 2.2.4 v danom bode.

```
      public double InterpolateFeature(ref int x, ref int y,  List<OctaveLayer>
         layers, out GeneralMatrix offset) ;
```

funkcia **IsExtremalPoint** - kontroluje hodnotu **pointValue** či je extrémom v 3x3 okolí bodu so súradnicami X, Y na danej vrstve.

```
        public short IsExtremalPoint(float pointValue, ref int X, ref int Y, bool
           sameLayer = false) ;
```

funkcia **AddFeature** - pridáva črtu do zoznamu.

```
        public SIFTFeature AddFeature(ref int x, ref int y, GeneralMatrix offset,
           SIFTOptions options);

    }
```

trieda **Octave** - reprezentuje oktávu pyramídy škálového priestoru, obsahuje zoznam OctaveLayer a stará sa o vytvorenie jednotlivých vrstiev s gausovým aj DoG obrázkom.

```
public class Octave {
        public List<OctaveLayer> layers { get; private set; }
        ...

        public Octave(Image<Gray, float> image, SIFTOptions options, int width, int
           height, double imageScale) {
           ...
           double k = Math.Pow(2, 1 / (double) scaleLevels);
           for ( int i = 0; i < scaleLevels + 2; i++ ) {

              Image<Gray, float> gaus = prevImage.SmoothGaussian(0, 0,
                 imageScale*Math.Sqrt(k*k-1), 0);
              Image<Gray, float> dog = gaus.Sub(prevImage);

              OctaveLayer layer = new OctaveLayer(prevImage, dog, layers.Count,
                 imageScale);
              ...

              layers.Add(layer);
              imageScale *= k;
           }
        }
```

funkcia **NextImage** vracia podvzorkovaný obrázok a vytvorený z *n*-2. vrstvy oktávy pozostávajúci z nepárnych stĺpcov a riadkov obrázku, kde $n = s + 3$ podľa sekcie 2.2.1.

```
        public Image<Gray, float> NextImage();

        ...

    }
```

Funkcia **GetFeatures** triedy **CdetectorFunc** sa potom stará o zdvojnásobenie veľkosti obrázka podľa a vybudovanie oktáv a nájdenie čŕt a následne ich vracia ako návratovú hodnotu.

```
override public Feature[] GetFeatures(Image<Gray, byte> image) {

        Image<Gray, float> startImage = image.Convert<Gray, float>();
        double imageScale = 0.5;

        int origWidth = image.Width;
        int origHeight = image.Height;

        if (((SIFTOptions) options).doubleImage) {
         startImage = startImage.Resize(startImage.Width * 2, startImage.Height * 2,
                 INTER.CV_INTER_LINEAR);
         imageScale = 1.0;
        }
```

```
List<SIFTFeature> features = new List<SIFTFeature>();
Octave octave;
int octaveCount = 0;
while (octaveCount++ < ((SIFTOptions) options).maxOctaves && (octave = new
  Octave(startImage, (SIFTOptions) options, origWidth, origHeight,
   imageScale)).scaleLevels > 0 ) {
        for ( int scaleLevel = 1; scaleLevel <= octave.scaleLevels;
                scaleLevel++ ) {
        octave.layers[scaleLevel].FindFeatures(octave.layers, (SIFTOptions)
                options);
        features.AddRange(octave.layers[scaleLevel].features);
 }

 startImage = octave.NextImage();
 imageScale *= 2;
}
```

3.2.3.2 Detekcia čŕt využitím rozkladu empirickým spôsobom - pracovný názov KHAN

Špecifické nastavenia pre túto metódu sú uložené v triede KHANOptions a štandardne predvolené podľa [Khan08]. Keďže v článku sa ale neuvádzajú všetky nastavenia (veľkosti okien, ktoré sa používajú na prechod segmentom a príslušným imf signálom, minimálna dĺžka segmentu a iné), tieto sme predvoli na základe pokusov.

```
public class KHANOptions : CDetectorOptions {
   public int wh = 5; // window size for thining edges
   public int N = 7; // minimal edge sequence
   public int s = 4;  // theta window size
   public int z = 10;  // zero crossing window size
   public float zeroCroosRatio = 0.3f; // zero crossing threshold ratio
   public int zcS = 11; // zero crossing neibourgh window size
   public float pixelDistance = 5f;

   public float joinDistance = 4f;

   // displaying partial images
   public bool showEdges = false;
   public bool showGradientImage = false;
   public bool showEdgeSegments = false;

}
```

Hlavná funkcia na detekciu čŕt GetFeatures triedy CdetectorFunc na získanie obrázka s hranami potom postupne trasformuje obrázok cez morfologické operácie využívajúc metódu MorphologyEx() triedy Emgu.CV.Image, ktorej vstupom je štrukturálny element v našom prípade 3x3 so stredovým bodom) a parameter definujúci operáciu:

otvorenia: `Emgu.CV.CvEnum.CV_MORPH_OP.CV_MOP_OPEN`

zavretia: `Emgu.CV.CvEnum.CV_MORPH_OP.CV_MOP_CLOSE`

gradientu: `Emgu.CV.CvEnum.CV_MORPH_OP.CV_MOP_GRADIENT`

Na získanie hodnoty prahu podľa sekcie 2.3.1 a samotné prahovanie sú použité metódy Emgu.CV.Image.Convolution() a Emgu.CV.Image.Threshold().

Na vytvorenie polí segmentov je použitá vlastná funkcia **BuildEdgeSegments()**, ktorá postupne prechádza body výsledného obrázka s hranami systémom „scanline" a ak je bod p=(x,y) biely, kontroluje, či niektorý z bodov (x-1, y), (x-1, y-1), (x, y-1), (x+1, y-1) v danom poradí je už pridaný do nejakého segmentu (tieto už boli spracovávané predtým a teda alebo sú čierne alebo patria nejakému segmetu), ak áno, spracovávaný bod p pridá do daného segmetu. V prípade, že v bode *p* sa spájajú rôzne segmenty, tieto sa zlúčia a body v daných segmentoch sa usporiadajú spôsobom, aby susedné body v obrázku boli susednými aj v segmente.
Ak sú hraničné body rôznych segmentov vzdialené o malú vzdialenosť (danú v nastaveniach vlastnosťou KHANOptions.**joinDistance**), tieto segmenty sú zlúčené. Následne sú odstránené segmenty s dĺžkou menšou ako KHANOptions .**N.**

Vyhľadávanie čŕt na segmente zabezpečuje funkcia **GenerateFeatures**

```
public Feature[] GenerateFeatures(List<KHANPoint> edge, Image<Gray, byte> image) {
        int N = edge.Count;
        float[] signal = new float[N];
        float[] xs = new float[N];
        float[] zero = new float[N];
```

Získanie príslušného θ signálu pre segment

```
        for (int i = 0; i < N; i++) {
                signal[i] = UpdateTheta(edge, i);
                xs[i] =i;
        }
        ...
```

Funkcia na získanie 1. IMF EMD rozkladu θ signálu pre segment. Metóda je transkripciou zdrojového kódu pre Matlab, ktorého autorom je Ivan Magrin-Chagnolleau na stránkach [EMDCode]. Na získanie kubickej interpolácie však využívame vlastnú funkciu **Spline()**, ktorá na získanie koeficientov potrebných na výpočet využíva TDMA (tridiagonal matrix algorithm) alebo tiež Thomasov algoritmus. Tento algoritmus sme použili z [TDMA].

```
        float[] imf = EMD1dIMF1(signal, null);
        ...
```

postupné prechody imf signálu na triedenie kanditádov podľa sekcie 2.3.3

```
        ...
}
```

3.2.3.3 SURF

Metódu SURF má implementovanú i samotná OpenCV aj EmguCV, takže sme využili túto možnosť a funkcia **GetFeatures** triedy **CdetectorFunc**, ktorá vracia pole čŕt jednoducho volá funkciu triedy Emgu.CV.Image **ExtractSURF.**

```
override public  Feature[] GetFeatures(Image<Gray, byte> image) {
   #region extract features from the object image

   MCvSURFParams myParams = new MCvSURFParams();
   myParams.hessianThreshold = ((SURFOptions) options).hessianThreshold;
   myParams.extended = ((SURFOptions) options).extended;
   myParams.nOctaves = ((SURFOptions) options).octaves;
   myParams.nOctaveLayers = ((SURFOptions) options).layers;

   SURFFeature[] features = image.ExtractSURF(ref myParams);
   Feature[] result = new Feature[features.Length];
   int index = 0;
   foreach ( SURFFeature ff in features ) result[index++] = new Feature(ff);
   #endregion
   return result;
}
```

Funkcia **ExtractSURF** má ako parameter inštanciu triedy McvSURFParams, ktorej sa dajú nastaviť nasledujúce parametre:

```
public class SURFOptions : CDetectorOptions {
   public double hessianThreshold = 500;
   public int extended = 0;
   public int octaves = 3;
   public int layers = 4;
}
```

4 Výsledky porovnaní

4.1 Pracovné prostredie použité pri testovaní

4.1.1 Hardvér

- Procesor: Intel® Core™ 2 Duo procesor T8300 (2.4GHz/3MB/800MHz) with nVidia Quadro NVS 140M
- Pamät: 4.0GB, 667MHz DDR2 SDRAM Memory (2 x 2048MB in dual channel)
- Grafická karta: NVIDIA QUADRO NVS 135M

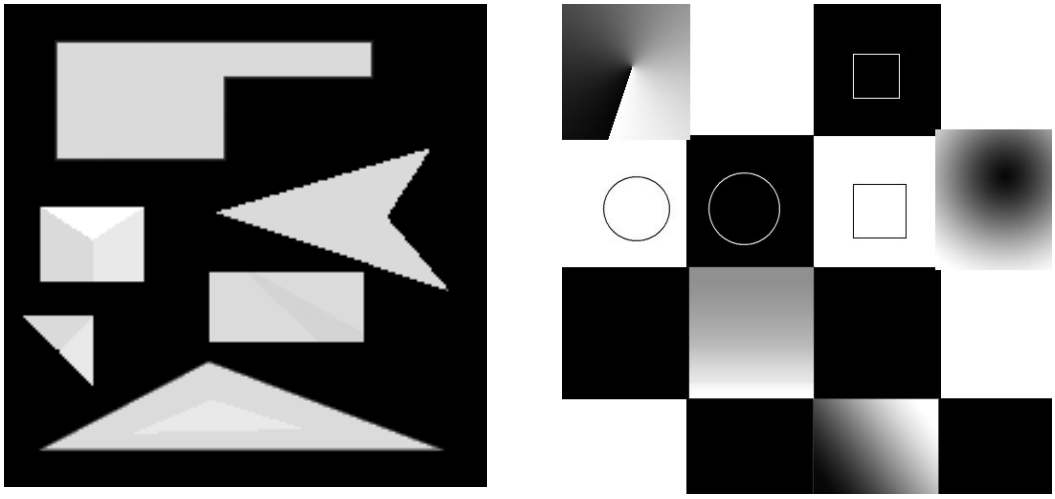
4.1.2 Softvér

- Použitý operačný systém: Genuine Windows Vista ™ Business SP1 32Bit - English - with Media
- Programovací jazyk: C#
- Programovacie prostredie: Microsoft Visual Studio 10.0 Beta2

4.1.3 Testovacia sada obrázkov

Obrázok 4.1.1: LENA.ppm 512x512 pixelov, PRIRODA.ppm 533x533 pixelov

Obrázok 4.1.2: DIVADLO.ppm 533x533 pixelov , ODTLACOK.pgm 500x696 pixelov



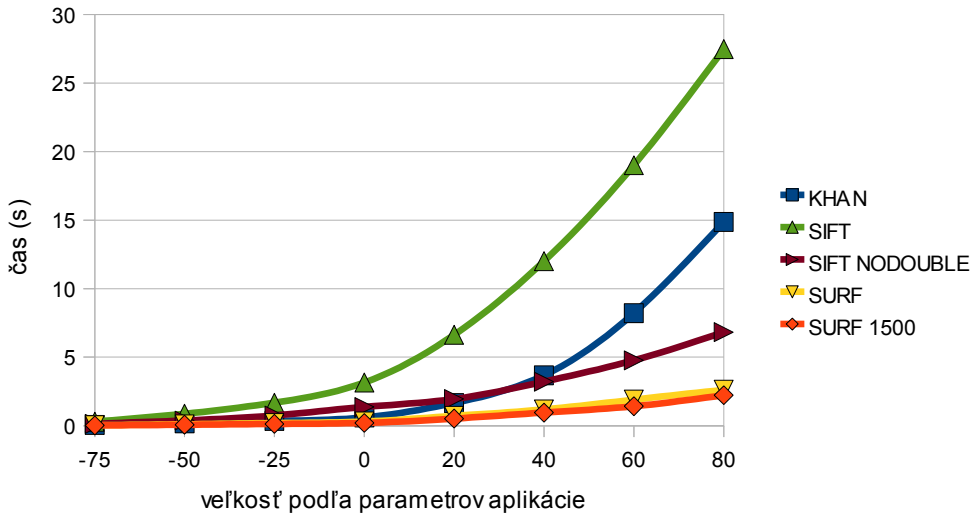
Obrázok 4.1.3: Syntetické obrázky: _syntetin.pgm 200x200 pixelov , SACHOVNICA.ppm 512x512 pixelov

4.2 Rýchlosť metód

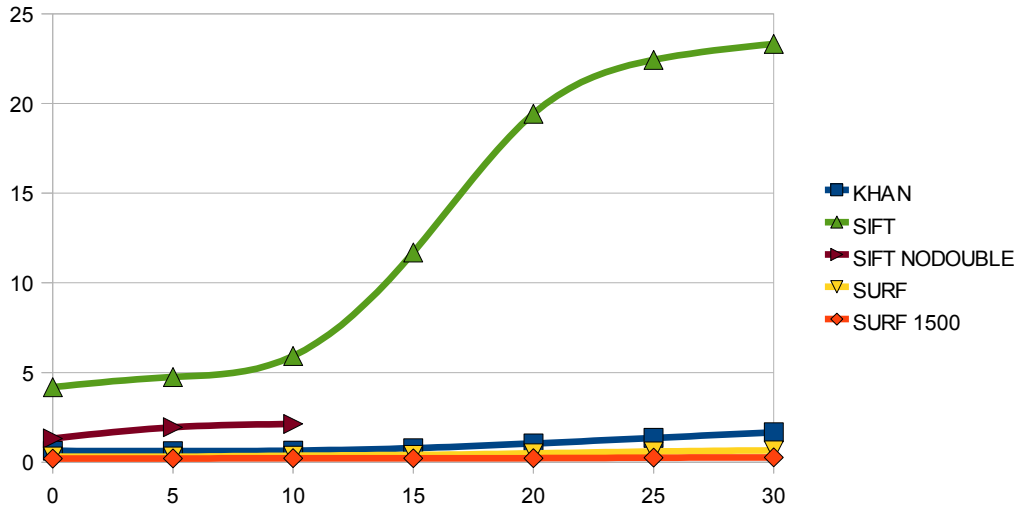
Na porovnanie sme okrem štandardne nastavených metód pridali aj 2 zmenené nastavenia, a to SURF 1500, čo je vlastne SURF s predvoleným prahom pre determinant hessianovej matice na hodnotu 1500 a ŠIFT nodouble, ktorý pozostáva z vynechania kroku zdvojnásobenia vstupného obrázku a to z dôvodu aby sme lepšie zistili vplyv týchto nastavení na výstupné detekované body a rýchlosť.

Rýchlosť metód sa potvrdila podľa očakávaní, kedy najlepšie časy dosahovala metóda SURF, potom KHAN a ako posledná skončila SIFT, ktorá oproti SURF v priemere

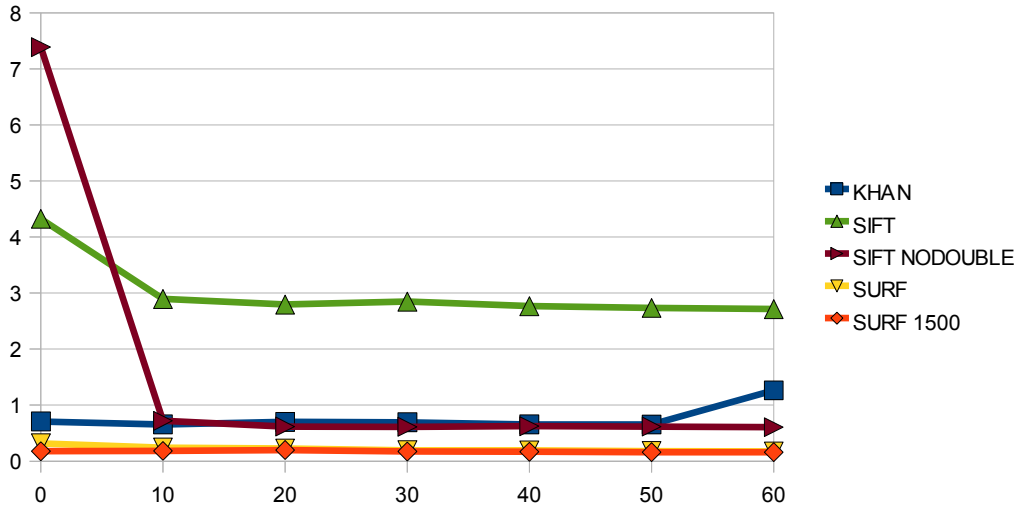
dosahovala 10x väčšie časy. Treba však ale povedať, že rýchlosti testovaných metód záviseli aj od typov obrázkov a počtu detekovaných bodov. Rýchlosť metódy SIFT narastala exponenciálne so zväčšovaním sa počtu pixlov obrázka, čo je spôsobené spôsobom, akým buduje škálový priestor. Taktiež pri veľmi veľkom počte detekovaných bodov, sa predlžoval čas výpočtu, napr. pri obrázkoch s veľmi vyšším šumom (viď graf XX). Podobne aj pri ostatných metódach sa so zvýšeným počtom detekovaných bodov narastal čas, aj keď nie tak veľmi ako u SIFT, čo súvisí s tým, že častejšie prebieha eliminácia kandidátov na základe kritérií, napr. iterácia, pre výpočet presnejšej pozície u metód SIFT a SURF. Toto vidieť aj u rozmazávaní obrázku a metóde SIFT, kedy rapídne klesal počet detekovaných bodov, naopak u metód SURF a KHAN boli časy skoro konštantné. U metódy KHAN naviac na rýchlosť vplýva štruktúra obrázku pri vytváraní hrán (ich počet a dĺžka) a k nim zodpovedajúcim θ signálom a iteratívnym výpočtom prvej IMF. Nasledujúce grafy slúžia ako ilustrácia, veľmi podobné výsledky sme dosiahli aj pri ostatných obrázkoch, preto ako príklad uvádzame zmeny na obrázku Lena.



Obrázok 4.2.1 **Rýchlosť detekcie na obrázku Lena, ktorému sa menila veľkosť**
Parametre zodpovedali zmene veľkostí -75: 0,31x rozmery obrázka, 0: pôvodné rozmery, 80: 3,4 x rozmery obrázka



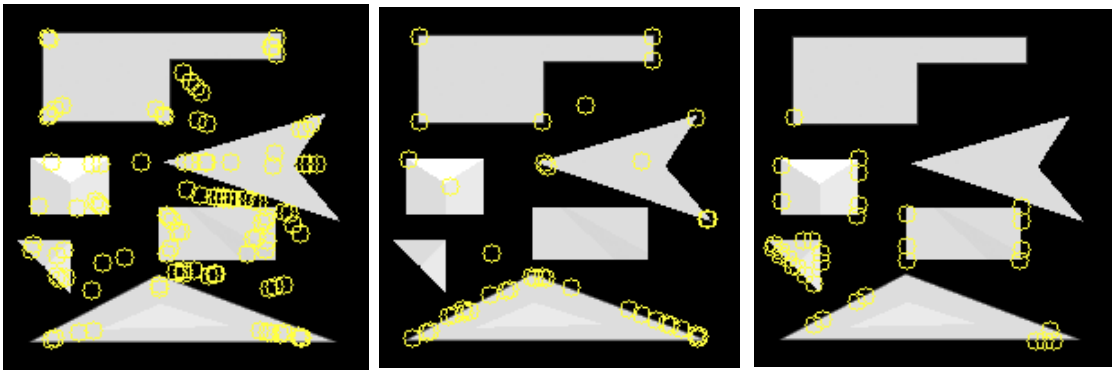
Obrázok 4.2.2 **Rýchlosť detekcie na obrázku Lena, ktorému sa pridával šum**
Parametre zodpovedali alfa kanálu šumového obrázku.



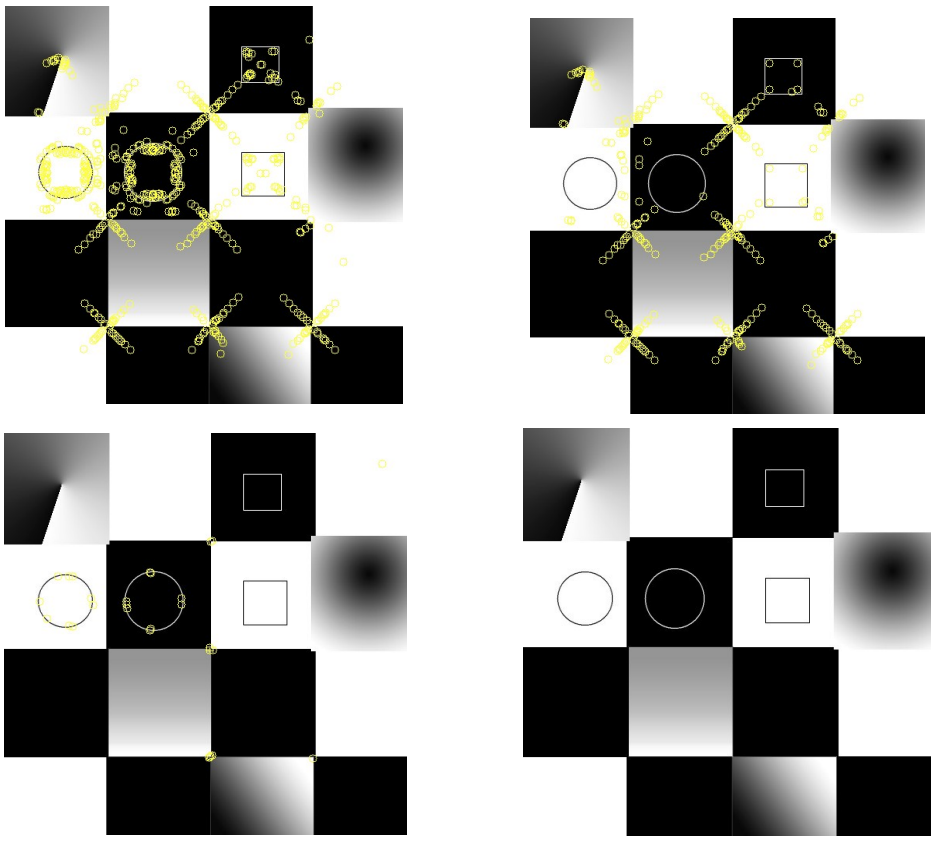
Obrázok 4.2.2 **Rýchlosť detekcie na obrázku Lena, ktorý bol rozmazávaný**
Parametre zodpovedali rozmazaniu obrázka gausovým jadrom pre σ v rozsahu $[0, 9{,}5]$.

4.3 Typy čŕt

Črty testovaných metód sa javia ako navzájom rôzne, čo môžeme vidieť na obrázkoch 4.3.1 a 4.3.2. Treba však podotknúť, že metóda KHAN vyhľadáva črty len na detekovaných hranách, ktoré boli získané automatickým prahom, čo spôsobuje isté obmedzenie v počte detekovaných bodov v rôznych typoch obrázkov oproti ostatným metódam.



Obrázok 4.3.1: **Typy nájdených čŕt pri štandardnom nastavení**
Zľava doprava: SURF, SIFT, KHAN



Obrázok 4.3.2: **Typy nájdených čŕt pri štandardnom nastavení**

Zľava doprava a zhora nadol : SURF, SURF 1500, SIFT, KHAN
Metóda KHAN nezdetekovala na tomto obrázku žiaden bod.

Metóda SURF na rozdiel od KHAN sa hranám vyhýba, keďže vyhľadáva na základe determinantu hessianovej matice. Na obrázku 4.3.2 (vľavo hore) to opticky vyzerá, že detekuje body aj na kružnici, ale v skutočnosti ich detekuje vedľa nej. Metóda SURF detekuje črty na rohoch, hranách (podľa parametra krivosti - pomeru) aj na škvrnách.

Pri typoch obrázkoch, v ktorých sú detekované hrany veľmi dlhé (čo je prípad aj obrázka 4.3.2) metóda KHAN zlyhávala. Následným skúmaním sme zistili, že podmienka eliminujúca kandidátov, pre ktorých počet prechodov IMF signálu v ich blízkosti - definovanom veľkosťou okna Wz je menší ako 1/3 všetkých prechodov cez nulu IMF signálu, je nevhodne definovaná, nakoľko pri dlhých hranách, sa zvyšuje celkový počet prechodov cez nulu a teda ja prah, kým počet prechodov cez nulu pre bod môže dosiahnuť maximum rovné veľkosti okna Wz. Je tiež možné, že veľkosti okien autori v [Khan08] menia dynamicky podľa dĺžky hrany, alebo skracovať hrany na maximálnu dĺžku, čo sme však z článku metódy nezistili.

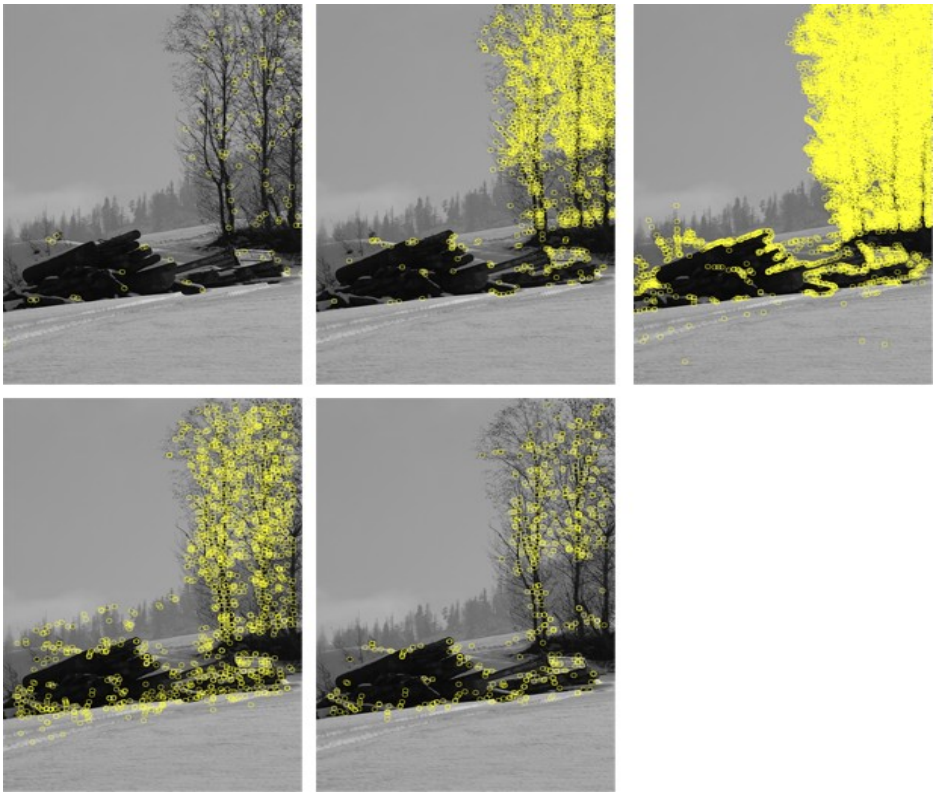
4.4 Robustnosť metód

4.4.1 Počet detekovaných bodov na rovnakom obrázku

Vo všeobecnosti počet detekovaných bodov môžeme ovplyvňovať konštantami jednotlivých metód, avšak pri štandardných nastaveniach uvádzaných ich autormi , SIFT a SURF detekujú oveľa viac bodov ako KHAN, čo je dané aj faktami spomenutými v predchádzajúcej sekcii. SURF a SIFT medzi sebou dosahujú podobné výsledky v celkovom priemere, avšak pri jednotlivých obrázkoch sa tieto hodnoty líšia vzhľadom na ich štruktúru.

	Lena	Šachovnica	Divadlo	Odtlačok	Príroda
KHAN	27	0	124	373	114
SIFT	373	29	667	6482	2124
SIFT nodouble	2189	392	4578	15370	11097
SURF	588	544	1732	5464	1573
SURF 1500	166	286	928	4224	564

Tabuľka 4.4.1: **Počet detekovaných bodov v jednotlivých obrázkoch pri štandardných nastaveniach**



Obrázok 4.4.1: **Detekované body na obrázku Príroda**
Zľava doprava: KHAN, SIFT, SIFT nodouble, SURF, SURF 1500

4.4.2 Robustnosť jednotlivých metód vzhľadom na transformácie

Z predchádzajúcich výsledkov počtu detekovaných bodov je zrejme, že zdvojnásobenie počiatočného obrázka pri SIFT metóde ma podstatný vplyv na kvalitu nájdených čŕt, keďže nezdvojnásobenie spôsobuje masívny nárast detekcie aj na "nedôležitých" miestach. Preto pri ďalších testoch budeme brať do úvahy len SIFT so štandardnými nastaveniami.

4.4.2.1 Rotácia

Počas rotácie obrázku sme sa rozhodli pre neorezávanie, teda do nového obrázku je vpísaný rotovaný obrázok a doplnený o čiernu farbu pozadie pri prázdnych častiach mimo. To nám pridalo do obrázku pri iných ako 90° rotáciách 4 nové "hrany", ktoré tvoria hranicu pôvodného obrázku a pozadia. To sa odrazilo aj na detekcii čŕt, kedy niektoré z metód detekovali body aj na týchto hranách, resp. mimo obrázka, čo najviac pozorovať u metódy SIFT.

	LENA				PRIRODA				DIVADLO			
°	K	SI	SU	SU2	K	SI	SU	SU2	K	SI	SU	SU2
0	37	373	588	166	96	1524	1192	457	221	667	1732	928
15	29	507	668	214	55	1815	1485	560	178	920	2063	1110
30	32	378	669	233	69	1737	1297	473	174	817	2107	1149
45	52	383	708	224	60	1688	1186	438	183	802	2108	1098
60	47	394	716	226	67	1753	1244	421	163	791	2091	1115
75	39	522	709	220	58	1809	1401	485	151	776	2233	1181
90	50	375	588	166	95	1521	1195	463	215	664	1730	927
105	32	510	667	220	94	1809	1489	566	151	925	2093	1126
120	48	370	672	224	88	1735	1299	475	159	804	2089	1152
135	57	388	708	223	96	1689	1193	437	215	811	2109	1100
150	37	398	729	233	80	1763	1244	422	165	793	2091	1119
165	31	524	709	223	84	1810	1419	489	168	833	2157	1115
180	36	374	588	166	134	1522	1191	464	205	653	1727	930

Tabuľka 4.4.2: **Detekované body na obrázkoch LENA, PRIRODA a DIVADLO v závislosti od rotácie obrázku**

4.4.2.2 Rozmazanie

Najlepšie s rozmazanými obrázkami si počínala metóda SURF, i keď sme pôvodne očakávali, že najrobustnejšia bude metóda SIFT. Metóda KHAN v tomto smere úplne zlyháva. V podrobnejšej analýze sme zistili, že to zapríčiňuje kritérium lokálneho maxima, nakoľko detekované hrany, na ktorých ležia kandidáti sú umiestnené na gradiente a preto kandidáti na nich nedosahujú extrémy. Od metód SIFT a SURF sa očakávala robustnosť, keďže sú škálovo invariantné.

		LENA				PRIRODA				DIVADLO			
σ	blur	K	SI	SU	SU2	K	SI	SU	SU2	K	SI	SU	SU2
0	**0**	37	373	588	166	96	1524	1192	457	221	667	1732	928
1,25	**5**	2	12	408	117	1	48	826	308	2	60	1470	836
2	**10**	0	21	257	77	0	68	493	167	0	159	1118	478
2,75	**15**	0	10	190	63	0	25	337	117	0	60	832	305
3,5	**20**	0	2	126	43	0	9	223	71	0	4	519	114
4,25	**25**	0	2	88	35	0	8	170	57	0	1	340	56
5	**30**	0	3	63	30	0	4	106	28	0	1	149	23
5,75	**35**	0	2	48	23	0	5	85	25	0	0	100	17

6,5	**40**	0	6	41	19	0	6	59	19	0	1	70	7
7,25	**45**	0	7	31	16	14	7	46	15	0	1	49	3
8	**50**	0	10	26	14	7	6	39	13	18	1	34	2

Tabuľka 4.4.3: **Detekované body na obrázkoch LENA, PRIRODA a DIVADLO v závislosti od vstupného rozmazania obrázku**

V stĺpci **blur** je hodnota parametra v aplikácia zodpovedajúca parametru σ

4.4.2.3 Šum

Pri nízkych hodnotách pridaného šumu (do 6) si pomerne dobre počínali všetky metódy. Najsenzitívnejšia na šum pri vyšších hodnotách bola metóda SIFT, čo sa dalo očakávať, keďže je postavená na deriváciách. Najlepšie si počínala metóda SURF, čo by sme mohli odvodiť použitím blokových filtrov ako aproximáciu derivácii s minimálnou veľkosťou 9. Ako naznačujú hodnoty v tabuľke, s vyšším prahom pred determinant sa dá zlepšiť senzitivita na šum, ale na úkor straty iných bodov. Metóda KHAN so zvyšujúcim sa šumom, detekuje väčšie množstvo malých hrán s minimálnou dĺžkou na ktorých pribúdajú body. Zväčšením prahu minimálnej dĺžky hrán, potom vieme čiastočne korigovať senzitivitu na šum.

	LENA				**PRIRODA**				**DIVADLO**			
šum	**K**	**SI**	**SU**	**SU2**	**K**	**SI**	**SU**	**SU2**	**K**	**SI**	**SU**	**SU2**
0	40	360	578	159	96	1524	1192	457	221	667	1732	928
3	46	418	594	161	101	1641	1246	465	186	1026	1797	943
6	59	487	655	164	118	1838	1281	495	197	1591	1885	937
9	109	679	829	192	140	1988	1426	511	181	2119	2003	969
12	112	1150	989	222	182	2387	1553	560	221	2767	2244	1029
15	194	2783	1273	259	224	3530	1756	609	216	4239	2494	1094
18	284	4654	1506	317	261	5222	1902	662	265	5720	2807	1144
21	387	5682	1801	377	412	6190	2257	737	264	6550	2953	1211

Tabuľka 4.4.4: **Detekované body na obrázkoch LENA, PRIRODA a DIVADLO v závislosti od pridaného šumu do obrázku**

V stĺpci š**um** je hodnota parametra v aplikácii, zodpovedá percentám alfa kanálu šumu pridávaného do obrázku.

4.4.2.4 Zmena veľkosti

Pri zmenšovaní obrázka sa metódy správali podľa očakávania, t.j. postupne klesajúcim počtom bodov. Naopak pri zväčšovaní obrázka sa počet detekovaných bodov zvyšoval jedine pri metóde SURF. KHAN a SIFT detekovali menší počet bodov ako v pôvodnom obrázku a to tak, že KHAN so zväčšením obrázka detekoval menší počet bodov, čiže mal klesajúcu tendenciu, naopak SIFT s prvým zväčšením klesol ale potom mal rastúcu tendenciu. Pri metóde KHAN to môžeme odôvodniť rovnako ako pri rozmazávaní, keďže pri lineárnej interpolácii medzi dvoma bodmi vzniká gradient a teda rozmazanie.

		LENA				PRIRODA				DIVADLO			
sc	**x**	**K**	**SI**	**SU**	**SU2**	**K**	**SI**	**SU**	**SU2**	**K**	**SI**	**SU**	**SU2**
-75	**0,32**	3	54	112	49	14	181	139	61	19	330	120	36
-50	**0,55**	15	131	243	81	36	343	364	155	43	550	536	231
-25	**0,77**	14	187	376	112	74	635	732	274	105	507	1110	540
0	**1**	30	354	578	159	96	1524	1192	457	221	667	1732	928
20	**1,6**	22	35	917	245	69	232	2343	902	44	141	3006	1786
40	**2,2**	25	73	1420	381	63	209	3951	1585	15	199	3965	2443
60	**2,8**	19	186	1863	478	39	439	5786	2319	15	316	4642	2861
80	**3,4**	32	211	2252	535	33	697	7585	2934	15	365	4956	3071

Tabuľka 4.4.5: **Detekované body na obrázkoch LENA, PRIRODA a DIVADLO v závislosti od zmeny ich veľkosti**

V stĺpci **sc** je hodnota parametra v aplikácii, a v stĺpci **x** je príslušné zväčšenie strán obrázku

Záver

V porovnaniach sme zistili, že tvrdenie autorov Khan a kol. [Khan08], že sa jedná o rotačne, škálovo a afinne invariantnú metódu, sa nám nepotvrdilo. Práve naopak, metóda sa ukázala ako nie veľmi vhodná na detekciu čŕt pri základných transformáciách. K tejto skutočnosti mohol prispieť aj fakt, že došlo z našej strany k chybnej interpretácii postupov z článku alebo nesprávnou implementáciou, keďže sme si nemali ako overiť správnosť algoritmu. Naproti tomu metóda SURF [Bay06] je porovnateľná s metódou SIFT [Lowe04],[Lowe99] a vzhľadom na svoju rýchlosť je veľmi vhodnou metódou k použitiu v real-time aplikáciach, napr. pri sledovaní tváre, špz a podobne. Za zmienku by stálo napríklad jej ďalšie otestovanie v metóde ASIFT [Yu09] (spomenutej v sekcii 1.4.4) namiesto metódy SIFT [Lowe04],[Lowe99], čím by sa mohli dosiahnuť podobné výsledky pri zrýchlení času, keďže SIFT sa v danej metóde vykonáva každú aproximáciu simulujúcu zmenu pohľadu na objekt.

Zoznam použitej literatúry

[Baumberg 00] BAUMBERG, A. 2000. Reliable Feature Matching across Widely Separated Views, CVPR, pp.: 1774-1781, 2000.

[Bay06] BAY, H. - TUYTELAARS, T. - VAN GOOL, L. 2006. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*. 2006

[Bay08]BAY, H., a kol. 2008. Speeded-up Robust Features (SURF). In *Computer Vision and Image Understanding: Elsevier Science Inc.* Volume 110 Issue 3, pp. 346-359, 2008

[Brown02] BROWN, M. - LOWE, D. G. 2002. Invariant features from interest point groups. In *British Machine Vision Conference, Cardiff, Wales*, pp. 656-665.

[Canny86] CANNY, J. 1986. A computational approach to edge detection. In IEEE *Transactions on Pattern Analysis andMachine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[Crowley84] CROWLEY, J. L. - PARKER, A. C. 1984. A representation for shape based on peaks and ridges in the difference of low pass transform. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 2, pp. 156–170, 1984.

[Evans09] EVANS, Ch. 2009. *Notes on the OpenSURF Library.* CSTR-09-001, University of Bristol. 2009. Dostupné na internete: http://www.cs.bris.ac.uk/Publications/Papers/2000970.pdf

[Fischler81] FISCHLER, M. - BOLLES, R. 1981. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM* , Volume 24 , Issue 6, pp. 381-395, 1981

[Harris88] HARRIS, C. - STEPHENS, M. 1988. A combined corner and edge detector. In *Fourth Alvey Vision, Conference, Manchester*, UK, pp. 147-151.

[Hartley04] HARTLEY, R. - ZISSERMAN, A. 2004. Multiple View Geometry in *Computer Vision Second Edition: Cambridge University Press*, March 2004

[Hasan00] HASAN, Y. M. Y. - KARAM, L. J. 2000. Morphological text extraction from images. In *IEEE Transactions on Image Processing*, vol. 9, no. 11, pp. 1978–1983, 2000.

[Huang98] HUANG, N. E. a kol. 1998. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. In *Proceedings of the Royal Society A*, vol. 454, no. 1971, pp. 903–995, 1998.

[Chin93] CHIN, K.-K. - SANIIE, J. 1993. Morphological processing for feature extraction. In *Image Algebra and Morphological Image Processing IV, vol. 2030 of Proceedings of SPIE*, pp. 288–302, San Diego, Calif, USA, July 1993.

[Khan08] KHAN, J. - BARNER, K. - ADHAMI, R., 2008. Feature Point Detection Utilizing the Empirical Mode Decomposition. In *Journal on Advances in Signal Processing, Hindawi Publishing Corporation EURASIP*. Volume 2008, Article ID 287061, 13 pages, doi:10.1155/2008/287061

[Kulkova07] KULKOVÁ, E. 2007. *Prehlad knižníc na podporu spracovania obrazu*: diplomová práca. Bratislava: Fakulta Matematiky, Fyziky a Informatiky Univerzity Komenského Bratislava, 2007, 99 s.

[Laganiere02] LAGANIERE, R. - VINCENT, E. 2002. Wedge-Based Corner Model for Widely Separated Views Matching. In *proceedings of the 16 th International Conference on Pattern Recognition*, Volume 3, pp. 30672, August 2002.

[Lindeberg94] LINDEBERG, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. In *Journal of Applied Statistics*, 21(2):224-270.

[Lindeberg98] LINDEBERG, T. 1998. Feature detection with automatic scale selection. In *International Journal of Computer Vision* 30 (2): pp 77--116

[Lindeberg09] LINDEBERG, T. 2009. Scale-space. School of CompuIn: Encyclopedia of Computer Science and Engineering (Benjamin Wah, ed), John Wiley and Sons, Volume IV, pp. 2495-2504, Hoboken, New Jersey, 2009. dx.doi.org/10.1002/9780470050118.ecse609 (Sep 2008)

[Mikolajczyk01] MIKOLAJCZYK, K. - SCHMID, C. 2001. Indexing based on scale-invariant interest points. In *Proceedings of the International Conference on Computer Vision*, pp. 525–531, Vancouver, Canada, 2001.

[Mikolajczyk04] MIKOLAJCZYK, K. - SCHMID, C. 2004. Scale and affine invariant interest point detectors. In *International Journal of Computer Vision* 60 (1): pp 63–86. doi:10.1023/B:VISI.0000027790.02288.f2.

[Mikolajczyk05] MIKOLAJCZYK, K. a kol. 2005. A comparison of affine region detectors. In *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[Quddus99] QUDDUS, A. - FAHMY, M. M. 1999. Fast wavelet-based corner detection technique. In *Electronics Letters*, vol. 35, no. 4, pp. 287– 288, 1999.

[Serra82] SERRA, J. 1982. *Image Analysis and Mathematical Morphology*. Academic Press, New York, NY, USA, 1982. ISBN: 0126372403

[Shi94] SHI, J. - TOMASI, C. 1994. Good Features to Track. In *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.

[Schmid98] SCHMID, C., - MOHR, R. - BAUCKHAGE, C. 1998. Comparing and evaluating interest points. In *Proceedings of the International Conference on Computer Vision*, pp. 230–235, 1998.

[Schmid00] SCHMID, C. - MOHR, R. - BAUCKHAGE, C. 2000. Evaluation of interest point detectors. In *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, 2000.

[Smith97] SMITH, S. M. - BRADY, J. M. 1997. SUSAN - A new approach to low level image processing. In *International Journal of Computer Vision,* vol. 23, no. 34, pp. 45–78, 1997.

[Lee95] LEE, J. S. - SUN, Y. N. - CHEN, C. H. 1995. Multiscale corner detection by using wavelet transformation. In *IEEE Transactions on Image Processing*. Vol. 4, no. 1, pp. 100-104. Jan. 1995

[Viola01] VIOLA, P. - JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. In: CVPR (1). pp. 511 – 518, 2001

[Yu09] YU, G. - MOREL, J.M. 2009. A Fully Affine Invariant Image Comparison Method. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), Taipei, 2009

[Tuytelaars08] TUYTELAARS, T. - MIKOLAJCZYK, K. 2008. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision* Vol. 3, No. 3 (2007) 177–280, 2008, DOI: 10.1561/0600000017

[WikiCD] Detekcia rohov, [20.4.2010]
http://en.wikipedia.org/wiki/Corner_detection

[OpenCV] OpenCV, [20.4.2010]
http://sourceforge.net/projects/opencvlibrary/files/opencv-win

[EmguCV] EmguCV, [20.4.2010]
http://www.emgu.com/wiki/index.php/Main_Page

[EMDCode] Empirical Mode Decomposition [21.8.2009]
http://www.owlnet.rice.edu/~elec301/Projects02/empiricalMode/

[TMDA] Tridiagonal matrix algorithm [25.8.2009]
http://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm

[DotNetMatrix] DotNetMatrix: Simple Matrix Library for .NET [25.8.2009]
http://www.codeproject.com/KB/recipes/psdotnetmatrix.aspx

[WikiEG] Epipolar Geometry [4.3.2010]
http://en.wikipedia.org/wiki/Epipolar_geometry