VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA PANORAMATICKÝCH FOTOGRAFIÍ

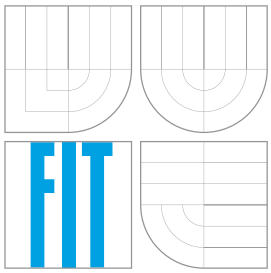
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL CACEK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA PANORAMATICKÝCH FOTOGRAFIÍ

PANORAMIC PHOTO CREATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL CACEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. KAMIL BEHÚŇ

BRNO 2015

Abstrakt

Tato práce se zabývá problematikou automatického skládání panoramatických fotografií z jednotlivých snímků. Postupně rozebírá jednotlivé kroky algoritmů, a metody v nich používané, které jsou využívány při tvorbě panoramat. Dále se zaměřuje na návrh vlastního systému založeného na diskutovaných metodách pro konstrukci panoramat. Tento systém je v rámci práce realizován pomocí knihovny OpenCV, a je k němu vytvořeno grafické rozhraní za pomoci knihovny Qt. Nakonec jsou zhodnoceny výstupy tohoto navrženého a implementovaného systému na dostupných datových sadách.

Abstract

This thesis deals with issues automatic composing panoramic photos from individual photos. Gradually examines the various steps of algorithms and methods used in them, which are used in creating panoramas. It also focuses on the design of the own system based on methods discussed to construct panoramas. This system is implemented using OpenCV library and it is created also a graphical interface using a Qt library. Finally, are in this thesis evaluated outcomes of this designed and implemented system on available datasets.

Klíčová slova

Panorama, Spojování snímků, Obrazové projekce, Harrisův rohový detektor, SIFT, SURF, RANSAC, Homografie, Integrální obraz, Graph Cuts, OpenCV, ANMS, Qt

Keywords

Panorama, Image stitching, Image projections, Harris corner detector, SIFT, SURF, RANSAC, Homografie, Summed area table, Graph Cuts, OpenCV, ANMS, Qt

Citace

Pavel Cacek: Tvorba panoramatických fotografií, diplomová práce, Brno, FIT VUT v Brně, 2015

Tvorba panoramatických fotografií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Kamila Behúňě. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Cacek
25. května 2015

Poděkování

Děkuji Ing. Kamilovi Behúňovi za odborné vedení této diplomové práce.

Obsah

Kapitola 1

Úvod

Skládání panoramatických snímků z fotografií je v dnešní digitální době často žádaná funkce. Panoramatická fotografie má velikou výhodu v tom, že je schopna zobrazit větší šířku záběru než jsou schopny vyfotit fotoaparáty. Nejprostším způsobem získání panoramata z fotografií je jejich prostý ořez a ruční poskládání v grafickém editoru. Tento přístup je pro autora velice časově náročný a není snadný.

Z toho důvodu je snaha o objevování automatizovaných přístupů ke skládání fotografií do panoramat, které po uživateli nechtějí takové profesní umělecké dovednosti a zkracují dobu vyhotovení snímku. Tudíž aplikace založené na těchto metodách mohou používat i obyčejní lidé, kteří nepotřebují znát a chápat vnitřní funkci této aplikace. Mezi aplikace, které tyto možnosti nabízejí, patří např. komerční Adobe Lightroom, nebo Zoner Photo Studio, a nebo nekomerční aplikace jako je Hugin [14] či AutoStitch [4].

V praxi má panoramatická fotografie velké využití, ať již umělecké či jako zobrazovač informace. Někdy se s panoramatickou fotografií setkáváme a ani si toho nevšimneme např. Google Street View, což jsou v podstatě složené fotografie, které umožňují virtuální procházky.

V této diplomové práci se budeme zabývat základními metodami pro automatické skládání fotografií do panoramat. Na těchto metodách bude následně navržen systém pro automatickou tvorbu panoramat a také metrika pro hodnocení těchto vytvořených panoramat.

V kapitole 2 budou popsány algoritmy používané při sestavování panoramat. Kapitola 3 popíše nalezené datové sady pro testování realizovaného systému. Dále v kapitole 4 bude rozebrán návrh systému tvorby panoramat, jehož implementace bude popsána v kapitole 5. V kapitole 4 bude také navržena metrika pro hodnocení panoramat. V následné kapitole 6 budou prezentovány výsledky implementovaného systému na snímcích datových sad z kapitoly 3. V závěru 7 budou shrnuty dosažené výsledky realizovaného systému a diskutovány jeho možné další vylepšení.

Kapitola 2

Rozbor metod používaných při tvorbě panoramat

Při vytváření panoramatického snímku je nutné se nejprve rozhodnout, zda vytvářené panorama bude planární (rovinné, perspektivní), cylindrické (válcové) nebo sférické (kulové), podle toho je nutné transformovat vstupní snímky, nad kterými jsou prováděny další operace. Existuje více přístupů ke skládání panoramatických snímků. Základní přístupy tvorby panoramat (viz. [6, 28, 30, 5, 15, 7]) jsou založeny na nalezení klíčových bodů v obraze, tyto body popisují významné body (oblasti) vstupního snímku a jsou často navíc filtrovány pro zmenšení jejich počtu (kvůli snížení časové náročnosti výpočtu). Následně je hledána korespondence klíčových bodů mezi jednotlivými snímky. Po nalezení korespondujících klíčových bodů je možné vytvořit transformační matici homografie, která převádí snímky do stejného souřadného systému. Homografie mezi danými snímky je hledána pomocí algoritmu RANSAC [6, 5], náhodně jsou testovány různé kombinace korespondujících bodů a homografie s nejlepším dosaženým výsledkem (skóre transformace) je zvolena za výslednou. Podle zvolené matice homografie jsou snímky sloučeny. Když je již složený výsledný snímek, používají se ještě metody na jeho čištění. Pod tímto je možné si představit odstranění duchů (objekty, které se vyskytují v některém z dílčích snímků před složením), hran překryvů původních obrázků apod. Tento popsaný přístup, kterým se bude práce zabývat, budu dále v práci nazývat základním přístupem. Mezi přístupy, které se odlišují od dříve popsaného přístupu patří např. tyto způsoby. Princip rozložení snímku na dva samostatné objekty, a to na oblohu a zemi, a následné hledání homografie snímků ve stejných objektech, více informací viz. [10]. Dalším možným přístupem je přístup popsaný v [12], kde se na jednotlivé snímky přiloží mřížka a hledají se korespondence částí mřížky mezi snímky.

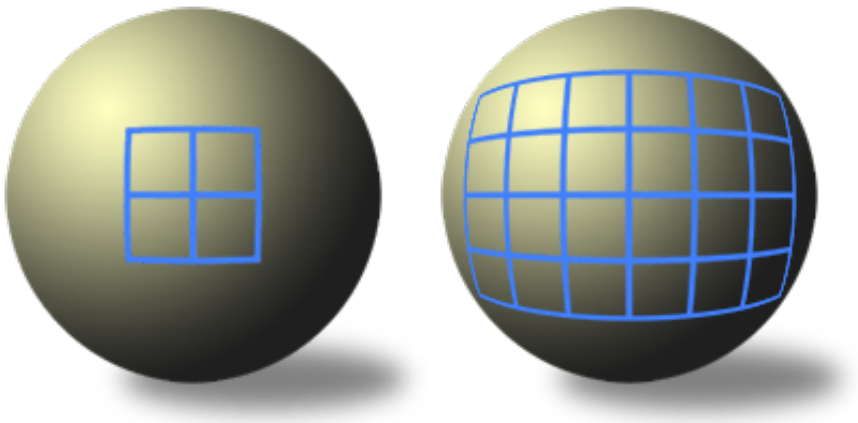
V této kapitole budou rozebrány používané obrazové projekce v panoramatických snímcích (sekce 2.1), algoritmy pro detekci klíčových bodů (sekce 2.2), filtrování počtu klíčových bodů (sekce 2.3), nalezení korespondenčních bodů mezi snímky (sekce 2.4). Dále výpočet homografie (sekce 2.6), algoritmus RANSAC (sekce 2.5), nalezení co nejlepší homografie pomocí RANSAC (sekce 2.7) a přístupy ke skládání výsledného obrazu (sekce 2.8).

2.1 Typy používaných panoramatických projekcí

Zorné pole člověka (či fotoaparátu) je možné si představit jako povrch koule ([16, 9, 13]). Transformace kulového pohledu na svět, do plochého monitoru počítače, či na papír, vyžaduje určitý způsob mapování (obrazové projekce) 3D kulové scény, ve které se nachází

fotoaparát a divák, do 2D zobrazovací plochy. Techniky používané při obrazových projekcích v panoramatické fotografii, jsou schodné s metodami, které se již dlouho využívají při tvorbě mapových podkladů (mapovaní zeměkoule do dvourozměrných map). Neexistuje jedna univerzální obrazová projekce, namísto toho je mnoho druhů projekcí s různými vlastnostmi a omezeními. V panoramatických fotografiích se nejvíce používají tři základní projekce a to planární (perspektivní), cylindrická (válcová) a sférická (kulová).

Když jsou snímky vyfocené fotoaparátem, jsou tímto snímáním již převedeny do planární projekce. Aby bylo možné snímky transformovat do jiných projekcí je nutné znát ohniskovou vzdálenost objektivu, kterým byly snímány, z těchto následně uvedených důvodů. Pro malé pozorovací úhly (dlouhá ohniska objektivů) je pozorovaná část povrchu koule (zorné pole) malá tudíž i zakřivení je nízké a pozorovaný obraz je téměř čtvercový (levá část obrázku 2.1). Naopak pro velké pozorovací úhly (krátká ohniska objektivů) je snímaná část povrchu koule velká a zkreslení je již značné (pravá část obrázku 2.1). Z toho vyplývá, že vstupem většiny projekcí (např. cylindrická či sférická) prováděných nad fotografiemi je nutná znalost ohniskové vzdálenosti, aby se mohla vyjádřit velikost zkreslení pro danou projekci (vliv velikosti ohniskové vzdálenosti při převedení snímku standardní planární šachovnice do cylindrických a sférických souřadnic viz. obrázky 2.7 a 2.8).



Obrázek 2.1: Ukázka závislosti zkreslení snímku na velikosti zorného úhlu pohledu (délce ohniska). Převzato z [9].

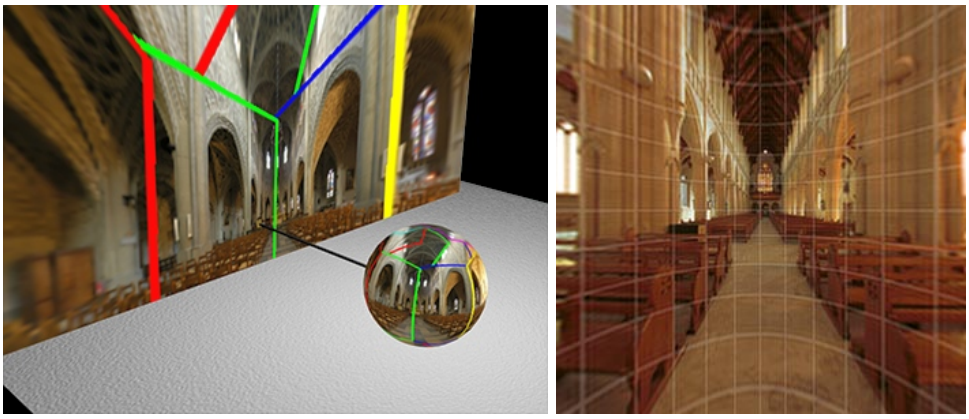
Základní definice projekcí jsou uvedeny vzhledem k převodu z povrchu koule, která reprezentuje zorné pole člověka, či fotoaparátu.

2.1.1 Planární projekce

Planární projekce (také nazývaná perspektivní, či rectilinearní) je nejjednodušší používanou projekcí při skládání panoramat. Planární projekce je definována tak, že každý pixel povrchu koule je mapován do tangenciální roviny koule (obrázek 2.2). Z toho vyplývají dvě omezení pro tuto projekci. Za prvé, že pouze pixely, které jsou čelem k projekční ploše můžou být mapovány (tzn. pouze polovina povrchu koule) do snímku, a za druhé pixely umístěné na okraji vnější hranice budou silně natažené.

Při této projekci se vstupní snímky z fotoaparátu nemusí nijak transformovat, protože vyfocením již byly přemapovány do tangenciální roviny. Výhodou je, že tato projekce zachovává rovnost horizontálních i vertikálních linií. Možnou nevýhodou této projekce (vyplývající

z druhého omezení) je to, že pomocí ní je možné skládat panoramata pouze do horizontálního i vertikálního úhlu záběru cca. 120°, větší panoramata jsou příliš deformována (obrázek 2.2).

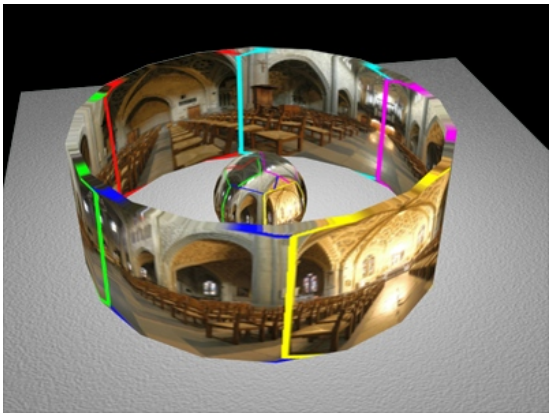


Obrázek 2.2: Na levém snímku je demonstrována projekce z povrchu koule na tangenciání rovinu. Na snímku vpravo je ukázka planární projekce. Je vidět, že rovnost vertikálních i horizontálních linií je zachována. Ze zobrazené mřížky je zřejmé, že větší úhel záběru by nepříjemně zvětšil okraje obrazu vůči středu. Převzato z [16, 13].

2.1.2 Cylindrická projekce

Cylindrická projekce (také nazývaná válcová) se často používá u 360°–vých panoramat skládaných z jedné řady snímků. V cylindrické projekci je povrch koule mapován na válec, který je okolo základní koule (obrázek 2.3). Omezením této projekce jsou případy, kdy pixely leží v blízkosti pólů, zde se vyskytne jev pozorovaný již u planární projekce a to nepříjemné roztažení těchto pixelů.

Při této projekci je nutné vstupní fotografie (perspektivní) převést do cylindrických souřadnic (viz. 2.1.4). Tato projekce zachovává rovnost pouze vertikálních linií, nikoli horizontálních. Pomocí cylindrické projekce lze promítat panoramata s horizontálním úhlem záběru 120° - 360° a vertikálním úhlem záběru do 120° (což vyplývá z omezení cylindrické projekce).



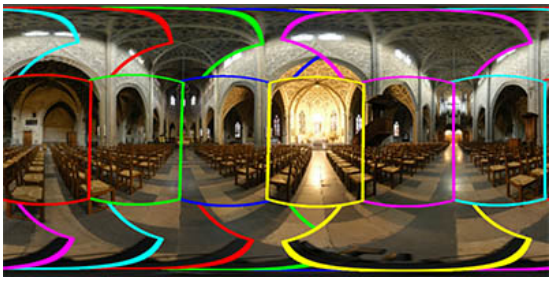
Obrázek 2.3: Na snímku je demonstrována projekce z povrchu koule na válec. Převzato z [16].

Obrázek 2.4: Na snímku je ukázka cylindrické projekce. Je vidět, že rovnost vertikálních linií je zachována, ne však linií horizontálních. Ze zobrazené mřížky je zřejmé, že pixely blížící se pólům jsou nepříjemně roztaženy. Převzato z [13].

2.1.3 Sférická projekce

Sférická projekce (také nazývaná kulová, či equirectangulární) se využívá pro skládání 360°–vých panoramat z více řad snímků. Žádné přemapování pixelů z povrchu koule není potřeba provádět, protože sférická projekce odpovídá povrchu koule (obrázek 2.5).

Při této projekci je nutné vstupní fotografie (perspektivní) převést do sférických souřadnic (viz. 2.1.5). Tato projekce obdobně jako cylindrická projekce zachovává rovnost vertikálních linií, horizontálních ne. Úhel záběru při sférické projekci je horizontálně 120° - 360° a vertikálně až 180° (obrázek 2.6).



Obrázek 2.5: Na snímku je demonstrována sférická projekce (rozbalený povrch koule). Převzato z [16].

Obrázek 2.6: Na snímku je ukázka sférické projekce. Je vidět, že rovnost vertikálních linií je zachována, ne však linií horizontálních. Ze zobrazené mřížky je zřejmé, že nedochází k žádnému nepříjemnému roztažení pixelů. Převzato z [13].

2.1.4 Inverzní cylindrická projekce

Aby bylo možné vytvořit panorama v cylindrickém projekčním prostoru, je nutné vstupní snímky nejprve převést do cylindrických souřadnic [8]. Pro správné spojení je nutné znát ohniskovou vzdálenost objektivu (úhel záběru), protože tato hodnota má rozhodující vliv na parametry projekce (při snímání fotek se s touto hodnotou mapoval obraz z povrchu koule na rovinu snímače fotoaparátu)(obrázek 2.7). V ideálním případě je také dobré znát koeficienty

radiálního zkreslení objektivu. Vzorce pro inverzní cylindrickou projekci jsou následovné:

$$\theta = \frac{(x_{cyl} - x_c)}{f} \tag{2.1}$$

$$h = \frac{(y_{cyl} - y_c)}{f} \tag{2.2}$$

$$\hat{x} = sin(\theta) \tag{2.3}$$

$$\hat{y} = h \tag{2.4}$$

$$\hat{z} = cos(\theta) \tag{2.5}$$

$$r^2 = \left(\frac{\hat{x}}{\hat{z}}\right)^2 + \left(\frac{\hat{y}}{\hat{z}}\right)^2 \tag{2.6}$$

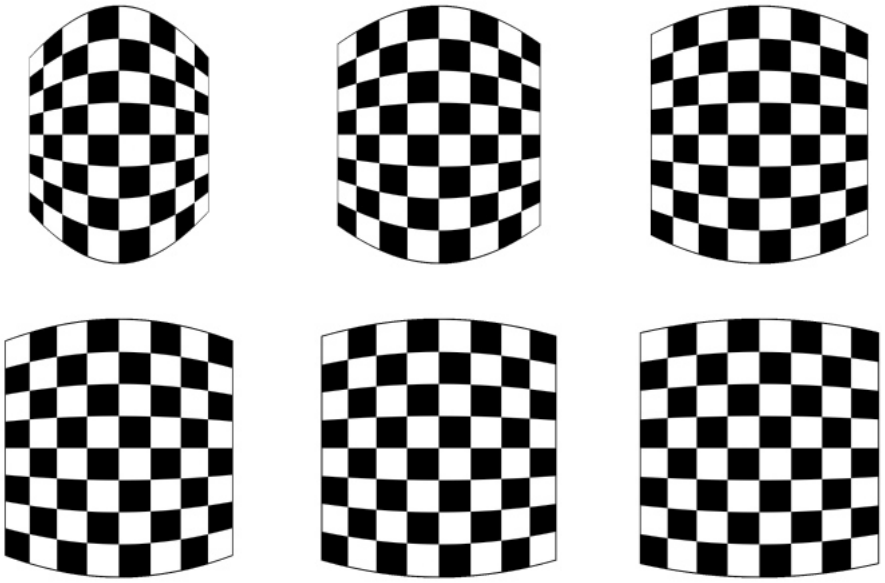
$$x_d = \frac{\hat{x}}{\hat{z}}(1 + k_1 r^2 + k_2 r^4) \tag{2.7}$$

$$y_d = \frac{\hat{y}}{\hat{z}}(1 + k_1 r^2 + k_2 r^4) \tag{2.8}$$

$$x = f x_d + x_c \tag{2.9}$$

$$y = f y_d + y_c, \tag{2.10}$$

kde x_{cyl} a y_{cyl} jsou cylindrické souřadnice pixelu výstupního snímku, x_c a y_c jsou středové souřadnice vstupního snímku, f je ohnisková vzdálenost objektivu v pixelech, k_1 a k_2 jsou koeficienty pro nápravu radiálního zkreslení objektivu a x a y jsou souřadnice odpovídajícího pixelu ve vstupním snímku. x a y nejsou většinou celá čísla, proto se pro získání hodnoty používá některá z interpolačních metod.



Obrázek 2.7: Ukázka vlivu ohniskové vzdálenosti při převodu do cylindrických souřadnic ohniskové vzdálenosti snímků: 300px/400px/500px/600px/700px/800px. Vstupem byl planární snímek šachovnice.

2.1.5 Inverzní sférická projekce

Pro vytvoření panoramatu ve sférickém prostoru je nejprve nutné převést snímky do sférických souřadnic [8]. Pro tento převod platí stejná pravidla jako pro převod do cylindrických souřadnic 2.1.4. Ukázky sférické projekce jsou na obrázku 2.8. Matematické vzorce pro převod do sférických souřadnic:

$$\theta = \frac{(x_{sfe} - x_c)}{f} \tag{2.11}$$

$$\varphi = \frac{(y_{sfe} - y_c)}{f} \tag{2.12}$$

$$\hat{x} = sin(\theta)cos(\varphi) \tag{2.13}$$

$$\hat{y} = sin(\varphi) \tag{2.14}$$

$$\hat{z} = cos(\theta)cos(\varphi) \tag{2.15}$$

$$r^2 = \left(\frac{\hat{x}}{\hat{z}}\right)^2 + \left(\frac{\hat{y}}{\hat{z}}\right)^2 \tag{2.16}$$

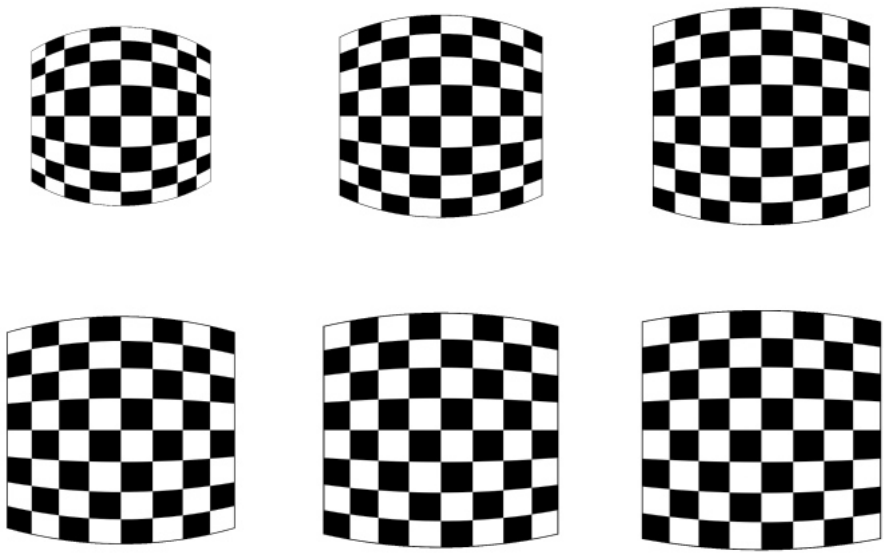
$$x_d = \frac{\hat{x}}{\hat{z}}(1 + k_1 r^2 + k_2 r^4) \tag{2.17}$$

$$y_d = \frac{\hat{y}}{\hat{z}}(1 + k_1 r^2 + k_2 r^4) \tag{2.18}$$

$$x = f x_d + x_c \tag{2.19}$$

$$y = f y_d + y_c, \tag{2.20}$$

kde x_{sfe} a y_{sfe} jsou sférické souřadnice pixelu výstupního snímku, význam ostatních koeficientů je shodný se vzorci pro 2.1.4.



Obrázek 2.8: Ukázka vlivu ohniskové vzdálenosti při převodu do sférických souřadnic ohniskové vzdálenosti snímků: 300px/400px/500px/600px/700px/800px. Vstupem byl planární snímek šachovnice.

2.2 Detekce klíčových bodů

Prvním důležitým krokem při vytváření panoramat je nalezení klíčových bodů v jednotlivých snímcích [6]. Klíčový bod snímku je místo, které má v daném snímku významnou informační hodnotu vzhledem ke svému okolí [20]. Jako klíčové body se často používají unikátní body v prostoru (osamocené body) či hrany a rohy v obraze. Důležitým požadavkem, který je kladen na detektory vzhledem k použití u panoramatických fotografií, je stálost detekce stejných klíčových bodů při změně jasu, pohybu, rotaci, zkosení či změně velikosti pohledu na scénu, aby bylo možné detekovat stejné klíčové body ve více snímcích. Vstupem detektoru je většinou snímek převedený do stupňů šedi. Výstupem detektoru klíčových bodů je buď daný bod (množina bodů), nebo vektorový (maticový) deskriptor, který popisuje daný bod v kontextu svého okolí, tudíž nese více informací o klíčovém bodu. V pracích, které se zabývají tvorbou panoramat [15, 28, 6], se jako výstup detektorů používají deskriptory, protože díky tomu, že nesou více informací o význačném bodu, je potom nalezení korespondencí ve více snímcích přesnější.

V následujících podkapitolách budou popsány některé z detektorů, jejichž použití bylo v literatuře (viz. [6, 28, 30, 5, 15]) popsáno ve spojitosti s tvorbou panoramat.

2.2.1 Harrisův rohový detektor

Některé spíše starší články [28], které se zabývají tvorbou panoramat, využívají pro hledání Harrisův rohový detektor [11]. Rohové detektory pracují tak, že v obraze hledají body, ze kterých vycházejí dvě nebo více hran. Harissův rohový detektor vychází z Moravcova rohového detektoru [11], který funguje tak, že po obrazu posouvá čtvercové okno v horizontálním, vertikálním a diagonálním směru a počítá změnu jasu v obraze, z čehož zjišťuje přítomnost či nepřítomnost rohu. Harrisův detektor odstraňuje nevýhodnou vlastnost Moravcova detektoru, kterou je posouvání čtvercového okna a namísto toho počítá gradienty obrazu pomocí derivací. Reformulací Moravcova detektoru získáme:

$$E(x,y) = (x,y)M(x,y)^T \tag{2.21}$$

$$M = \begin{pmatrix} A & C \\ C & B \end{pmatrix} = \begin{pmatrix} X^2 * w & (XY) * w \\ (XY) * w & Y^2 * w \end{pmatrix}, \tag{2.22}$$

kde X^2 je druhá parciální derivace obrazu podle x, kde Y^2 je druhá parciální derivace obrazu podle y a (XY) je derivace podle x a y. Dále jsou popsány tyto rovnice:

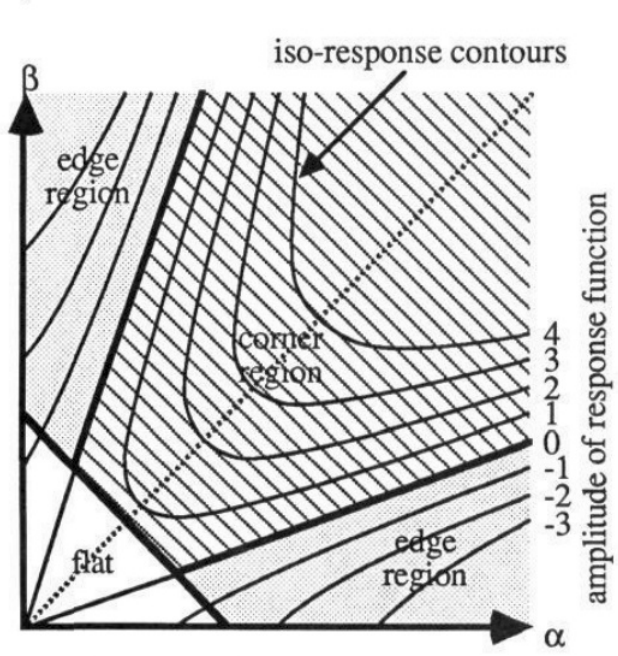
$$Tr(M) = A + B \tag{2.23}$$

$$Det(M) = AB - C^2 \tag{2.24}$$

$$R = Det(M) - k * Tr(M)^2, \tag{2.25}$$

kde Tr trace vyjadřuje tzv. stopu, což je součet prvků na hlavní diagonále, Det je determinant, k je empiricky zjištěná konstanta typicky mezi $0,04 - 0,06$. R vyjadřuje odezvu detektoru a jeho hodnota je pro nalezené rohy pozitivní, pro hrany negativní a pro rovinné útvary se blíží 0.

Na obrázku 2.9 je vidět graf závislosti A na B a také to, jak se zobrazují hrany, plochy a rohy. Jako výstup (deskriptor) Harrisova detektoru se používá matice např. 9x9 hodnot pixelů obrazu se středem v bodě, kde byl detekován roh (takovémuto deskriptoru se v technické terminologii říká **patch**).



Obrázek 2.9: Odezva Harrisova detektoru. α představuje A a β představuje B. Převzato z [11].

2.2.2 SIFT

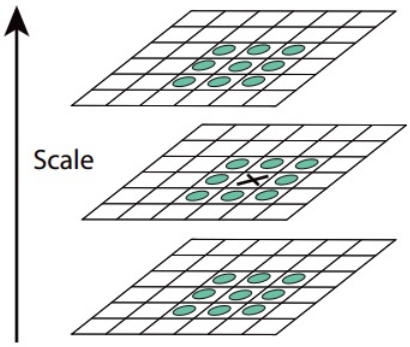
SIFT (Scale-Invariant Features) je metoda vytváření deskriptoru, který obsahuje údaje o klíčovém bodu získaném pomocí metody DoG (Difference of Gaussians). Metoda SIFT byla publikována D. G. Lowem v [20] a [21]. V následujících podkapitolách je ukázán postup vytvoření SIFT deskriptoru. Výhoda SIFT deskriptoru tkví v tom, že jeho výstup je rotačně, měřítkově i částečně jasově nezávislý. Významnou částí SIFT je metoda DoG 2.2.2, která se využívá pro nalezení klíčových bodů.

Difference of Gaussians

Metoda Difference of Gaussians (DoG) [21] je vylepšením metody Laplacian of Gaussian (LoG). Tyto metody jsou klasifikovány jako detektory významných oblastí v obraze (tzv. blobů), detekují oblasti lišící se od svého okolí barvou či jasem. Metoda LoG při svém běhu vytváří tzv. Scale space, což je měřítkově nezávislá reprezentace původního snímku. Scale space se vytváří tak, že se vezme původní snímek a z něj se postupným zmenšováním vytvoří sada snímků s různou velikostí. Nad těmito snímky se postupně provádí rozmazání pomocí Gaussovského filtru s různými odchylkami (většinou se začíná na $\sqrt{2}$ a pokračuje jejími násobky) viz. 2.26. Tím nám vznikne reprezentace m měřítek snímků s n úrovněmi rozmazání, celkově $m * n$ snímků. LoG potom za kandidáty na klíčové body určí ty, jejichž hodnota je maximální/minimální vůči svému 3x3 regionu ve třech po sobě jdoucích vrstvách (tzn. 26-okolí) v Scale space (viz 2.10 LoG je vidět v levé části obrázku), tímto jsou rychle vyřazeny neklíčové body. Rovnice pro výpočet LoG je uvedena zde:

$$L(x, y, \sigma) \quad = \quad G(x, y, \sigma) * I(x, y), \tag{2.26}$$

kde L je LoG, G je Gaussovský filtr s odchylkou σ, I je původní snímek a $*$ je konvoluční operátor.



Obrázek 2.10: LoG a DoG porovnání okolí bodu. Porovnávaný bod je označen křížkem. Převzato z [21].

Nyní zpět k metodě DoG. Ta jako základ využívá metody LoG, kde ale navíc každé dva sousedící rozmazané snímky (tzn. lišící se odchylkou Gaussova filtru σ o $\sqrt{2}$) při stejném měřítku v Scale space od sebe odečte viz. rovnice 2.27 a obrázek 2.11 v jeho pravé části. Význačný bod v DoG je nalezen obdobně jako v LoG viz. obrázek 2.10, ale vyhledává se v rozdílové vrstvě tzn. DoG vrstvě.

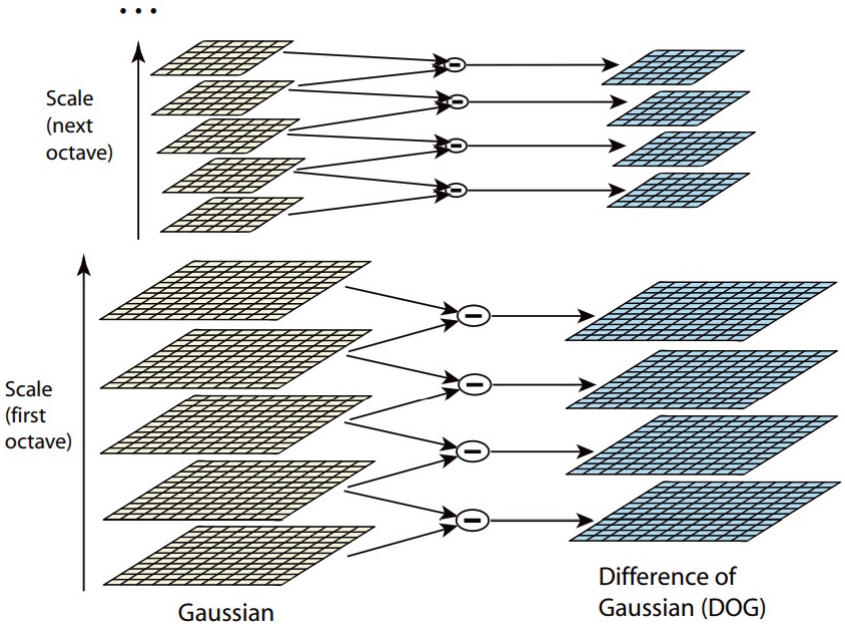
$$D(x, y, \sigma) \quad = \quad L(x, y, k * \sigma) - L(x, y, \sigma), \tag{2.27}$$

kde D je DoG, L je LoG s použitou odchylkou σ a k je koeficient násobení odchylky.

SIFT deskriptor

U metody SIFT D. Lowe (v [21]) definoval vytváření Scale space takto. Vezme se vstupní snímek a je bilineárně zvětšen na dvojnásobnou velikost, následně je rozmazán Gaussovským filtrem (takto nám vznikne obrázek L) s danou odchylkou σ. Tento snímek je dále opět rozmazán Gaussovským filtrem s odchylkou σ násobenou koeficientem k tolikrát, kolik je požadováno vrstev (Lowe doporučil 4 vrstvy). Tímto vznikne tzv. **octáva** snímky ve stejném měřítku s rozdílným rozmazáním. Tyto snímky jsou pomocí bilineární interpolace zmenšovány do požadovaného počtu měřítek, standardně se zmenšuje vždy na poloviční velikost. Oktávy jsou ukázány na obrázku 2.11. Následně jsou po sobě jednotlivé sousedící vrstvy stejné oktávy od sebe odečteny, čímž je získán DoG Scale space, zde již hledáme kandidáty na klíčové body stejným postupem popsaným v LoG. Lowe ve své práci experimenty zjistil, že dostatečný počet vrstev v oktávě u DoG Scale space jsou tři vrstvy (ty vzniknou ze čtyřvrstvé oktávy LoG).

Tyto kandidáti jsou ještě pročištěni, protože pomocí DoG vznikají nestabilní body podél hran. Body ležící na hraně s gradientem se směru hrany mají malou odezvu, naproti tomu body s gradientem kolmo na hranu mají odezvu velkou. Na toto ošetření se používá Hessiánova matice, pomocí které se určí zda je bod stabilní či není (více viz [21] kapitola 4.1).



Obrázek 2.11: Difference of Gaussians. Převzato z [21].

Následně je pro každý nalezený klíčový bod spočítána velikost $M_{x,y}$ (magnituda) a směr gradientu $R_{x,y}$ pomocí rovnic 2.28 a 2.29. Tímto výpočtem je možné zajistit nezávislost snímku na rotaci a měřítku. Odolnost proti změně jasu je dosažena tím, že gradient bodu je navíc normalizován.

$$M_{x,y} = \sqrt{(L_{x+1,y} - L_{x-1,y})^2 + (L_{x,y+1} - L_{x,y-1})^2} \tag{2.28}$$

$$R_{x,y} = tan^{-1}((L_{x,y+1} - L_{x,y-1})/(L_{x+1,y} - L_{x-1,y})) \tag{2.29}$$

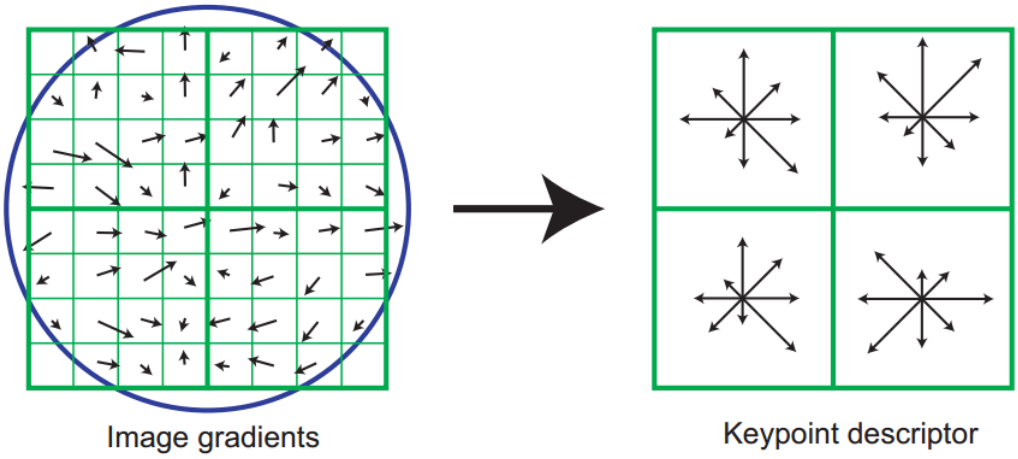
Následně se sestavuje SIFT deskriptor klíčového bodu. SIFT deskriptor se pro daný bod konstruuje s gradientů bodů v okolí daného klíčového bodu, v oktávě, ve které byl klíčový bod nalezen. Jak je ukázané na obrázku 2.12 vezme se 8x8 okolí klíčového bodu a toto okolí se rozdělí na čtyři samostatné 4x4 pod-okolí, v těchto pod-okolích je vypočten histogram o osmi binech (směrech), a ten je umístěn do pole 2x2. Toto je již výsledný deskriptor, který má velikost 2x2x8 = 32 hodnot.

Lowe ve své práci [21] pomocí experimentů vyhodnotil jako univerzální variantu brát 16x16 okolí, to rozdělit na 4x4 pod-okolí, v pod-okolích histogramy o osmi binech a 4x4 velký výsledný deskriptor, tudíž 4x4x8 = 128 hodnot.

Výsledný klíčový bod v SIFTu je popisován svou pozicí v původním snímku, směrem gradientu (natočením), magnitudou a deskriptorem popisujícím jeho okolí.

2.2.3 SURF

Metoda SURF (Speed-Up Robust Features) je další metoda vytvářející deskriptor na základě okolí vybraného bodu. Představena byla v [1] a jejím hlavním cílem bylo zrychlení vytváření a porovnávání deskriptorů, oproti metodě SIFT. Dalším důležitým cílem byla robustnost proti



Obrázek 2.12: SIFT deskriptor. Převzato z [21].

rotaci, změně měřítka, změně jasu a kontrastu ve vstupním snímku, a dle [1] ve výsledku také disponuje vyšší odolností vůči šumu než SIFT.

Podobně jako v metodě SIFT se výpočet SURF dělí na dvě fáze, na nalezení klíčových bodů a jejich následný popis pomocí deskriptoru.

Nalezení bodů zájmu je založeno na Hessově matici, u které se nepoužívá její přesný výpočet, ale pouze aproximace. Autoři SURFu (v této práci [1]) detektor nazvali prostě **Fast-Hessian Detector**, ten je založen na hledání determinantu aproximované Hessovy matice (rovnice 2.30), a využívá konvoluční jádra pro horizontální, vertikální a diagonální směr. Hessova rovnice je definována následovně:

$$\mathbf{H}(\mathbf{x},\sigma) \quad = \quad \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix}, \tag{2.30}$$

kde $L_{xx}(x,\sigma)$ vyjadřuje konvoluci vstupního snímku s druhou derivací Gaussovy funkce v bode $\mathbf{x}= (x,y)$, obdobně pro $L_{yy}(x,\sigma)$ a $L_{xy}(x,\sigma)$.

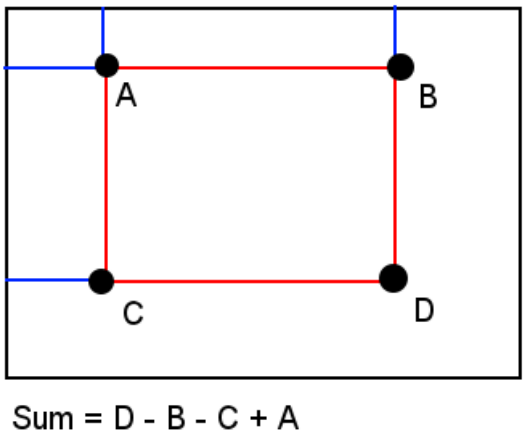
Determinant Hessovy aproximované matice v SURFu vyjadřuje hodnotu jedné položky v Scale space. Při výpočtu, se pro urychlení, využívá integrální reprezentace obrazu.

Integrální obraz

Integrální obraz je speciální reprezentace obrazu, kdy každý bod vyjadřuje součet všech předešlých bodů, které mají stejnou nebo menší x-ovou a y-novou souřadnici. To platí v případě, že souřadnicový systém obrazu má počátek v levém horním rohu. Výpočet integrálního obrazu je možné vypočítat v lineárním čase.

Výhodou použití integrálního obrazu je výpočet plochy libovolné obdélníkové či čtvercové oblasti v obrazu v konstantním času (velké zrychlení). Ukázka výpočtu plochy vyznačené body A, B, C a D je na obrázku 2.13.

Integrální obraz byl do metody SURF zaveden proto, že díky němu je možné na obraz velmi rychle aplikovat konvoluční filtry (díky vlastnostem integrálního obrazu na velikosti těchto filtrů nezáleží).



Obrázek 2.13: Znázornění výpočtu plochy mezi vrcholy A, B, C a D v integrálním obrazu. Počátek je v levém horním rohu. Převzato z [31].

Detekce klíčových bodů u metody SURF

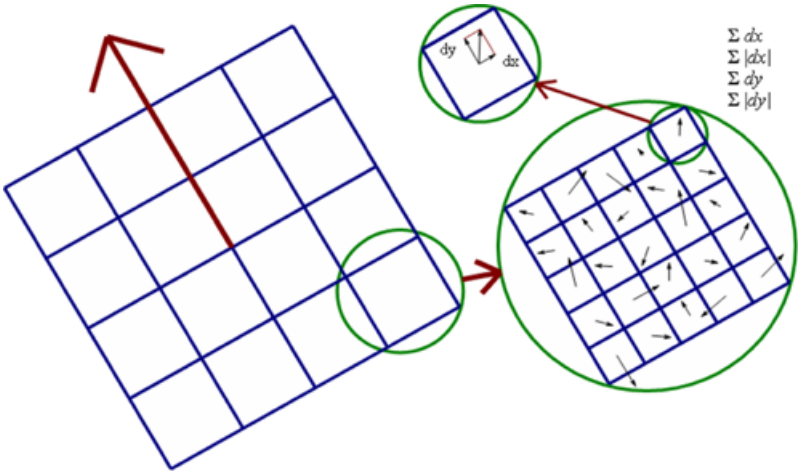
U metody SURF se stejně jako u SIFT vytváří Scale space. Vstupní snímek zůstává v původním měřítku a změna měřítka jednotlivých oktáv se dociluje zvětšováním jádra filtru, což díky používání integrálního obrazu neovlivňuje rychlost výpočtu. Jádra filtrů představují derivace potřebné pro spočítání Hessovy matice. Jednotlivé oktávy se tvoří takto. V první oktávě je nad snímkem provedena konvoluce s jádry filtrů o rozměrech 9x9, 15x15, 21x21, 27x27 tzn. zvětšují se o šest (viz. [1]), čímž vzniknou jednotlivé vrstvy oktávy. Další oktáva Scale space se vytvoří tak, že se na původní snímek použije jádro filtru stejné velikosti jako bylo použito pro druhou úroveň předcházející vrstvy (tzn. u druhé vrstvy 15x15), dál se již zvětšuje o dvojnásobek koeficientu zvětšování filtru předchozí vrstvy (pro druhou vrstvu 12, pro třetí 24 . . .), následující vrstvy jsou vypočteny obdobně. Filtry jsou na snímek aplikovány pomocí integrálního obrazu, což urychluje výpočet, a protože se všechny operace provádí nad původním snímkem (nezávisí na sobě) mohou být prováděny paralelně.

Hledání význačných bodů probíhá obdobně jako u metody SIFT. Bod je v Scale space porovnán se svým 26–ti okolím (3x3x3) a pokud je v tomto okolí maximem/minimem je označen za klíčový.

SURF deskriptor

Při popisu okolí význačného bodu v metodě SURF, se autoři snažili docílit podobných vlastností jako má deskriptor SIFT. Těmito již dříve zmiňovanými vlastnostmi je robustnost proti rotaci, změně měřítka, jasu a kontrastu. Navíc se autoři metody snažili o zmenšení deskriptoru pro urychlení výpočtu při porovnávání SURF deskriptorů mezi sebou. Požadovaných vlastností bylo docíleno tak, že je ke každému význačnému bodu s ohledem na jeho okolí spočítána jeho orientace. Výpočet orientace bodu je realizován pomocí výpočtu odezvy Haarových vlnek v okolí daného bodu. Orientace je určena ve směru, kde je největší hustota odezev na Haarovy vlnky. Velikost okolí bodu a vlnky závisí na měřítku, ve kterém byl bod nalezen (více viz. [1] kapitola 4.1).

Následně je ve směru orientace významného bodu přiloženo okno (velikost závisí na



Obrázek 2.14: Deskriptor v metodě SURF. Vlevo je vidět oblast 4x4 regionů přiložená ve směru orientace významného bodu. Vpravo znázornění výpočtu vektoru **v**. Převzato z [2].

měřítku), které oblast rozdělí do 4x4 regionů. V každém z těchto regionů, který je rozdělen na další oblasti, je v jednotlivých oblastech spočítána odezva Haarových vlnek d_x (horizontání směr) a d_y (vertikální směr) vzhledem k orientaci daného významného bodu. Následně je v každém regionu vypočítán čtyřprvkový vektor **v** viz. rovnice 2.31.

$$\mathbf{v} = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y| \right), \tag{2.31}$$

kde první dva prvky jsou sumy vlnkových odezev d_x a d_y a následující dva prvky jsou sumy absolutních hodnot vlnkových odezev $|d_x|$ a $|d_y|$. Výsledný SURF deskriptor se tedy skládá z 4x4 regionů, kde je pro každý region vypočten čtyřprvkový vektor **v**, tudíž výsledný SURF deskriptor obsahuje 64 hodnot, což splnilo snahu o snížení počtu hodnot oproti 128–mi hodnotám deskriptoru SIFT. Grafické znázornění vytvoření SURF deskriptoru ze 4x4 regionů je na obrázku 2.14.

Autoři metodu SURF v [1] porovnali s metodou SIFT a dosáhli těchto výsledků:

- Metoda SIFT nalezla o 6% více významných bodů něž SURF
- Metoda SURF byla naproti tomu 3x rychlejší než SIFT

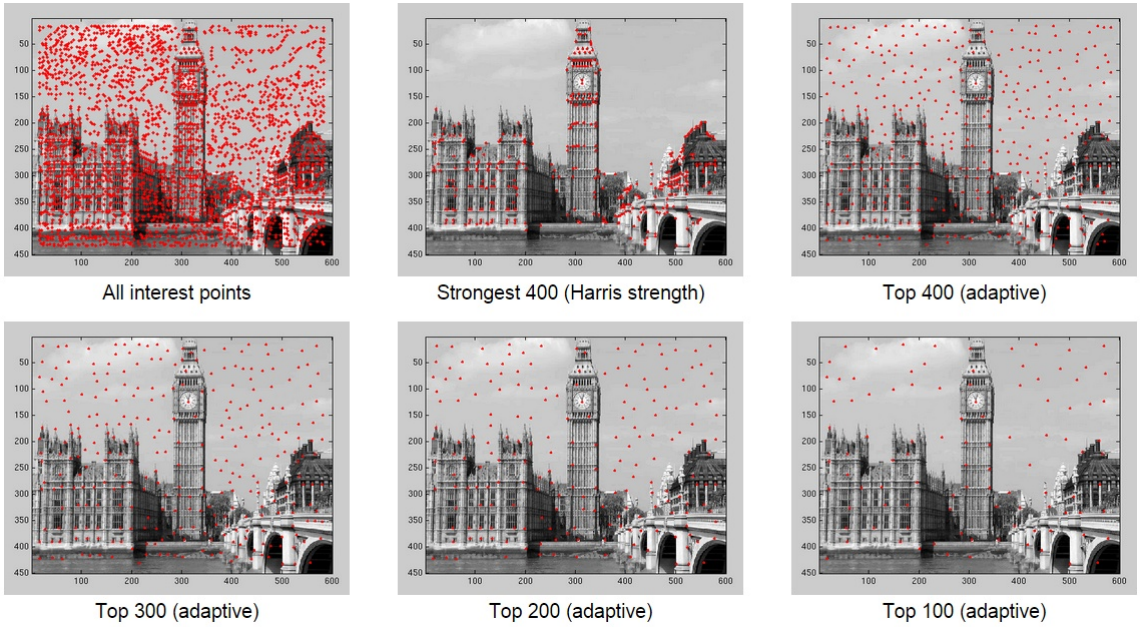
Z toho je vidět velká rychlostní výhoda metody SURF, a proto se této metody využívá např. u vestavěných zařízení, kde je snaha o velkou rychlost, ale vysoký výkon je tam často problémem.

2.3 Filtrování počtu klíčových bodů

Z důvodu velké výpočetní náročnosti hledání korespondencí mezi klíčovými body se v některých popsaných přístupech zavádí pojem filtrování počtu klíčových bodů. Jedna z používaných metod nazývaná Adaptive Non-Maximal Suppression je popsaná v [28].

Adaptive Non-Maximal Suppression

Tato metoda je zkráceně nazývaná ANMS. Cílem této metody je snížit počet klíčových bodů na stanovenou hodnotu a zároveň zachovat částečně rovnoměrné rozložení klíčových bodů po celém snímku. Tato metoda byla použita společně s Harrisovým rohovým detektorem (v [28]), protože tyto klíčové body jsou ohodnoceny kvalitou bodu (corner strength). Metoda funguje tak, že pro každý klíčový bod se vypočítá maximální radius r, což je velikost okolí, ve kterém je jeho kvalita vyšší něž kvalita ostatních klíčových bodů v tomto radiusu (tzn. až po bod s vyšší kvalitou). Následně jsou klíčové body seřazeny sestupně podle velikosti radiusu a jako výstup je ponechám požadovaný počet bodů od začátku seznamu.



Obrázek 2.15: Ukázka rozdílu při výběru bodů pomocí Harris strength, nebo podle ANNS. U ANMS vidíme, že body jsou rozloženy po celé ploše snímku což je pro následné zpracování výhodnější. Převzato z [30].

2.4 Nalezení korespondencí klíčových bodů mezi snímky

Potom, co jsou ve vstupních snímcích detekovány klíčové body, které jsou popsány pomocí deskriptorů, je nalezeno vše potřebné pro fázi, ve které budou vyhledány korespondence jednotlivých bodů mezi snímky. Výsledkem hledání bude množina obsahující dvojice bodů, které spolu mohou korespondovat. Hledání korespondujících bodů bude naznačeno mezi dvěma snímky (při více snímcích by se měnily časové složitosti).

Elementárním přístupem je porovnání každého bodu prvního snímku s každým bodem snímku druhého. Toto porovnání má časovou složitost $O(n^2)$. Porovnání se provádí tak, že se porovnají odpovídající položky deskriptorů a podobnost se měří pomocí např. Euklidovské metriky a pokud je odchylka pod určitou mezí, jsou body prohlášeny za korespondující.

V práci M. Browna [6] bylo popsáno, že časovou složitost algoritmu je možné výrazně snížit využitím k-dimenzionálního vyhledávacího stromu. Časová složitost při použití tohoto

stromu je $O(n \cdot \log n)$, tudíž výrazně příznivější.

2.5 RANSAC

Zkratka RANSAC, popisuje metodu nazývanou RANdom SAmple Consensus (česky Shoda náhodných vzorků)[23], která byla publikována Fischleren a Bollesem již v roce 1981. RANSAC je iterativní metoda, která se snaží v množině vstupních bodů nalézt hledaný model (např. přímku, kružnici ...). Iterace se provádí dokud není dosažen maximální počet iterací, nebo míra shody nalezeného řešení nepřekročí stanovený práh. Čím vyšší počet iterací je použito, tím je pravděpodobnější nalezení lepšího řešení. Algoritmus RANSAC je využíván v mnoha odvětvích počítačového vidění např. hledání geometrických objektů, hledání homografie při skládání snímků či při hledání korespondencí v obrazech.

Algoritmus je postaven na tom, že množina vstupních bodů obsahuje dva druhy bodů:

- *Inliers* - body, které patří hledanému modelu (na obrázku 2.16 označeny zelenou barvou)
- *Outliers* - body, které do hledaného modelu nepatří, jsou od něj vzdálené (na obrázku 2.16 označeny černě)

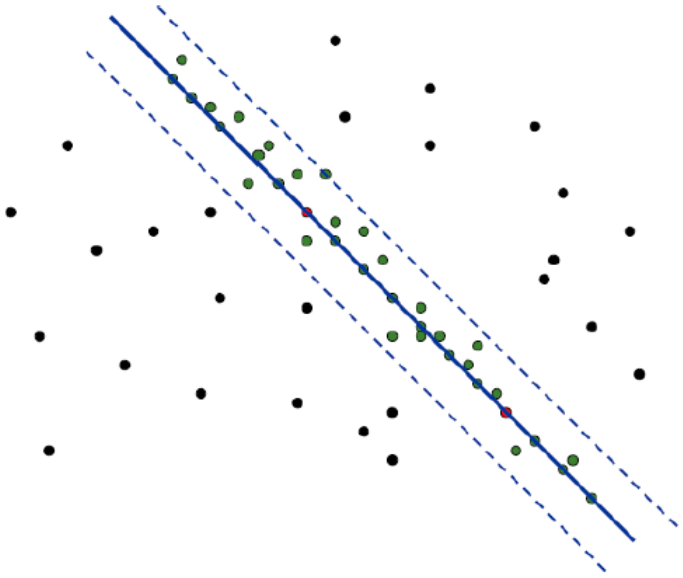
2.5.1 Činnost algoritmu RANSAC

Činnost algoritmu RANSAC je možné popsat následujícím pseudokódem (vytvořeno podle [23]).

1. Předpokládejme že máme množinu velikosti n datových bodů $X = \{x_1, x_2, \ldots, x_n\}$, u kterých doufáme, že patří do hledaného modelu, který je určen m body ($m \leq n$, např. pro přímku se $m = 2$ přímka je určena 2-mi body)
2. Nastavíme počítadlo iterací $k = 1$
3. Náhodně vybereme m bodů z X a vypočítáme model
4. Pro každý model používáme toleranční hranici e, která určí kolik prvků z množiny X náleží aktuálně vyhodnocovanému modelu (toto jsou prvky *inliers*). Pokud počet *inliers* přesáhne hodnotu prahu t je aktuálně nalezený model považován za výsledek a výpočet ukončen. Jinak je porovnán se zatím nejlepším dosaženým modelem a pokud je lepší je uschován.
5. Hodnota k je zvýšena ($k+ = 1$) a pokud je splněno $k \leq K$ jde se na bod 3. Jinak bylo dosaženo maximálního počtu iterací a jako výstup algoritmu se vrací nejlepší dosažený model.

Vysvětlení použitých zkratek v algoritmu

- m = minimální počet bodů pro výpočet hledaného modelu
- k = počítadlo iterací
- K = maximální počet iterací
- e = tolerance při rozhodování zda bod patří k modelu



Obrázek 2.16: Ukázka nalezení přímky v množině bodů pomocí RANSAC. Převzato z [23].

- t = práh zda jsme našli hledaný model

Funkci popsaného algoritmu můžeme vidět na obrázku 2.16. Za m byly vybrány body na obrázku označeny červeně, které vyjadřují model přímky znázorněný modře. Čárkovanou modrou čárou jsou vyznačeny hranice e. Body, které patří k modelu (spadají mezi hranice) tzv. *inliers* jsou označeny zeleně. Ostatní body *outliers* jsou černé.

Jaký použít maximální počet iterací metody RANSAC je možné vypočítat podle následujících vztahů:

$$P_g = \frac{pocet\ inliers}{celkovy\ pocet\ bodu} \tag{2.32}$$

$$P_{fail} = (1 - P_g^m)^k \tag{2.33}$$

$$k = \frac{\log P_{fail}}{\log(1 - P_g^m)}, \tag{2.34}$$

kde P_g pravděpodobnost, že náhodně vybrané body (data) náleží vybranému modelu, P_{fail} pravděpodobnost, že algoritmus nenalezne řešení, k počet iterací, které je potřeba uskutečnit při hledání modelu.

2.6 Homografie

Homografie vyjadřuje, jak je možné mapovat body jednoho snímku na snímek druhý. Její nalezení je významným bodem při spojování snímků. Homografie vyjadřuje transformaci mezi obrazy. Transformační matice homografie má ve 2D rozměry 3x3 a je znázorněna v rovnici 2.37.

$$p_a = \begin{bmatrix} x_a \\ y_b \\ 1 \end{bmatrix} \tag{2.35}$$

$$p_b = \begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} \tag{2.36}$$

$$H_{ab} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2.37}$$

$$p_b = H_{ab} p_a \tag{2.38}$$

$$p_a = H_{ba} p_b \tag{2.39}$$

$$H_{ba} = H_{ab}^{-1} \tag{2.40}$$

Předcházející rovnice ukazují použití matice homografie při mapování bodů mezi snímky. Jednotlivé položky matice homografie vyjadřují různé geometrické transformace, a proto budou alespoň základní geometrické transformace popsány v následující podkapitole.

2.6.1 Geometrické transformace

V současné počítačové grafice patří geometrické transformace mezi nejpoužívanější operace. Geometrické transformace můžeme chápat jako změnu pozice vrcholů v aktuálním souřadnicovém systému nebo jako změnu souřadnicového systému.

Aby bylo možné se všemi geometrickými transformacemi pracovat jednotně, jako se součinem matic a vektorů, byly zavedeny *homogenní souřadnice bodu*. Homogenní souřadnice umožňují provádět složité geometrické transformace vynásobením vektoru bodů výslednou transformační maticí, která vznikla prostým skládáním jednoduchých transformačních matic za sebe (násobení jednotlivých matic) [18].

Homogenní souřadnice bodu ve 2D s kartézskými souřadnicemi $[x, y]$ je uspořádaná trojice $[X, Y, w]$, pro kterou platí $x = X/w, y = Y/w$. Bod je svými homogenními souřadnicemi určen jednoznačně. Souřadnici w nazýváme váhou bodu. Hodnota váhy je většinou $w = 1$, v případě lineárních transformací.

Transformace ve 2D pomocí homogenních souřadnic

Zde jsou ukázány základní geometrické transformace používané v počítačové grafice.

Posunutí - bodu o d ve směru dané osy

$$x' = x + d_x \tag{2.41}$$

$$y' = y + d_y \tag{2.42}$$

$$\begin{bmatrix} x', y', 1 \end{bmatrix} = [x, y, 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{bmatrix} \tag{2.43}$$

Rotace - o úhel α s středem otáčení v počátku souřadného systému

$$
\begin{aligned}
x' &= x \cdot \cos\alpha - y \cdot \sin\alpha && (2.44)\\
y' &= x \cdot \sin\alpha + y \cdot \cos\alpha && (2.45)\\
[x', y', 1] &= [x, y, 1] \cdot \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} && (2.46)
\end{aligned}
$$

Změna měřítka - bodu o faktor S ve směru dané osy

$$
\begin{aligned}
x' &= x \cdot S_x && (2.47)\\
y' &= y \cdot S_y && (2.48)\\
[x', y', 1] &= [x, y, 1] \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} && (2.49)
\end{aligned}
$$

Zkosení - bodu v rovině o faktor zkoseni S_h

$$
\begin{aligned}
x' &= x + S_{hx} \cdot y_1 && (2.50)\\
y' &= y + S_{hy} \cdot x_1 && (2.51)\\
[x', y', 1] &= [x, y, 1] \cdot \begin{bmatrix} 1 & S_{hy} & 0 \\ S_{hx} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} && (2.52)
\end{aligned}
$$

2.6.2 Výpočet homografie

Matice homografie H se vypočítá následovně (dle [17]). Pro každou dvojici korespondujících bodů p a p' platí:

$$wp' = Hp, \tag{2.53}$$

kde w je parametr měřítka (homogenní složka). Pokud $p = [x, y, 1]$ a $p' = [x', y', 1]$ jsou body, které spolu korespondují, tak můžeme psát:

$$\begin{bmatrix} wx' \\ wy' \\ w1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.54}$$

Následným roznásobením získáme tři algebraické rovnice, kde vyjádřením w ze třetí rovnice a jeho dosazením do předchozích dvou a převedením hodnot na levou stranu, dostaneme dvě lineární algebraické rovnice:

$$
\begin{aligned}
h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' - h_{33}x' &= 0 && (2.55)\\
h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' - h_{33}y' &= 0 && (2.56)
\end{aligned}
$$

Tyto rovnice si vytvoříme pro všechny korespondence (pro homografii dvou fotografií se používají čtyři korespondence). Pro n korespondencí získáme lineární systém A o $2n$ lineárních rovnicích. Řešením homografie je soustava rovnic o tvaru $Ah = 0$, kde h je vektor koeficientů homografní matice. Tento systém můžeme zapsat takto:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' & -y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -y_nx_n' & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n' & -y_ny_n' & -y_n' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \tag{2.57}$$

Vzhledem k faktu, že byly použity homogenní souřadnice, má matice A osm stupňů volnosti. Hodnoty h můžeme pro dané korespondence vypočítat pomocí SVD rozkladu (Singular value decomposition).

2.7 RANSAC a homografie

Ke zjištění, jakou matici homografie použít, se využívá tato metoda nastíněná v [23]. Vezme se množina všech nalezených korespondencí a použije se RANSAC takto. RANSAC v každé iteraci vybere náhodně čtyři korespondence, pro ty se vypočítá matice homografie a ohodnotí se všechny korespondence např. pomocí Symmetric transfer error (také nazýváno back-projection error) 2.58, což je metoda, která hodnotí, jak se transformace povedla.

$$d^2_{transfer} = d(x, H^{-1}x')^2 + d(x', Hx)^2, \tag{2.58}$$

, kde x a x' jsou korespondenční body, H je homografie z prostoru bodu x do prostoru bodu x' a d funkce výpočtu euklidovské vzdálenosti bodů.

Podle této hodnoty se určí úspěšnost dané matice homografie a pro finální použití se algoritmem RANSAC vybere ta, která se při běhu jevila jako nejlepší (nejmenší Symmetric transfer error).

2.8 Složení snímku

Podle zjištěné matice homografie se převede jeden snímek do souřadného systému snímku druhého a tyto snímky se spojí. Pro zjednodušení uvedu postup pro dva snímky, kdy snímek **A** je statický a pomocí homografie jsme transformovali snímek **B**. Snímek **B** v souřadném systému snímku **A** leží napravo od **A**.

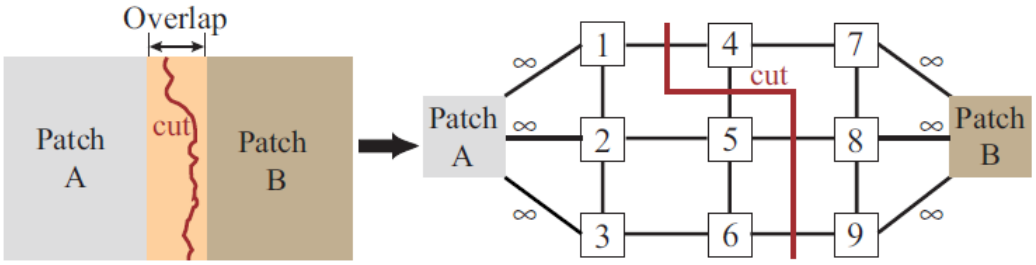
Pro vytvoření výsledného snímku je nejdříve nutné vypočíst jeho rozměry, to se provede tímto způsobem. Snímek **B** již byl transformován pomocí homografie a nyní již je v souřadném systému levého snímku. Šířku výsledného snímku vypočteme tak, že vezmeme x-ovou souřadnici v pravém horním a dolním rohu transformovaného snímku **B** a větší z pozic udává šířku snímku (je předpokládáno, že levý snímek je v počátku souřadného systému). Výška je zjištěna obdobně, ale kontrolují se i rohy snímku **A** (pokud by pravý byl transformací

zmenšený). Také se u výpočtu výšky řeší situace, kdy snímek **B** v horní části přesahuje nad snímek **A** (souřadnice y jsou záporné), v tom případě jsou oba obrázky posunuty v ose y o příslušnou hodnotu (pomocí matice posunutí).

Následně je vytvořen výsledný snímek s danými rozměry a **A** i **B** snímky jsou do něj nakopírovány. Pixely v překrývajících se oblastech jsou průměrovány, nebo vzaty jen z jednoho snímku, čímž vznikají artefakty (duchové, hrany atd.). Aby se při spojování zamezilo vzniku artefaktů v místech překryvu dílčích snímků, používá se metoda Graph Cuts (popis podkapitola 2.8.1) [19, 30].

Při spojování více snímků se všechny snímky převádí do souřadného systému jednoho vybraného hlavního snímku. Pro všechny snímky se potom obdobně dopočítají maximální a minimální x a y souřadnice a z nich se vypočítá velikost výstupu a také matice posunutí ze záporné oblasti (snímky nalevo od vybraného hlavního snímku). Další postup je shodný s postupem v předchozím odstavci.

2.8.1 Graph Cuts



Obrázek 2.17: Ukázka nalezené hranice mezi překrývajícími se částmi snímku v levé části. V pravé části ukázka vytvořeného grafu Patch A odpovídá v textu popsanému uzlu **Source** a Patch B odpovídá uzlu **Sink**. Převzato z [19].

Pomocí této metody (popsána v [19]) se rozhodne, které pixely výsledného snímku (v překrývající se části), jsou brány z prvního původního snímku, a které z druhého (dále budu první snímek označovat jako **A** a druhý jako **B**). Toto se rozhodne tak, že pomocí metody Graph Cuts se mezi snímky nalezne hranice (hranici můžeme vidět v levé části obrázku 2.17), následně se pak pixely z jedné strany hranice vezmou ze snímku **A** a z druhé strany hranice ze snímku **B**. Body této hraniční linie by měli v ideálním případě vyjadřovat pixely, jejich hodnoty se v obou snímcích rovnají (tzn. hranice vyjadřuje linii nejpodobnějších pixelů v překryvu). Tato hranice se hledá tak, že se z překrývajících se části snímků sestaví graf. Pixely v překrývající se části snímků tvoří uzly tohoto grafu a mezi sousedícími pixely (uzly) ve čtyř-okolí (horní, pravý, dolní a levý sousední pixel) jsou vytvořeny hrany grafu a tyto hrany jsou ohodnoceny následující funkcí.

$$M(s,t,\mathbf{A},\mathbf{B}) = \|\mathbf{A}(s) - \mathbf{B}(s)\| + \|\mathbf{A}(t) - \mathbf{B}(t)\|, \tag{2.59}$$

kde **A** a **B** jsou dva překrývající se snímky, s a t jsou dva sousední pixely (uzly) v překrývající se oblasti a $\|$ představuje normu (např. euklidovskou vzdálenost). Dále jsou vytvořeny ještě uzly **Source** a **Sink**. Uzly grafu sousedící s nepřekrývající se částí snímku **A** jsou

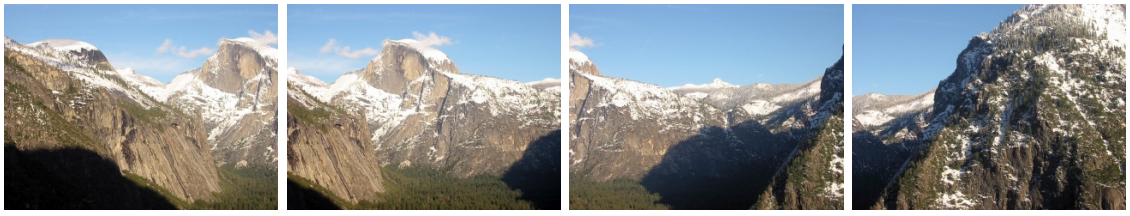
s nekonečným ohodnocením hrany připojeny k uzlu **Source**, obdobně jsou uzly sousedící s nepřekrývající se částí snímku **B** připojeny hranami k uzlu **Sink**. Následně je tento graf pomocí algoritmu minimálního řezu (Minimum cut) rozdělen na dva podgrafy. Jeden podgraf je díky nekonečným vahám připojen k uzlu **Source** to jsou pixely, které budou vzaty ze snímku **A**, druhý podgraf okolo uzlu **Sink**, pixely snímku **B**.

Kapitola 3

Popis datových sad

Pro testování vytvářené aplikace, jejíž návrh bude popsán v kapitole 4, jsem hledal veřejně dostupné datové sady pro panoramatické snímky. Takovéto datové sady jsem nalezl čtyři. U tří z těchto sad je uvedena ohnisková vzdálenost objektivu, kterým byly snímky pořizovány, čímž jsou vhodné pro skládání do cylindrických a sférických panoramat. Obsah datových sad je následující.

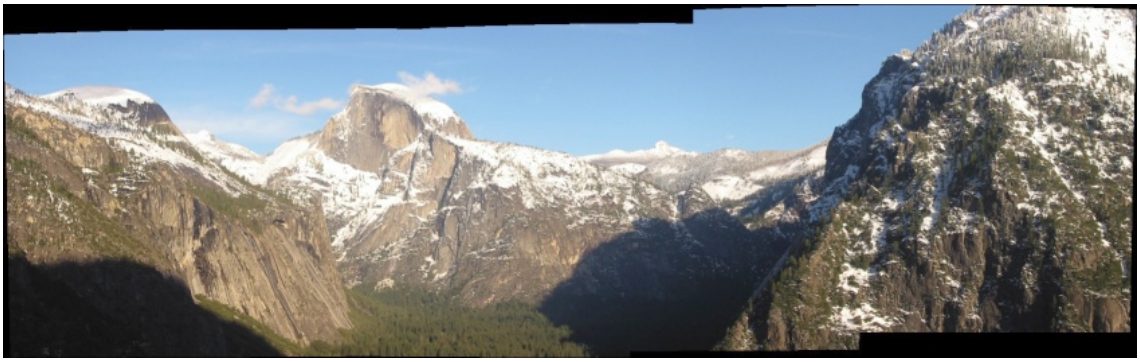
První datovou sadou je sada s názvem Adobe Panoramas Dataset [3]. Sada obsahuje 10 obrázkových sad pro tvorbu panoramat, snímky jsou v odstínech šedi. V této sadě jsou snímky takové, že posun mezi jednotlivými snímky sady je v horizontální i vertikální rovině. Nevýhodou této sady je absence údaje o ohniskové vzdálenosti objektivu, což znesnadňuje použití při projekcích s větším horizontálním úhlem záběru.



Obrázek 3.1: Ukázka jedné ze sad (`yosemite`) obsažených v druhé datové sadě [26]. Názvy jednotlivých snímků jsou zleva `yosemite1`, `yosemite2`, `yosemite3` a `yosemite4`. Názvy jsou uvedeny z důvodu, že některé části implementace (kapitola 5) budou demonstrovány na těchto snímcích, tak aby bylo možné se odkazovat jmény.

Další datová sada (viz. [26]) je již barevná a obsahuje tři sady snímků. Snímky této sady jsou zajímavé v tom, že první podsada (nazvaná `campus`) obsahuje snímky, na kterých se vyskytuje mnoho travnatých oblastí, a proto bude zajímavé sledovat, jak si s tím poradí detektor klíčových bodů. Snímky druhé podsady (nazvané `trees`) zase obsahují hodně asfaltových cest a kvetoucích stromů, kde by mohly být problémy se správnou detekcí korespondenčních klíčových bodů. A třetí podsada (nazvaná `yosemite`) obsahuje snímky hor (tato podsada je zobrazena na snímcích 3.1 a 3.2). Výhodou této celé datové sady je to, že u všech snímků je uvedena ohnisková vzdálenost a navíc i korekční parametry radiálního zkreslení k_1 a k_2.

Třetí datová sada (viz. [27]), obsahuje 3 podsady. Na jedné (nazvané `grass`) se vyskytuje množství zeleně, což je obdobné jako u předchozí datové sady. Druhá podsada, která je nazvaná `department` (obrázek 3.3) obsahuje snímky cihlové budovy s kamenným nádvořím



Obrázek 3.2: Výsledné panorama složené ze snímků (`yosemite`) druhé datové sady [26], která je zobrazená na obrázku 3.1. Složeno pomocí AutoStitch [4].

a třetí podsada (nazvaná `camp`) obsahuje snímky nasnímané v počítačové hře, tudíž obsahující počítačově vytvořenou grafiku, u které bude zajímavé pozorovat, jak budou vypadat výsledná panoramata. U této sady jsou také uvedeny ohniskové vzdálenosti.

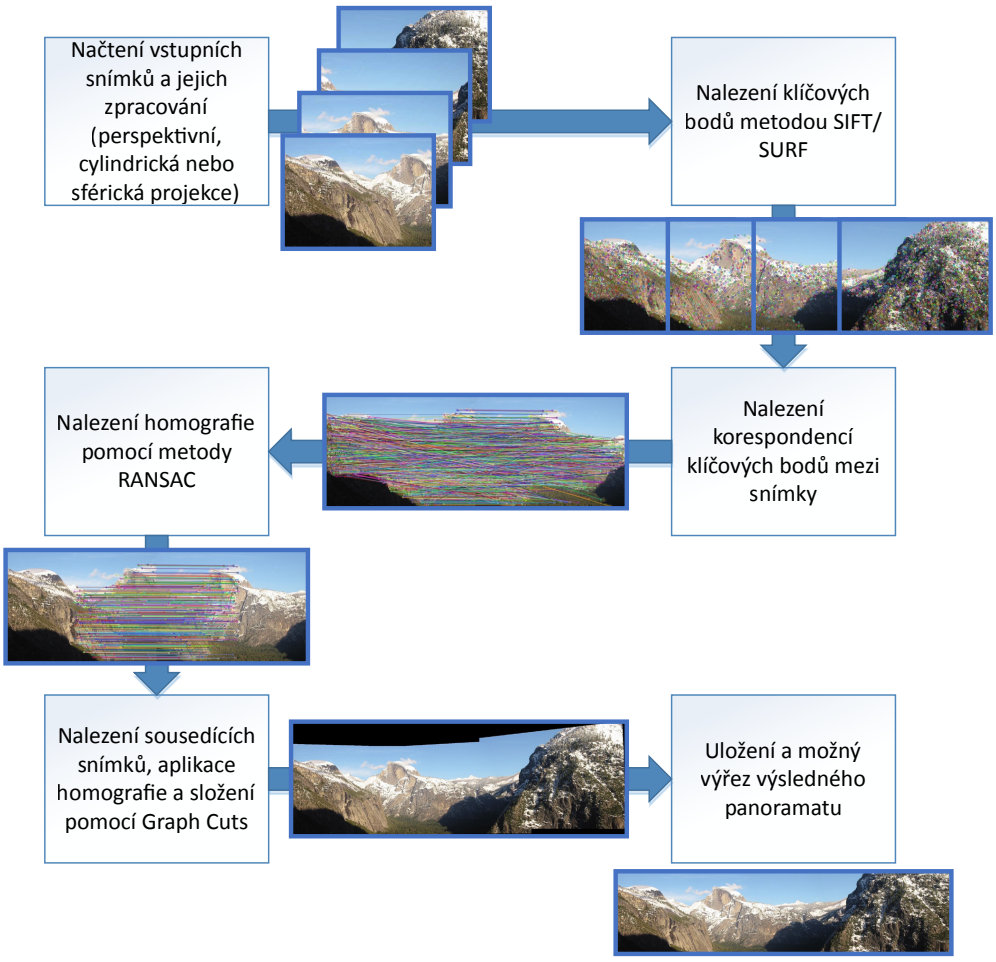
Čtvrtá a poslední datová sada (viz. [29]) se skládá z dvou podsad. Výhodou této sady jsou velké překryvy mezi jednotlivými snímky a navíc znalost ohniskové vzdálenosti a korekčních členů radiálního zkreslení objektivu. Datové podsady se nazývají `cars` a `park`.

Obrázek 3.3: Ukázka sady `department` obsažené v třetí datové sadě [27].

Kapitola 4

Návrh systému pro tvorbu panoramat

Ze získaných znalostí a informací o možnostech tvorby panoramatických snímků, popsaných v kapitole 2, jsem vlastní systém pro skládání snímků do panoramat založil na přístupu popsaném jako základní. Tento přístup je založený na detekci klíčových bodů, vyhledání souhlasných klíčových bodů mezi snímky (korespondenční body), vypočítání matice homografie, nalezení nejlépe překrývajících se snímků, a jejich následném spojení.



Obrázek 4.1: Blokové schéma navrženého řešení.

Můj navržený systém pro skládání snímků do panoramat je zobrazen na obrázku 4.1 a jeho jednotlivé části budou založeny na tomto principu:

- Nejdříve si program načte vstupní snímky a podle způsobu zvolené projekce, je buď převede do cylindrických souřadnic (cylindrická projekce), sférických souřadnic (sférická projekce) a nebo je nechá v původním načteném tvaru (perspektivní projekce).
- Následně pomocí algoritmu SIFT, nebo SURF budou nalezeny význačné body v jednotlivých snímcích. V návrhu jsem zvolil realizaci obou variant, aby si uživatel mohl vybral podle svých preferencí (na rychlost, či kvalitu).

  Výhodou algoritmu SURF 2.2.3 je, že jeho výstupem je deskriptor, který je robustní vůči změně měřítka, rotaci, či změně jasu a kontrastu. Navíc je oproti metodě SIFT rychlejší a podle [1] více odolný proti šumu v obraze než SIFT.

  Naopak výhodou metody SIFT 2.2.2 je vyšší přesnost nalezených klíčových bodů (nachází asi o 6% více významných klíčových bodů) [1], ale je pomalejší. Její deskriptor je robustní vůči rotaci a změně měřítka, ale jenom částečně robustní proti změně jasu.
- Následně bude provedeno nalezení korespondencí mezi deskriptory klíčových bodů jednotlivých snímků.
- V dalším kroku bude nalezena co nejkvalitnější homografie pomocí algoritmu RANSAC pro jednotlivé dvojice snímků.
- Následně bude vyhodnoceno, které snímky opravdu sousedí s kterými, a tyto snímky budou složeny aplikováním daných matic homografie. V místě překryvu jednotlivých snímků budou artefakty odstraněny za pomoci algoritmu Graph Cuts.
- Nakonec bude obrázek možné oříznout a uložit.

Obrázek 4.2: Zjednodušené schéma použitých tříd.

Navržené řešení bude založeno na implementaci tří tříd (obrázek 4.2). Rozdělení funkčnosti těchto tříd je založeno na návrhovém vzoru **MVC** (Model - View - Controller). Třída `ImageStitch` bude zajišťovat metody pro práci nad vstupními daty, tj. hledání klíčových bodů, hledání korespondencí, nalezení homografie, spojení snímků a výřez nad snímky (**Model**). Třída `Computation` (**Controller**) bude zajišťovat řízení běhu výpočtů nad instancí třídy `ImageStitch`, a také bude komunikovat a reagovat na podněty od rodičovského nadřazeného objektu třídy `Stitcher` (**View**). Třída `Stitcher` bude zajišťovat realizaci uživatelského prostředí aplikace a reagovat na podněty z toho prostředí voláním metod instance třídy `Computation`.

V následujících podkapitolách bude popsán návrh metriky pro hodnocení kvality spojení jednotlivých snímků výsledného panoramatu (podkapitola 4.1) a návrh grafického uživatelského rozhraní aplikace (podkapitola 4.2). Implementace jednotlivých částí navrženého systému pro tvorbu panoramat je popsána v kapitole 5.

4.1 Návrh metriky pro hodnocení panoramat

Protože kvalitu spojení jednotlivých snímků navrženého systému by mělo být možné ohodnotit, navrhl jsem metriku pro takovéto hodnocení. Při její tvorbě jsem se inspiroval metrikou popsanou v kapitole 4 článku [7]. Zde popsaná metrika je založena na znalosti ideální homografie mezi snímky (je nazývána zlatá homografie) a na programově určené homografii ze snímků. Následně je náhodně určena pozice bodu v původním snímku a tato pozice je promítnuta vypočtenou i zlatou homografií a odchylka výsledných pozic je brána jako hodnocení. Tato odchylka se spočítá pro n pozic ve všech sousedících snímcích a vytvoří se průměr, což je výsledné hodnocení. Tuto metriku jsem nemohl využít, protože jsem neznal ideální homografii mezi snímky datových sad. Druhou metrikou, kterou jsem se inspiroval při tvorbě vlastní metody hodnocení, byla metrika popsaná v podkapitola 4.2 článku [32]. Tato metrika byla založena na výpočtu odchylek mezi korespondujícími klíčovými body. Jeden klíčový bod byl homografií promítnut do souřadného systému druhého a byla vypočtena odchylka jejich pozic pro všechny klíčové body.

Při tvorbě vlastní metriky pro hodnocení jsem chtěl zohlednit kvalitu spojení mezi dvojicemi snímků na základě pixelové (obrazové) shodnosti. Výstupem mojí metriky jsou tři hodnoty a to:

$$mean = \frac{1}{size(S)} \sum\nolimits_{\forall s \in S} \frac{1}{R} \sum_{j=(randX,randY)}^{R} \|s.A(j) - s.B(j * s.H_{AtoB})\| \tag{4.1}$$

$$max = max \left\{ \frac{1}{R} \sum_{j=(randX,randY)}^{R} \|s.A(j) - s.B(j * s.H_{AtoB})\| \middle| \forall s \in S \right\} \tag{4.2}$$

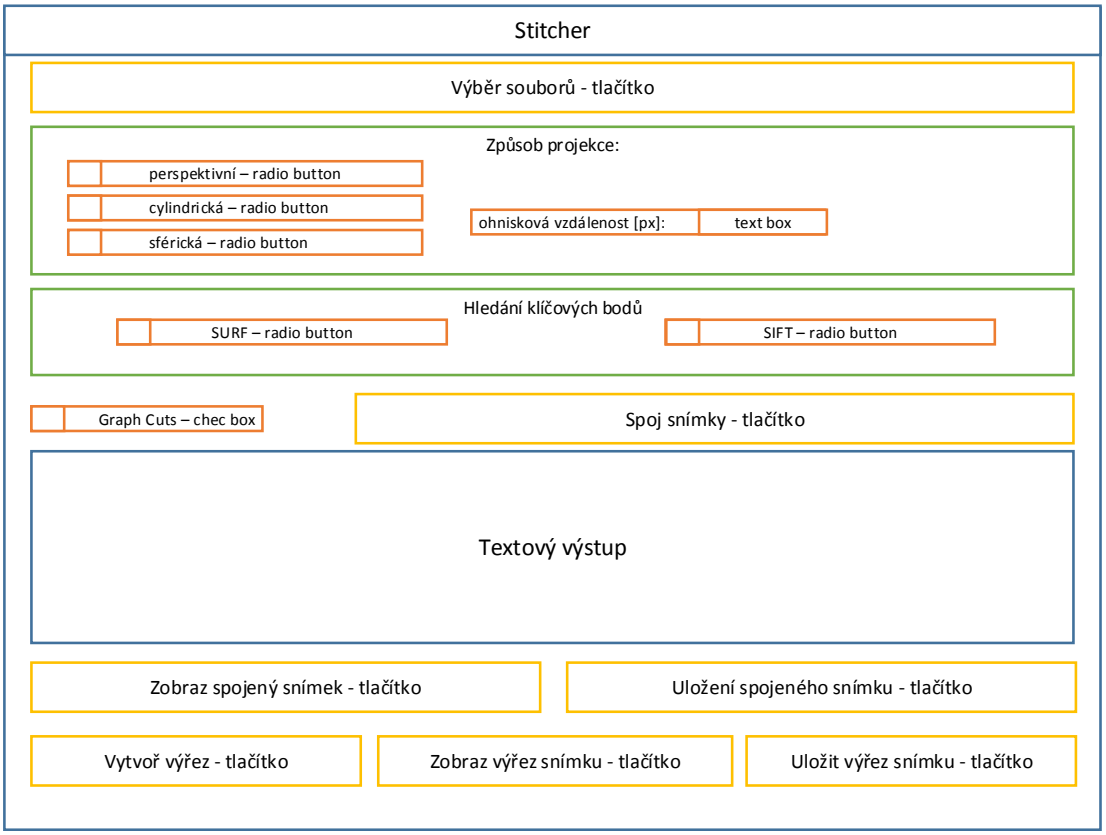
$$min = min \left\{ \frac{1}{R} \sum_{j=(randX,randY)}^{R} \|s.A(j) - s.B(j * s.H_{AtoB})\| \middle| \forall s \in S \right\}, \tag{4.3}$$

kde S zjednodušeně reprezentuje pole dvojic snímků patřících do výsledného panoramatu, kde položky $s \in S$ obsahují $s.A$, $s.B$ a $s.H_{AtoB}$, což jsou obrazové body snímků A, snímku B a homografie H_{AtoB} z A do B. K pixelové hodnotě snímku v s lze přistoupit takto $s.A(x, y)$. Položka R je daný počet náhodně vygenerovaných pozic bodů $(randX, randY)$, kde $(randX, randY) \in s.A$ a $(randX, randY) * s.H_{AtoB} \in s.B$. $\|$ reprezentuje euklidovskou vzdálenost, tj. vyjadřuje odchylku hodnoty pixelů (ideálně by měla být při ideálních snímcích a ideální homografii nulová).

Výstupy mojí metriky pro hodnocení panoramat tedy vyjadřují tyto vlastnosti. Hodnota *mean* vyjadřuje průměrnou odchylku pixelů ve všech spojeních výsledného panoramatu. Hodnota *max* vyjadřuje průměrnou odchylku pixelů nejhoršího spojení snímků výstupu a hodnota *min* nese průměrnou odchylku pixelů nejlepšího spojení snímků výsledného panoramatu. Čím se výstupní hodnota více blíží číslu nula, tím je kvalita výstupu lepší. Implementace této metriky je popsána v části 5.13. Použití vypočtených hodnot této metriky je ukázáno v kapitole 6.

4.2 Návrh grafického rozhraní aplikace

Pro snadné ovládání navržené aplikace jsem se pokusil navrhnout uživatelsky co nejpřívětivější GUI. Navržené grafické uživatelské rozhraní bude implementováno ve třídě `Stitcher` a jeho nákres je ukázán na obrázku 4.3.



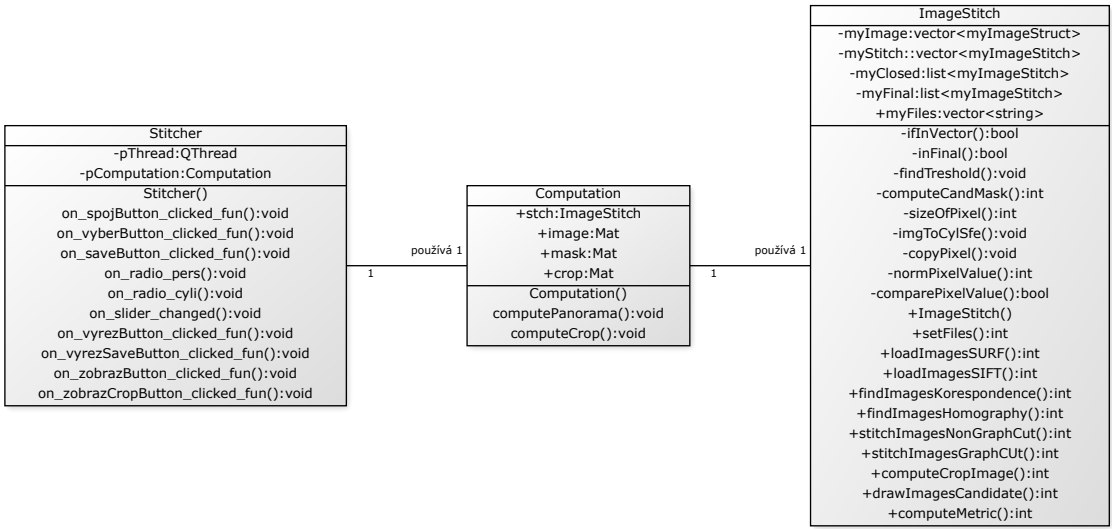
Obrázek 4.3: Návrh grafického uživatelského rozhraní.

Navržené uživatelské rozhraní se skládá z tlačítka pro výběr vstupních snímků, které budou spojovány, tlačítka pro zahájení spojení dle nastavených parametrů. Mezi parametry bude patřit výběr typu projekce pomocí radio přepínače. Pro projekce cylindrickou a sférickou bude možné nastavit ohniskovou vzdálenost objektivu v pixelech a parametry radiálního zkreslení objektivu. Dalším parametrem bude výběr metody pro hledání klíčových bodů SURF či SIFT radio přepínačem. Posledním vstupním parametrem bude zaškrtávací položka pro nastavení spojování snímků pomocí Graph Cuts. Dále bude grafické uživatelské rozhraní ještě obsahovat tlačítka pro uložení výsledného panoramatu, vytvoření maximálního výřezu z tohoto panoramatu a uložení tohoto výřezu. Implementace grafického rozhraní je popsána v sekci 5.14.

Kapitola 5

Implementace navrženého systému pro tvorbu panoramat

Aplikace vyvíjená pro účely této diplomové práce je realizována v jazyce C++ a nazývá se `Stitcher`. Tato aplikace je implementována dle návrhu systému pro tvorbu panoramat popsaného v kapitole 4. Hlavní roli v této aplikaci hraje implementace tříd `Stitcher`, `Computation` a `ImageStitch`.



Obrázek 5.1: Diagram tříd navrženého řešení.

Implementace metod pro práci s obrazovými daty ve třídě `ImageStitch` je založena na knihovně **OpenCV** (viz. podkapitola 5.1). Struktury, které jsou používány na uložení snímků a potřebných hodnot ve třídě `ImageStitch` jsou popsány v sekci 5.3. Ve třídě `ImageStitch` jsou implementovány metody pro její inicializaci, nastavení vstupních snímků a parametrů (sekce 5.4) a také realizace obrazových projekcí na vstupních snímcích (sekce 5.5). Dále jsou zde implementovány metody na vyhledání a uložení klíčových bodů a jejich deskriptorů (sekce 5.6), nalezení korespondencí klíčových bodů mezi snímky (sekce 5.7) a nalezení co nejlepší homografie nad těmito korespondencemi (sekce 5.8). Dle nalezených homografií je určeno, které snímky spolu opravdu sousedí a z těchto vazeb mezi snímky se určí, které snímky patří do výsledného panoramatu a jak se do něj promítají (sekce 5.9). Nako-

nec jsou snímky, u kterých jsme zjistili, že patří k sobě, spojeny do výsledného panoramatu prostým překrytím (sekce 5.10), nebo pomocí metody Graph Cuts (sekce 5.11). Potom co je výsledné panorama vytvořeno, je vypočítáno hodnocení, které udává jeho kvalitu (sekce 5.13). Z výsledného panoramatu je možné nakonec ještě vytvořit výřez (sekce 5.12).

Dále aplikace obsahuje třídu `Computation`, která implementuje řízení výpočtu nad metodami třídy `ImageStitch`, a třídu `Stitcher`, která pomocí knihovny **Qt** (viz. podkapitola 5.2) realizuje grafické uživatelské rozhraní (diagram tříd je znázorněn na obrázku 5.1). Pomocí grafického rozhraní uživatel volí parametry pro konstrukci výsledného panoramatického snímku. Implementace grafického uživatelského rozhraní je popsána v sekci 5.14.

V této kapitole budou postupně detailně popsány všechny implementované metody třídy `ImageStitch`, ve kterých jsou realizovány jednotlivé kroky z navrženého systému pro tvorbu panoramat. V následující kapitole 6 budou ukázány a zhodnoceny výsledky implementovaného řešení na deklarovaných datových sadách 3.

5.1 Knihovna OpenCV

OpenCV [24] je volně dostupná knihovna šířená pod BSD licencí. Knihovnu lze použít při implementaci v jazycích C, C++, Python a Java a podporuje provoz na Windows, Linux, Mac OS, iOS a Androidu. OpenCV se nejvíce zaměřuje na zpracování obrazu, ať již statických snímků, či videa. Při vývoji aplikace byla použita ve verzi 2.4.10.

Pro vyvíjenou aplikaci bude tato knihovna využita pro uschování načtených vstupních snímků, hledání klíčových bodů, nalezení ideální homografie a transformace snímků podle homografie.

5.2 Knihovna Qt

Knihovna Qt [25] je knihovna pro vytváření grafického uživatelského rozhraní pro aplikace. Její výhodou je multiplatformnost a ve verzi `Community` také její volná licence. Implementace grafického rozhraní probíhá v programovacím jazyce C++, ale klony této knihovny existují i pro jiné jazyky. Pro vyvíjenou aplikaci byla použita ve verzi 5.4.1.

5.3 Využívané datové typy a struktury ve třídě `ImageStitch`

Pro uložení snímků a vypočítaných hodnot ve třídě `ImageStitch` používám proměnné založené na dále popsaných ručně definovaných strukturách. Pro základní uložení vstupních snímků používám vektor `myImage` (viz. zjednodušené schéma principu zavedených proměnných na obrázku 5.2), který je založen na položkách struktury `myImageStruct` (zdrojový kód 5.1).

Struktura `myImageStruct` je záznamem pro informace a hodnoty jednoho snímku. V položce `name` je uloženo jméno vstupního snímku, `vectorID` obsahuje index daného snímku ve vektoru `myImage`. Položka `image` obsahuje vstupní snímek a v `mask` je maska tohoto snímku. Maska je důležitá v případě cylindrických a sférických snímků, kde převedené snímky nejsou obdélníkové, tudíž nepokrývají celou část položky `image`, a proto je v položce `mask` uloženo, které pixely jsou obsaženy ve snímku, a které jsou pouze okolí. Hodnoty `keypoints` a `descriptors` obsahují vypočtené klíčové body a jejich deskriptory pro daný snímek. Položka `candidate` je vektor založený na struktuře `myImageCandidate` (zdrojový kód 5.2) a obsahuje pro daný snímek vypočtené záznamy mezi ním a všemi ostatní snímky.

```
typedef struct{
    string name;
    int vectorID;
    Mat image;
    Mat mask;
    vector<Mat> descriptors;
    vector<KeyPoint> keypoints;
    vector<myImageCandidate> candidate;
} myImageStruct;
```

Zdrojový kód 5.1: Popis struktury `myImageStruct` pro uschování snímků.

```
typedef struct{
    int vectorID;
    vector<DMatch> matches;
    vector<DMatch> inliers;
    vector<DMatch> outliers;
    int inliersCount;
    int thresholdCount;
    vector<uchar> homographyMask;
    Mat homography;
} myImageCandidate;
```

Zdrojový kód 5.2: Popis struktury `myImageCandidate` kandidátních snímků.

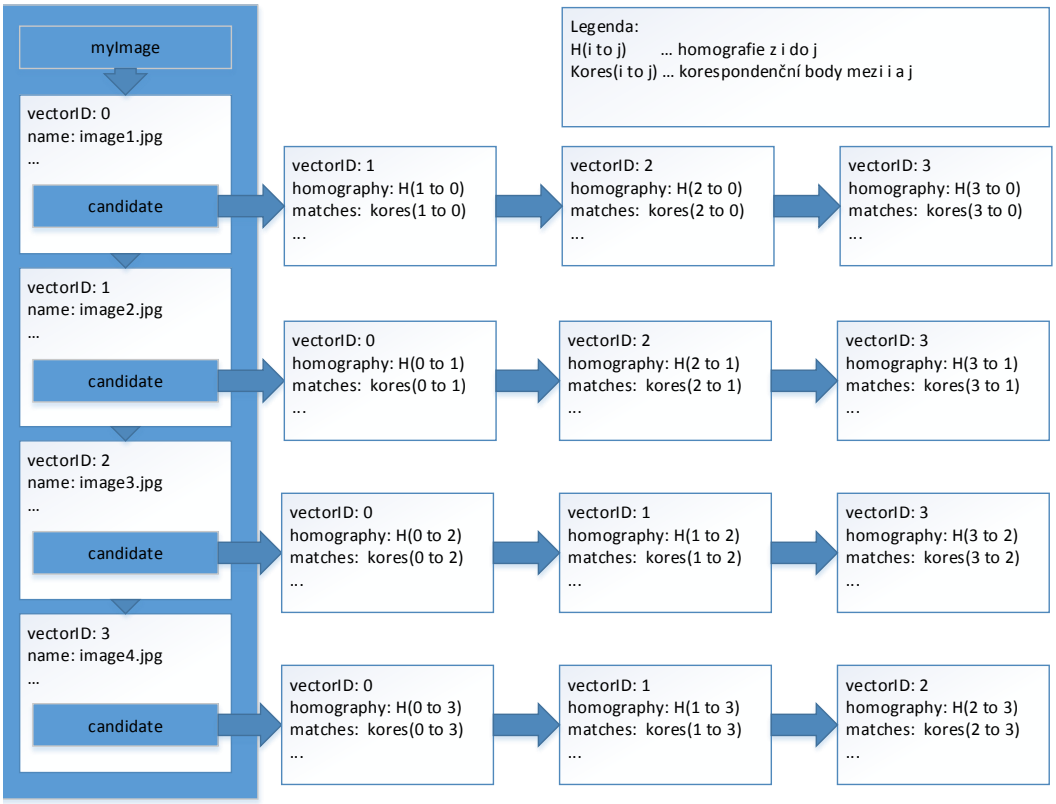
Struktura `myImageCandidate` obsahuje hodnoty vypočtené mezi *hlavním* snímkem, uloženým v nadřazené `myImageStruct`, a ostatními snímky. Položky této struktury jsou následující `vectorID` je index `kandidátního` snímku ve vektoru snímků (`myImage`). Vektor `matches` pro uložení vypočtených kandidátních korespondenčních bodů mezi *hlavním* a *kandidátním* snímkem. Položka `homography` na uschování vypočtené matice homografie a `homographyMask` je vektor obsahující informace o tom, které z korespondenčních bodů jsou pro danou homografii považovány za *inliers*. Hodnota `inliersCount` obsahuje počet *inliers* bodů a `thresholdCount` obsahuje vypočtený počet *inliers* a *outliers* bodů v překrývající se oblasti daných snímků.

Poslední používaná struktura `myImageStitch` (zdrojový kód 5.3) se používá v seznamech při výběru ideálního spojení mezi snímky (podkapitola 5.9), které jsou na základě předchozích zjištění považovány za reálně sousedící. Položky `vectorIDmain` a `vectorIDcand` jsou indexy odkazující na reálné místo uložení daných snímků ve `myImage`. Položky `homography`, `homographyOld` a `homographyMain` obsahují různě upravované matice homografie pro dané odkazované snímky, ostatní položky této struktury jsou pouze pomocné.

Dále třída `ImageStitch` při svém běhu využívá veřejnou proměnnou `myFiles`, což je vektor řetězců obsahující cesty ke vstupním snímkům pro zpracování. A také tři privátní proměnné, a to již dříve zmíněnou proměnnou `myImage`, která obsahuje data k jednotlivým snímkům. Pomocnou proměnnou `myClosed` a také proměnnou `myFinal`, což je list položek struktury `myImageStitch` a jsou do něj uloženy záznamy sloužící pro výsledné spojení snímků.

```
typedef struct{
    int vectorIDmain;
    int vectorIDcand;
    double inliersRank;
    bool switched;
    Mat homography;
    Mat homographyOld;
    Mat homographyMain;
} myImageStitch;
```

Zdrojový kód 5.3: Popis struktury ideálních kandidátů.



Obrázek 5.2: Zjednodušené schéma uložení vstupních snímků, korespondenčních bodů, matic homografií v proměnné `myImage`. Demonstrováno na čtyřech vstupních snímcích.

5.4 Inicializace aplikace

Při spuštění aplikace `Stitcher` se vytvoří instance dříve zmíněné třídy `ImageStitch`. Následně se pomocí grafického rozhraní zadají vstupní snímky, a cesty k těmto snímkům se pomocí metody *setFiles* uloží do veřejné proměnné `myFiles`.

Dále se pomocí uživatelského rozhraní aplikace navolí parametry určující druh vytvářeného panoramatu podle způsobu projekce vstupních snímků (perspektivní, cylindrická či sférická). Při volbě cylindrické a sférické projekce je nutné zadat ohniskovou vzdálenost

objektivu v pixelech a volitelně je možné dodat parametry k_1 a k_2 pro korekci radiálního zkreslení objektivu. Uživatel dále navolí, zda ve spojovaných snímcích budou klíčové body vyhledány pomocí metody SURF nebo SIFT. Jako poslední možnou volbu, důležitou pro inicializaci vstupních snímků, je volitelné zmenšení vstupních snímků před samotným zpracováním. To může být výhodné v situaci, kdy mají vstupní snímky vysoké rozlišení, ale výstup není v tak vysokém rozlišení požadován, a také pro případ urychlení výpočtu.

5.5 Obrazové projekce vstupních snímků

Potom, co uživatel zadal potřebné vstupní parametry je volána metoda třídy `ImageStitch`, nazývaná *loadImagesSURF*, či *loadImagesSIFT* podle zvoleného principu pro hledání klíčových bodů. Samotné hledání klíčových bodů bude popsáno v následujících podkapitolách. Předtím, než může proběhnout toto hledání, je nutné zpracovat vstupní snímky.

Z toho důvodu se v dříve zmíněných metodách postupně v cyklu projdou všechny vstupní snímky, které jsou zadány ve formě absolutních cest k jejich fyzickému umístění. Snímky jsou jednotlivě pomocí funkce `cv::imread` uloženy do proměnné a je zkontrolováno, zda se načtení podařilo v pořádku. Následně, pokud je požadovaná cylindrická nebo sférická projekce, je do daných souřadnic snímek převeden pomocí implementované metody *imgToCylSfe*. Tato metoda jako vstup bere zadaný vstupní snímek, ohniskovou vzdálenost, parametry radiálního zkreslení, typ projekce a jejím výstupem je snímek převedený v požadovaných souřadnicích a jeho maska. Při prostém převádění do cylindrických, nebo sférických souřadnic zůstává velikost potřebné matice na uložení snímku shodná s velikostí vstupní matice, ale při nastavení korekce radiálního zkreslení se výstupní snímek v některých případech zvětšuje (prohýbají se hrany), proto nastavuji velikost výstupního snímku na 1.2 násobek vstupní velikosti.

Obrázek 5.3: Ukázka převodu snímku `yosemite1` (datová sada na obrázku 3.1) do sférických souřadnic s ohniskovou vzdáleností 765px a koeficienty radiálního zkreslení $k_1 = -0.21$ a $k_2 = 0.26$. Z levé strany máme vstupní snímek, snímek ve sférických souřadnicích, masku snímku a snímek ve sférických souřadnicích s rozšířenými okraji.

Metoda *imgToCylSfe* je implementována následovně. Pro převod do cylindrických/-sférických souřadnic se používá *inverzní* cylindrická/sférická projekce. Z toho vyplývá, že pro každý pixel výsledného snímku se vypočítá pozice odpovídajícího pixelu ve vstupním snímku. Výpočet pozice je realizován pomocí vzorců uvedených v podkapitolách 2.1.4 a 2.1.5. Výsledná pozice (x,y) pixelu ve vstupním snímku skoro vždy nevychází celočíselně, ale desetinně, tudíž někam mezi stávající pixely. Z toho důvodu moje implementace hodnotu výsledného pixelu určuje pomocí bilineární interpolace (s váhami odpovídajícími vzdálenostem, je hodnota výsledného pixelu vypočtena z hodnot čtyř nejbližších pixelů) nad desetinou hodnotou pozice. Výsledný převedený snímek ve sférických souřadnicích je možné si prohléd-

nout na druhém snímku v obrázku 5.3 a masku (černé pixely vyjadřují okolí, které nepatří do převedeného snímku, bílé označují platné pixely výsledného snímku) tohoto převedeného snímku na třetím snímku obrázku 5.3.

Z důvodu, že při následné manipulaci se snímky transformují podle matic homografie, a černé okolí pozměňovalo barvu krajových pixelů při interpolaci do daných tvarů snímků, ještě snímky po převedení do cylindrických/sférických souřadnic upravím do podoby vyobrazené na čtvrtém snímku obrázku 5.3. A to provedu tak, že rozšířím hodnotu pixelů ležících na hranici validní části obrazových pixelů (oblast hranice černé a bílé části v masce) do jejich okolí tzn. černé části masky. Tudíž, když se bude daný snímek interpolovat do nějaké specifické podoby, nebudou se při interpolaci hodnot krajních pixelů brát z hraničního okolí černé pixely, ale pixely podobné hodnoty jako je v okolí, a proto nebude výsledná hodnota krajních pixelů nijak zvlášť zkreslena.

Výsledné převedené snímky a jejich masky jsou navráceny do volající funkce. Následně, pokud bylo zadáno, aby byli snímky před zpracováním na panoramata zmenšeny (provádí se i pokud byla zvolena perspektivní projekce, tzn. snímky nebyly převáděny do jiné obrazové projekce), je snímek i jeho maska zmenšena pomocí funkce `cv::resize` dle požadovaného "scalu". Potom, co jsou nad daným snímkem provedeny dříve zmíněné operace, přistupuje se k vyhledání klíčových bodů. Implementované způsoby hledání klíčových bodů jsou popsány v následující podkapitole 5.6.

5.6 Hledání klíčových bodů ve snímcích

V implementované třídě `ImageStitch` je nalezení významných bodů SURF implementováno v metodě *loadImagesSurf* a SIFT v metodě *loadImagesSift*. Tyto metody byli již dříve zmíněny v podkapitole 5.5, v těchto metodách se nejdříve realizují úpravy vstupních snímků. Potom, co jsou již snímky převedeny do požadovaných formátů, je nad nimi provedeno hledání významných bodů podle postupu popsaného dále v této kapitole. Nalezené významné body a jejich deskriptory jsou uloženy k danému snímku do vektoru vstupních snímků `myImage` viz. podkapitola 5.3 ukázka kódu 5.1. Pro hledání klíčových bodů jsem využil funkce **OpenCV** obsažené v modulu `nonfree/features2d`.

Pro metodu SURF je třída pro nalezení klíčových bodů nazvána `SurfFeatureDetector` a třída pro výpočet deskriptoru těchto klíčových bodů se nazývá `SurfDescriptorExtractor`. Ukázka použití funkcionality obsažené v těchto třídách viz. zdrojový kód 5.4. Jak je vidět z ukázky, je pro použití nutné instance těchto tříd v konstruktoru inicializovat danými parametry. Pro metodu SURF je pořadí parametrů následující `(hessianThreshold, nOctaves, nOctaveLayers, extended, upright)` a jejich význam je:

- `hessianThreshold` – práh pro detekci výskytu klíčového bodu. Čím je práh vyšší tím méně je nalezeno klíčových bodů. Pro svoji aplikaci jsem zvolil hodnotu 300, která je v doporučeném rozmezí 300-500 z manuálu OpenCV.
- `nOctaves` – počet oktáv, to je počet různých měřítek ve Scale space 2.2.3.
- `nOctaveLayers` – počet úrovní rozmazání dané oktávy ve Scale space 2.2.3.
- `extended` – boolovská hodnota, která nastavuje zda se vypočte rozšířený deskriptor tzn. pro `true` 128 hodnotový a pro `false` pouze 64 hodnotový.
- `upright` – boolovská hodnota určující zda se pro každý klíčový bod vypočte jeho orientace. `False` označuje výpočet orientace a `true` naopak nepočítání orientace klíčového

bodu. Výpočet orientace klíčového bodu zajišťuje jeho rotační invariantnost a proto pro různě natáčené snímky do panoramatu je znalost orientace výhodná.

Použité hodnoty pro `nOctaves` a `nOctaveLayers` jsem převzal jako doporučené z manuálu OpenCV (jsou ukázány ve zdrojovém kódu 5.4). Samotné vyhledání klíčových bodů se provádí zavoláním metody `detect` nad vstupním snímkem, kde nalezené klíčové body jsou uloženy do vektoru položek typu `KeyPoint`, a navíc jde hledání klíčových bodů omezit pouze na vybrané oblasti zadáním masky vstupního snímku, čehož jsem při implementaci využil, protože maska určuje oblasti, kde se vyskytují skutečná obrazová data a nikoli okolí (jak bylo uvedeno v podkapitole 5.5 o obrazových projekcích, kde při cylindrické/sférické projekci platné pixely nepokrývají celou oblast snímku). Nakonec je nad vstupním snímkem a nalezenými klíčovými body zavolána metoda `compute`, která vypočte deskriptory daných významných bodů. Ukázku nalezených klíčových bodů SURF je možné si prohlédnout v levé části obrázku 5.4.

```
SurfFeatureDetector surfDetector(300, 4, 2, true, false);
SurfDescriptorExtractor surfExtractor;
surfDetector.detect(image, keypoints, mask);
surfExtractor.compute(image, keypoints, descriptor);
```

Zdrojový kód 5.4: Výpočet klíčových bodů metodou SURF pomocí OpenCV.

Obrázek 5.4: Ukázka nalezených klíčových bodů ve snímku `yosemite1` (sada na obrázku 3.1) v perspektivní projekci. V levém snímku metodou SURF a v pravém metodou SIFT.

Hledání významných bodů a výpočet jejich deskriptorů pro metodu SIFT je implementován ve třídách `SiftFeatureDetector` a `SiftDescriptorExtractor`. Instance tříd je opět nutné při konstrukci inicializovat parametry `(nfeatures, nOctaveLayers, contrastThreshold, edgeThreshold, sigma)` v tomto významu:

- `nfeatures` – počet navrácených významných bodů. Body jsou ohodnoceny pomocí skóre a navráceno je pouze `n` nejlepších. Zadáním čísla 0 jsem navráceny všechny nalezené bez omezení počtu.
- `nOctaveLayers` – počet úrovní rozmazání dané oktávy ve Scale space 2.2.2.

- `contrastThreshold` – kontrastní práh pro filtrováni nízko-kontrastních oblastí. Čím je práh nastaven na vyšší hodnotu tím je nalezeno méně významných bodů.
- `edgeThreshold` – práh používaných k odfiltrování hranových významných bodů. Čím je práh nastaven na vyšší hodnotu tím méně, hranových významných bodů je odfiltrováno.
- `sigma` – velikost odchylky Gaussovského filtru použitá na první oktávě.

Parametry použité v implementaci byly převzaty ze základního nastavení SIFT funkce v OpenCV (jsou ukázány ve zdrojovém kódu 5.5). Ukázka použití výpočtu SIFT významných bodů je ve zdrojovém kódu 5.5. Nalezené klíčové body metodou SIFT je možné si prohlédnout v pravé části obrázku 5.4.

```
SiftFeatureDetector siftDetector(0, 3, 0.04, 10, 1.6);
SiftDescriptorExtractor siftExtractor;
siftDetector.detect(image, keypoints, mask);
siftExtractor.compute(image, keypoints, descriptor);
```

Zdrojový kód 5.5: Výpočet klíčových bodů metodou SIFT pomocí OpenCV.

5.7 Hledání korespondencí klíčových bodů

Máme-li nalezeny klíčové body a jejich deskriptory pro všechny vstupní snímky, pak následně je nutné mezi každými dvěma různými vstupními snímky porovnat klíčové body a nejpodobnější prohlásit za korespondenční body. Pro implementaci této funkčnosti je možné z knihovny OpenCV použít dva již implementované porovnávače deskriptorů a to `BFMatcher` nebo `FlannBasedMatcher`.

`BFMatcher` je založen na prostém brute-force porovnání každého deskriptoru s každým, což je pro mnoho detekovaných klíčových bodů silně výpočetně náročné pokud porovnáváme deskriptory všech snímků mezi sebou.

Z důvodu velké výpočetní náročnosti `BFMatcheru` jsem pro použití v implementaci systému na tvorbu panoramat zvolil `FlannBasedMatcher`. `FlannBasedMatcher` (Fast Library for Approximate Nearest Neighbors) je knihovna realizovaná podle [22]. Tato metoda podporuje velice rychlé vyhledání dvojic přibližně nejpodobnějších deskriptorů. Funguje tak, že se *matcher* nejdříve natrénuje na deskriptory významných bodů jednoho snímku. Při tomto natrénování si pro dané deskriptory vytvoří vyhledávací strukturu. Je možné zvolit různé druhy vyhledávacích struktur. Pro účely této implementace jsem zvolil základní vyhledávací strukturu založenou na kd-stromech. V ukázce z implementovaného kódu 5.6 je možné vidět vytvoření `FlannBasedMatcher` s vyhledávací strukturou `KDTreeIndexParams(4)`, kde parametr udává, že se vytvoří čtyři paralelní kd-stromy. Parametr `SearchParams(32)` udává kolikrát je ve kd-stromě/kd-stromech rekurzivně hledána nejlepší shoda, jako výsledek je určena ta s nejlepším hodnocením.

V ukázce kódu 5.6 je vidět vytvoření *matcheru*, následné přidání deskriptorů jednoho snímku *image2*, pomocí metody `add` a natrénování na tyto deskriptory, pomocí metody `train`. Po těchto krocích je možné voláním metody `match` nad deskriptory jiného snímku *image1* vypočíst vektor korespondenčních významných bodů `matches`. Vektor `matches` obsahuje položky struktury `DMatch`, a těchto položek obsahuje stejný počet, jako obsahuje

```
FlannBasedMatcher matcher(new cv::flann::KDTreeIndexParams(4), new cv::
flann::SearchParams(32));
matcher.add(image2.descriptors);
matcher.train();
matcher.match(image1.descriptors[0], matches);
matcher.clear();
```

Zdrojový kód 5.6: Nalezení kandidátů na korespondenční body pomocí OpenCV.

Obrázek 5.5: Ukázka nalezených korespondenčních klíčových bodů ve snímcích `yosemite1` a `yosemite2` (sada na obrázku 3.1) v perspektivní projekci.

snímek *image1* klíčových bodů, tj. pro každý klíčový bod deskriptoru vloženého do `match` je nalezen co nejpodobnější klíčový bod z natrénovaného deskriptoru. Položky `DMatch` obsažené ve výstupním vektoru obsahují tři důležité hodnoty: index klíčového bodu ze snímku *image1* `queryIdx`, `trainIdx` hodnota indexu klíčového bodu ze snímku *image2* a `distance`, což je míra podobnosti klíčových bodů. Potom je také možné voláním metody `clear` smazat natrénovanou vyhledávací strukturu, a případně použitím `add` a `train` natrénovat jinou.

Tímto způsobem popsaným v předcházejícím odstavci, je implementována metoda *findImagesKorespondence* ve třídě `ImageStitch`. Pro každý snímek (query) ve vektoru `myImage` jsou nalezeni kandidáti pro významné body ve všech ostatních snímcích (train). V záznamu každého snímku (viz. zdrojový kód 5.1 nebo schéma uložení obrázek 5.2) je vektor `candicate`, do kterého se uloží pro každý jiný snímek, než kterému tento vektor patří, záznam obsahující index kandidátního snímku `vectorID`, a vypočtené korespondenční body `matches`. V `matches` jsou klíčové body nadřazeného snímku indexovány jako `queryIdx` a klíčové body kandidátního snímku `trainIdx`. Ukázku nalezených korespondenčních bodů je možné si prohlédnout na obrázku 5.5. Jak je na obrázku zřejmé z množství korespondenční bodů zatím nejsme schopni homografii příliš poznat.

5.8 Nalezení homografie mezi snímky

Potom, co jsme pro každou dvojici snímků vypočítali korespondenční klíčové body, pak pomocí implementované metody *findImagesHomography* nalezneme pro tyto snímky algoritmem RANSAC matici homografie. Na samotné hledání homografie jsem použil funkci

z OpenCV, která se jmenuje `findHomography(srcPoints, dstPoints, method, ransacReprojThreshold, mask)`.

```
homography = findHomography(srcPoints, dstPoints, CV_RANSAC, 3.0, mask);
```

Zdrojový kód 5.7: Ukázka nalezení homografie na korespondenčních bodech pomocí OpenCV.

Tato funkce (ukázka kódu 5.7) bere jako první dva parametry vektory bodů, kdy vektor `srcPoints` obsahuje pozice významných bodů kandidátního snímku (train) a vektor `dstPoints` obsahuje body nadřazeného snímku ke kandidátovi (query). V obou vektorech jsou body dány tak, že body na stejných indexech jsou korespondující body. Funkce `findHomography` se potom snaží vytvořit homografii tak, aby po aplikaci matice homografie na pozice bodů ve vektoru `srcPoints`, souhlasily výsledné pozice s pozicemi bodů ve vektoru `dstPoints` na stejných indexech. Jako další (třetí) parametr funkce `findHomography` přijímá nastavení metody hledání klíčových bodů. V návrhu jsem specifikoval, že homografie bude v implementaci hledána pomocí metody RANSAC, proto je jako třetí parametr zadáno CV_RANSAC. RANSAC implementovaný ve funkci `findHomography` pracuje na principu popsaném v podkapitole 2.7, tj. hledá matici homografie takovou, aby chyba zpětné projekce (back-projection error) byla co nejmenší. Čtvrtým parametrem je `ransacReprojThreshold` – tato hodnota vyjadřuje jak daleko se může bod ze `srcPoints` po aplikaci homografie nacházet od `dstPoints` (viz. vzorec 5.1), aby tyto body byly označeny, že náleží do inliers (body náležící hledané homografii), jinak jsou označeny jako outliers.

$$d(dstPoints_i, H_{SrcToDst} * srcPoints_i) > ransacReprojThreshold, \tag{5.1}$$

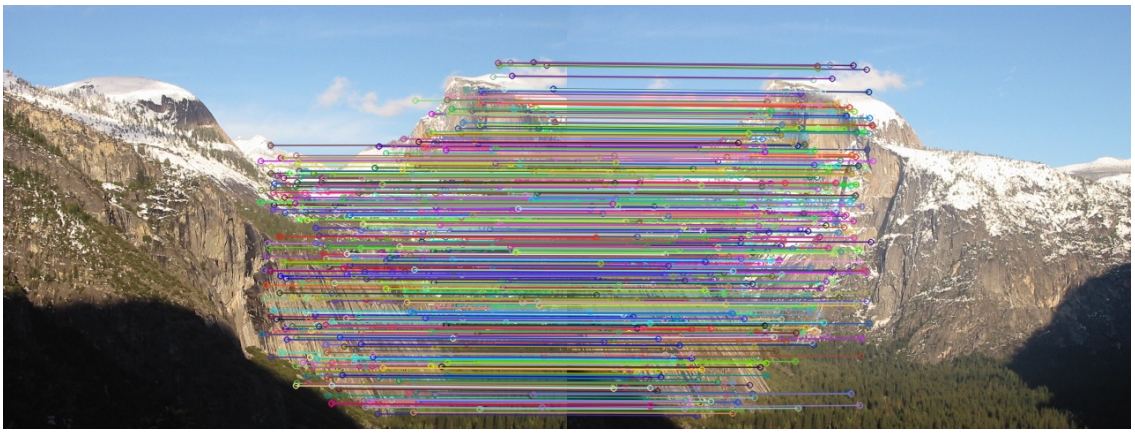
kde d je Euklidovská vzdálenost a $H_{SrcToDst}$ je homografie z prostoru bodů $srcPoints$ do $dstPoints$.

Z toho vyplývá, že parametrem `ransacReprojThreshold`, lze korigovat počty inliers/outliers korespondencí. Posledním zadávaným parametrem funkce `findHomography` je maska, přes kterou nám funkce vrátí informaci o tom, které korespondence náleží do homografie (inliers), a které ne (outliers). Maska je prostý vektor o stejné velikosti jako vektory `srcPoints` a `dstPoints`, a pokud ve vektoru masky na indexu i je hodnota 1, pak korespondence bodů v `srcPoints` a `dstPoints` na indexu i jsou inliers, jinak outliers.

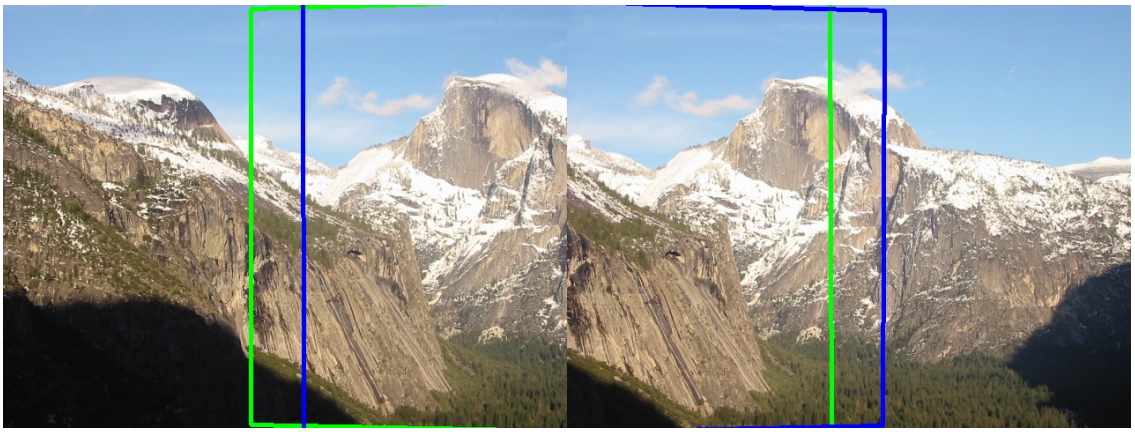
Vypočtenou matici homografie funkce `findHomography` předává jako návratovou hodnotu (ukázka kódu 5.7). Ukázka korespondenčních bodů z obrázku 5.5 označených jako inliers je na obrázku 5.6 a jako outliers na obrázku 5.8. Na obrázku 5.7 je ukázáno promítnutí jednoho snímku do souřadného systému druhého snímku a naopak, dle vypočítané matice homografie.

Takto popsaným postupem je pro všechny položky `myImage` a jejich odpovídající položky v `candidate` vypočtena homografie a uložena do kandidátské položky (znázornění obrázek 5.2). Také jsou uloženy inliers a outliers korespondence. Dále je pro vypočtenou homografii nad danými snímky zavolána implementovaná funkce *findTreshold*, která vypočte počet korespondencí (inliers i outliers) v překrývající se oblasti snímků. Překrývající se oblast je určena pomocí vypočtené homografie. Znalost počtu korespondencí v překrývající se oblasti je důležitá pro určení, zda jsou dané dva snímky opravdu sousedící, mají překryv a jdou podle homografie opravdu validně spojit (kapitola 3.2 v [6]). Tato vypočítaná hodnota je

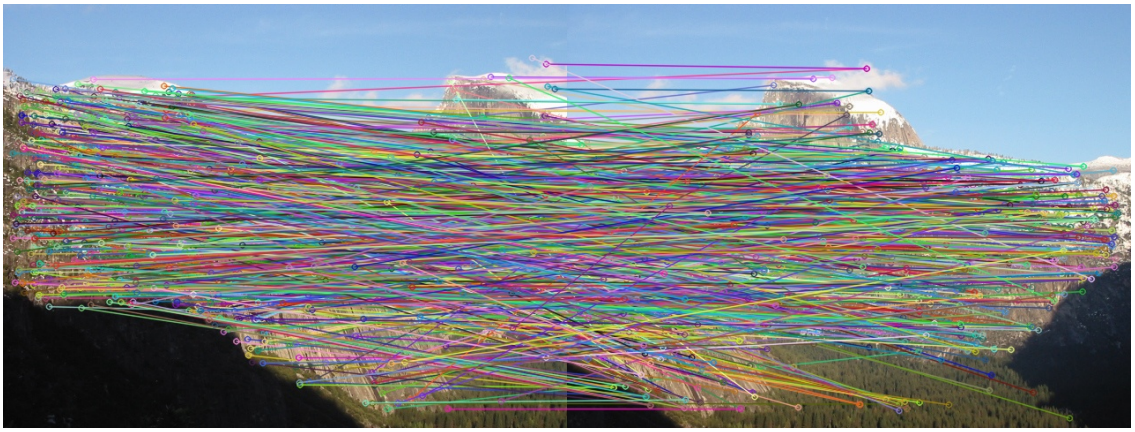
uložena ke kandidátnímu snímku do proměnné `thresholdCount`. Bližší vysvětlení jejího použití bude popsáno v příští podkapitole 5.9.



Obrázek 5.6: Ukázka nalezených inliers korespondenčních bodů ve snímcích `yosemite1` a `yosemite2` (sada na obrázku 3.1) v perspektivní projekci.



Obrázek 5.7: Ukázka pozice umístění pravého snímku `yosemite2` do souřadného systému levého snímku (zelený obdélník) a pozice levého snímku `yosemite1` do pravého snímku (modrý obdélník) podle vypočítané matice homografie a její inverze. Hodnocení kvality této homografie `inliersRank` je 12,74. Ukázka je znázorněna na snímcích ze sady na obrázku 3.1 v perspektivní projekci.



Obrázek 5.8: Ukázka nalezených outliers korespondenčních bodů ve snímcích `yosemite1` a `yosemite2` (sada na obrázku 3.1) v perspektivní projekci.

5.9 Nalezení ideálního spojení snímků

Určení toho, které snímky se v reálné scéně opravdu překrývají tzn. spolu sousedí, je docela obtížný úkol. Implementovaná metoda *findImagesIdealStitch* ve třídě *ImageStitch* nejprve určuje, které snímky spolu opravdu sousedí. Určení tohoto je založeno na zkoumání kvality vypočtené homografie mezi snímky. Následně se nad dvojicemi snímků snaží určit jejich co nejlepší propojení a spojováním homografií převést všechny snímky do souřadného systému jednoho hlavního snímku.

Funkce nejprve vytvoří pomocný vektor `myNeigh` položek struktury `myImageStitch`. Do tohoto vektoru umístí ty spojení dvojic snímků, které na základě vypočtených hodnot určí jako kvalitní. Při tvorbě způsobu, jakým poměřuji kvalitu homografií, jsem se inspiroval v kapitole 3.2 v [6]. Princip tam popsaný je založen na tom, že pokud je vzorec 5.2 pro danou homografii splněn, je toto spojení považováno za kvalitní.

$$inliersCount > \alpha + \beta * thresholdCount, \tag{5.2}$$

kde $inliersCount$ je počet inliers korespondencí, $thresholdCount$ počet korespondencí v překryvu snímků (více poslední odstavec podkapitoly 5.8), α a β jsou konstanty. Z důvodu jiné implementace než v [6], jsem musel experimentálně zjistit vlastní hodnoty konstant α a β po testech na datových sadách jsem jako vyhovující uznal hodnoty $\alpha = 8$ a $\beta = 0.05$.

Do vektoru `myNeigh` se tedy vloží pouze ty spojení, které splní podmínku 5.2. Položka `myNeigh` následně obsahuje index snímku z `myImage` (hodnota `vectorIDmain`), do jehož souřadného systému daná hodnocená homografie převádí snímek z jeho `candidate` vektoru (uložen je i index kandidátního snímku v `myImage` do `vectorIDcand`). Následně je k tomuto záznamu uložena příslušná matice homografie a také hodnota `inliersRank`, což je číselná hodnota vyjadřující kvalitu spojení, vypočtená následovně:

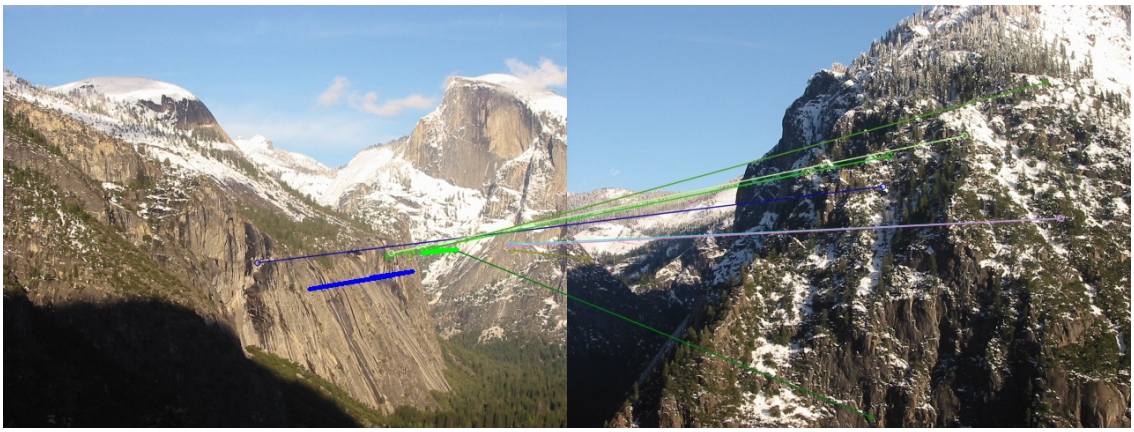
$$inliersRank = \frac{inliersCount}{\alpha + \beta * thresholdCount} \tag{5.3}$$

Ze způsobu výpočtu `inliersRank` vyplývá, že pokud by nebyla splněna podmínka 5.2 bylo by `inliersRank` menší nebo rovno 1. V následující tabulce 5.1 jsou zobrazeny hodnoty `inliersRank` pro všechny spojení snímků ze sady `yosemite` (obrázek 3.1).

inliersRank	**dest**			
src	`yosemite1`	`yosemite2`	`yosemite3`	`yosemite4`
`yosemite1`		12.99	0.27	0.08
`yosemite2`	12.74		12.01	0.10
`yosemite3`	0.10	11.39		8.16
`yosemite4`	0.64	0.11	11.91	

Tabulka 5.1: Tabulka `inliersRank` obsahuje hodnocení homografií, které převádějí **src** snímky do souřadného systému **dest** snímků.

Velmi dobře hodnocenou homografii můžeme vidět na obrázku 5.7, tato homografie z `yosemite2` do `yosemite1` má `inliersRank` 12,74, což nám říká, že snímky spolu opravdu sousedí a homografie je kvalitní. Naopak nekvalitní homografii s hodnocením 0,64 je možné vidět na obrázku 5.9, z tohoto hodnocení vyplývá, že snímky se nepřekrývají.



Obrázek 5.9: Ukázka nalezených inliers a promítnutých pozicí jako v obrázku 5.7, ale pro nepřekrývající se snímky `yosemite1` a `yosemite4` (datová sada na obrázku 3.1) v perspektivní projekci. Hodnocení homografie je 0,64, což říká, že toto spojení není vhodné do výsledného panoramatu. To je také z obrázku patrné.

Nyní tedy máme ve vektoru `myNeigh` všechny homografie mezi dvojicemi snímků, které považujeme za kvalitní. Tento vektor seřadíme pomocí hodnoty `inliersRank` sestupně. Často se stává, že ve vektoru `myNeigh` existují dva záznamy pro stejné spojení a to takové, že jeden má např. $vectorIDmain = 3$ a $vectorIDcand = 5$ (homografie ze snímku id 5 do id 3) a druhý má $vectorIDmain = 5$ a $vectorIDcand = 3$ (homografie ze snímku id 3 do id 5), takového záznamy označíme jako neunikátní. Abychom toto eliminovali, vytvoříme seznam `myClosed`, do kterého umístíme všechny unikátní spojení a dále tam vložíme z každé dvojice neunikátních spojení to s lepším hodnocením. Záznamy do seznamu `myClosed` vkládáme tak, aby nedošlo k porušení seřazenosti záznamů. Ukázka záznamů v seznamu `myClosed` je v tabulce 5.2.

Pokud by seznam `myClosed` byl prázdný, znamená to, že nebylo možné nalézt žádné kvalitní spojení mezi dvojicemi snímků, pak je oznámeno, že nebylo možné nalézt spojení mezi snímky. Nyní, pokud máme neprázdný seznam kvalitních spojení mezi snímky, musíme z těchto záznamů vytvořit pseudo-graf (obrázek 5.10) spojení snímků, kde vybereme

myClosed			
vectorIDmain	vectorIDcand	inliersRank	homography
yosemite2	yosemite1	12.99	homografie z 1 do 2
yosemite3	yosemite2	12.01	homografie z 2 do 3
yosemite3	yosemite4	11.91	homografie z 4 do 3

Tabulka 5.2: Ukázka seznamu `myClosed` pro snímky sady `yosemite`(datová sada na obrázku 3.1). Hodnocení je shodné s tabulkou 5.1. Pro lepší přehlednost jsou v tabulce místo prostých ID uvedeny názvy snímků.

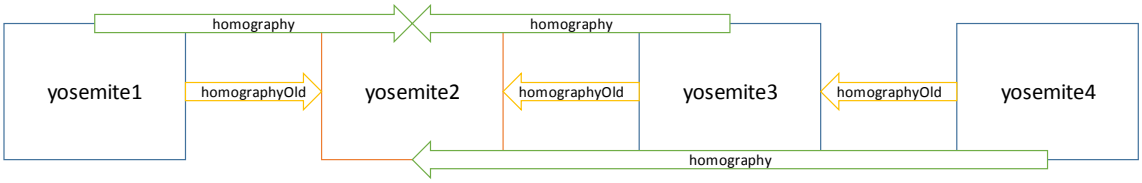
jeden snímek jako **hlavní** a pomocí skládání nalezených homografií (násobení matic) se snažíme vytvořit pro každý ostatní snímek homografii převádějící daný snímek do souřadného systému hlavního snímku. Sestavování tohoto grafu probíhá do nového seznamu nazvaného `myFinal` (viz. tabulka 5.3), tento seznam bude ve výsledku obsahovat posloupnost záznamů obsahujících identifikátor snímku a jeho homografii pro převod do systému hlavního snímku.

Seznam `myFinal` je strukturován následujícím způsobem. Jako první položka je přímo vložen záznam ze seznamu `myClosed` a další položky jsou přidávány postupem popsaným níže. První položka seznamu `myFinal` je důležitá protože snímek v ní identifikovaný jako `vectorIDmain` je považován pro další zpracování jako **hlavní** snímek. Záznam v seznamu `myFinal` obsahuje tyto položky:

- `vectorIDmain` – identifikátor hlavního (`vectorIDmain` v prvním záznamu) a nebo návazného snímku
- `vectorIDcand` – identifikátor snímku převáděného do prostoru hlavního snímku
- `inliersRank` – hodnocení kvality homografie
- `homography` – homografie z prostoru `vectorIDcand` daného záznamu do prostoru hlavního snímku (`vectorIDmain` v prvním záznamu)
- `homographyOld` – homografie z prostoru `vectorIDcand` do `vectorIDmain` daného záznamu
- `homographyMain` – homografie z prostoru `vectorIDmain` daného záznamu do prostoru hlavního snímku (`vectorIDmain` v prvním záznamu)

myFinal					
vectorIDmain	vectorIDcand	inliersRank	homography	homographyOld	homographyMain
yosemite2	yosemite1	12.99	h. z 1 do 2	h. z 1 do 2	h. z 2 do 2
yosemite2	yosemite3	12.01	h. z 3 do 2	h. z 3 do 2	h. z 2 do 2
yosemite3	yosemite4	11.91	h. z 4 do 2	h. z 4 do 3	h. z 3 do 2

Tabulka 5.3: Ukázka seznamu `myFinal` pro snímky sady `yosemite` (datová sada na obrázku 3.1). Tučným písmem označený snímek je brán jako hlavní, tj. do jeho souřadného systému jsou ostatní snímky převáděny. Použitá zkratka h. vyjadřuje homografie se snímku číslo do snímku číslo.



Obrázek 5.10: Obrázek seznamu `myFinal` schématicky naznačující, kam která homografie transformuje. Červeně označený snímek je hlavní snímek. Zelené šipky vyjadřují výsledné homografie do prostoru hlavního snímku. Obrázek se schoduje s údaji z tabulky 5.3.

Konstrukce `myFinal` probíhá následovně. Při konstrukci se používají dva cykly nad seznamem `myClosed` vnější a vnitřní. Ve vnějším cyklu se po záznamech prochází seznam `myClosed` a aktuálně aktivní záznam je vložen do prázdného seznamu `myFinal`, protože je `myClosed` seřazen podle kvality homografií vkládají se podle kvality. V `myFinal` je nyní jeden záznam obsahující `vectorIDmain` snímku označeného jako hlavní. Jako `homography` a `homographyOld` se vloží homografie z daného záznamu `myClosed` a jako `homographyMain` se vloží 3x3 matice s jedničkami na diagonále (neutrální homografie). Následně se ve vnitřním cyklu prochází seznam `myClosed`, uvnitř tohoto cyklu se prochází seznam `myFinal` a hledají se návaznosti záznamů takto. Záznam z `myFinal` si označíme `fi` a záznam z vnitřního cyklu nad `myClosed` `cl`. Nyní jsou 4 možnosti návaznosti mezi `fi` a `cl` ($\notin$ `myFinal` znamená, že dané ID se nenachází v žádném záznamu `myFinal` na pozici `vectorIDmain` ani `vectorIDcand`).

1. Pokud `fi.vectorIDmain == cl.vectorIDmain AND cl.vectorIDcand` $\notin$ `myFinal` – potom na konec seznamu `myFinal` přidáme záznam `new` obsahující:
   `new.vectorIDmain = cl.vectorIDmain`
   `new.vectorIDcand = cl.vectorIDcand`
   `new.inliersRank = cl.inliersRank`
   `new.homography = fi.homographyMain*cl.homography`
   `new.homographyOld = cl.homography`
   `new.homographyMain = fi.homographyMain`

2. Pokud `fi.vectorIDmain == cl.vectorIDcand AND cl.vectorIDmain` $\notin$ `myFinal` – potom na konec seznamu `myFinal` přidáme záznam `new` obsahující:
   `new.vectorIDmain = cl.vectorIDcand`
   `new.vectorIDcand = cl.vectorIDmain`
   `new.inliersRank = cl.inliersRank`
   `new.homography = fi.homographyMain*cl.homography`$^{-1}$
   `new.homographyOld = cl.homography`$^{-1}$
   `new.homographyMain = fi.homographyMain`

3. Pokud `fi.vectorIDcand == cl.vectorIDmain AND cl.vectorIDcand` $\notin$ `myFinal` – potom na konec seznamu `myFinal` přidáme záznam `new` obsahující:
   `new.vectorIDmain = cl.vectorIDmain`
   `new.vectorIDcand = cl.vectorIDcand`
   `new.inliersRank = cl.inliersRank`
   `new.homography = fi.homography*cl.homography`
   `new.homographyOld = cl.homography`

`new.homographyMain = fi.homography`

4. Pokud `fi.vectorIDcand == cl.vectorIDcand AND cl.vectorIDmain` $\notin$ `myFinal` – potom na konec seznamu `myFinal` přidáme záznam `new` obsahující:
   `new.vectorIDmain = cl.vectorIDcand`
   `new.vectorIDcand = cl.vectorIDmain`
   `new.inliersRank = cl.inliersRank`
   `new.homography = fi.homography*cl.homography`$^{-1}$
   `new.homographyOld = cl.homography`$^{-1}$
   `new.homographyMain = fi.homography`

Pokud se provede jeden z předešlých kroků je cyklus přes `myFinal` ukončen a vnitřní cyklus nad `myClosed` také. Ihned jsou ale spuštěny znovu, pro hledání další návaznosti. Pokud cykly prošly všechny kombinace záznamů `myFinal` a `myClosed` a k žádnému nalezení návaznosti nedošlo (tzn. do `myFinal` v daném kroku nepřidána nová návaznost), tak pokud je velikost seznamu `myFinal` stejně velká jako počet možných propojení mezi vstupními snímky je konstrukce seznamu `myFinal` ukončena, tento seznam označen za úplný a vnější cyklus nad `myClosed` je ukončen, jinak je do pomocné proměnné uložen maximální nalezený seznam `myFinal`. Obsah seznamu `myFinal` je vymazán a z vnějšího cyklu nad seznamem `myClosed` je jako hlavní záznam do `myFinal` vložena následující aktivní položka. Pokud se během výpočtu vyzkouší jako hlavní položky všechny záznamy seznamu `myClosed` (nikdy není výpočet ukončen nalezením úplného seznamu `myFinal`), tj. projde se celý vnější cyklus nad seznamem `myClosed`, vrátí se jako `myFinal` maximální nalezený seznam návazností, který byl uschovaný v pomocné proměnné.

V následující podkapitole 5.10 bude popsán princip prostého spojení snímků seznamu `myFinal` do panoramatu.

5.10 Prosté překrytí snímků

Poté, co již máme pro sousedící snímky vypočítány homografie a toto všechno uloženo v seznamu `myFinal`, můžeme přistoupit ke spojení těchto snímků. V této podkapitole se zaměříme na implementovanou funkci *stitchImagesNonGraphCut*, která snímky přes sebe prostě jednoduše překryje, pro odstranění viditelnosti přechodů nedělá zatím nic. Spojování snímků s úpravou viditelnosti přechodů bude popsáno v podkapitole 5.11.

Než můžeme přistoupit k samotnému spojení snímků, je nejdříve nutné vypočítat rozměry výsledného spojeného snímku. Tento výpočet provedeme tak, že si vytvoříme proměnné `Xmin`, `Ymin`, `Xmax` a `Ymax`. Do těchto proměnných si uložíme hodnoty **hlavního** snímku (`vectorIDmain` první položky v `myFinal`) takto:

- Xmin = 0 – minimální x-ová souřadnice snímku
- Ymin = 0 – minimální y-ová souřadnice snímku
- Xmax = šířka snímku – počet sloupců pixelů
- Ymax = výška snímku – počet řádků pixelů

Následně je pro každý záznam `fi` z `myFinal` provedena následující ukázka kódu 5.8. V té se pro každý záznam `fi` souřadnice rohových bodů daného kandidátního snímku promítnou do prostoru **hlavního** snímku. Potom se jejich promítnuté souřadnice x a y porovnají s `Xmin`, `Ymin`, `Xmax`, `Ymax` a proměnné jsou dle svého pojmenování řádně modifikovány.

```
vector<Point2f> candCorners(4);
candCorners[0] = Point2f(0, 0);
candCorners[1] = Point2f(myImage[fi.vectorIDcand].image.cols, 0);
candCorners[2] = Point2f(myImage[fi.vectorIDcand].image.cols,
                         myImage[fi.vectorIDcand].image.rows);
candCorners[3] = Point2f(0, myImage[fi.vectorIDcand].image.rows);
vector<Point2f> candCornersH(4);
//promitnuti rohovych bodu pomoci homografie
perspectiveTransform(candCorners, candCornersH, fi.homography);

for (unsigned k = 0; k < img2CornersH.size(); k++){
    Xmin = min(Xmin, candCornersH[k].x);
    Ymin = min(Ymin, candCornersH[k].y);
    Xmax = max(Xmax, candCornersH[k].x);
    Ymax = max(Ymax, candCornersH[k].y);
}
```

Zdrojový kód 5.8: Ukázka výpočtu velikosti výsledného snímku.

Po proběhnutí předcházejícího výpočtu je již z hodnot možné vypočítat rozlišení, a to tak, že šířka výsledného snímku se rovná `Xmax - Xmin` a výška výsledného snímku `Ymax - Ymin`. Protože se snímky připojují i nalevo od hlavního snímku, z toho důvodu jsou `Xmin` a `Ymin` často záporné hodnoty. Levý horní roh hlavního snímku je umístěn na pozici (0,0), a tudíž se nám snímky nalevo od něho promítají do záporných souřadnic. Proto potřebujeme pozici levého horního rohu hlavního snímku posunout doprava a dolů, aby se všechny snímky promítaly do oblasti s levým horním rohem (0,0) a pravým dolním (`Xmax - Xmin`,`Ymax - Ymin`), tzn. do kladných souřadnic. Toho docílíme tak, že hlavní snímek posuneme pomocí zkonstruované matice posunu (viz. vzorec 5.4) a tuto matici násobením také aplikujeme na matice homografie promítaných snímků.

$$posun = \begin{bmatrix} 1 & 0 & -Xmin \\ 0 & 1 & -Ymin \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Dále, abych zabránil výstupům příliš obrovských snímků, zmenšuji rozlišení výstupního snímku podle toho, kolik zabírá místa v paměti tzn. (řádků snímku)∗(sloupců snímku)∗(velikost datového typu pixelu) na maximálních 25MB, což při standardní velikosti datového typy pixelu CV_8UC3 (tří-kanálový `unsigned char`) 3B umožňuje vytvořit snímky o rozlišení cca 8000x1000 px, což je na výstup dostatečné.

Samotné zmenšení rozlišení probíhá, pokud není pro originální velikost splněna podmínka maximální povolené velikosti. Zmenšení je realizováno tak, že se původní šířka a výška zmenšuje koeficientem 0.95, dokud není podmínka splněna. Tyto hodnoty výšky a šířky jsou následně použity při inicializaci výstupních snímků. Protože dojde ke zmenšení výstupních snímků, musíme toto zmenšení aplikovat i na promítané snímky do výstupu. To realizujeme pomocí matice na změnu měřítka (viz. vzorec 5.5), kde hodnotu `scaleY` získáme jako (zmenšená výška)/(originální výška) a `scaleX` jako (zmenšená šířka)/(originální šířka).

$$meritko = \begin{bmatrix} scaleX & 0 & 0 \\ 0 & scaleY & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

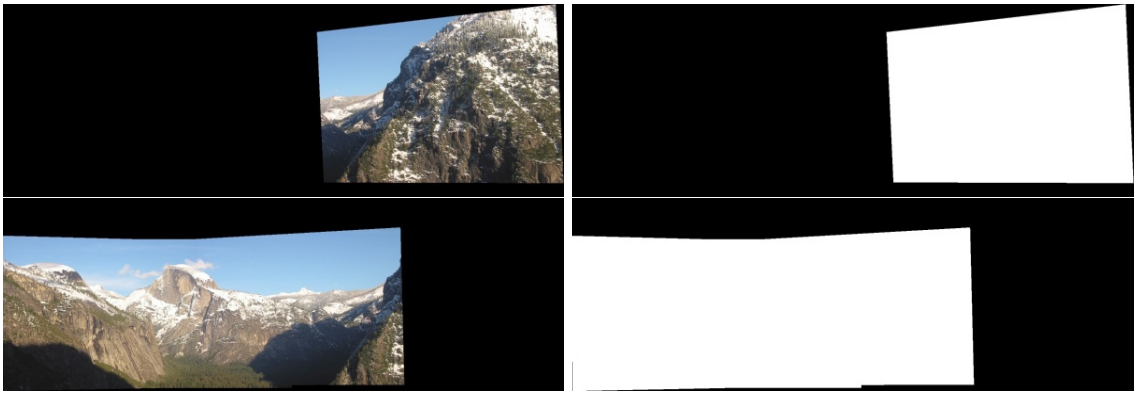
V tuto chvíli se již přistoupí k samotnému spojování snímků. Na transformaci snímku podle homografie je využita funkce knihovny OpenCV `cv::warpPerspective(src, dst, M, dsize, flags)`. Parametry této funkce jsou:

- `src` – vstupní snímek.
- `dst` – výstupní pole, v našem případě velikosti výstupního panoramatu.
- `M` – transformační matice, které budou aplikovány na vstupní snímek (posun, meřítko, homografie),
- `dsize` – velikost výstupního pole,
- `flags` – nastavení interpolační metody pro transformaci snímků. Použita byla kubická metoda interpolace `INTER_CUBIC`.

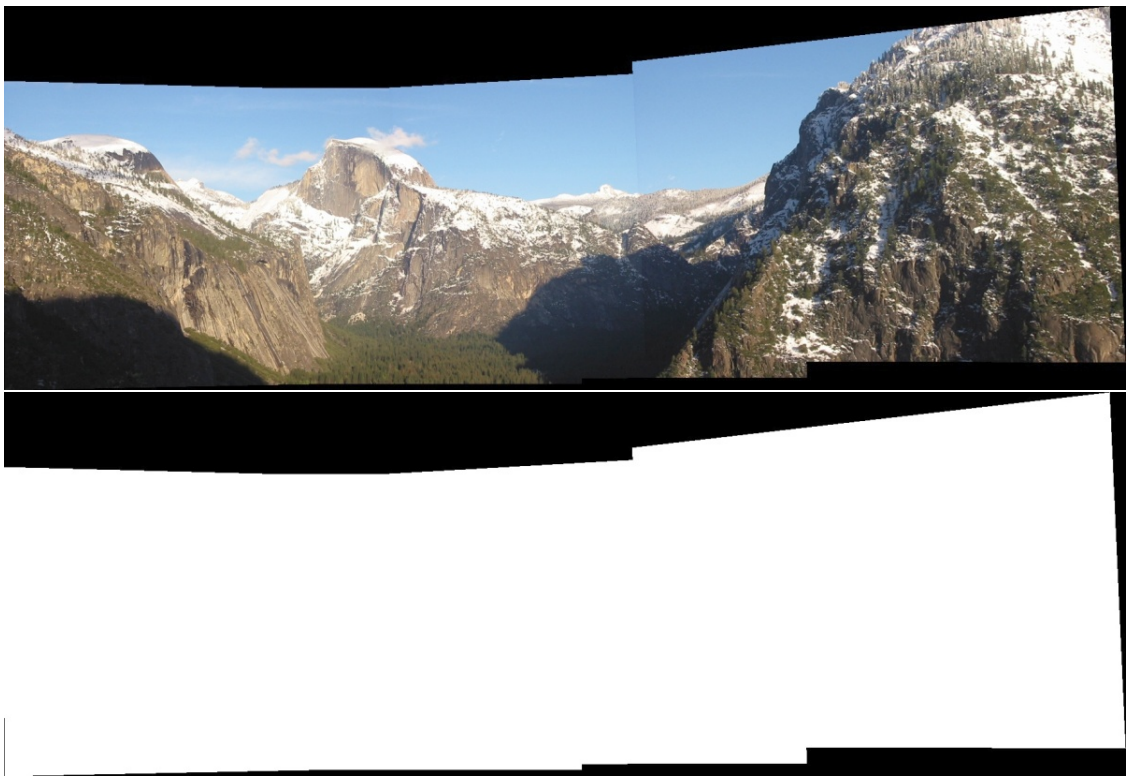
Výstupem popisované funkce *stitchImagesNonGraphCut* je proměnná `image`, která obsahuje výstupní panorama a proměnná `mask` vyjadřující, které pixely výstupního panoramatu jsou obrazové, a které jsou pouze okolí. Promítání jednotlivých snímků do výstupu probíhá následovně. Nejdříve je do pomocné proměnné `imgMain` pomocí funkce `warpPerspective` s daným posunem a změnou měřítka promítnut hlavní obrázek a do proměnné `mask` je promítnuta maska hlavního obrázku (ukázka obrázek 5.11). Následně jsou pixely z proměnné `imgMain`, které jsou pokryty nenulovými pixely masky překopírovány do výstupní proměnné `image`.

Obrázek 5.11: Hlavní snímek (viz. tabulka 5.3) `yosemite2` promítnutý do proměnné `imgMain` a jeho maska.

Dále se prochází seznam `myFinal` po jednotlivých záznamech a pro každý záznam jsou provedeny následující operace. Vytvoří se proměnná `imgCand` o velikosti výstupního panoramatu, do ní je promítnut snímek s `vectorIDcand` daného záznamu a také proměnná `maskCand`, do které je promítnuta maska daného snímku (ukázka obrázek 5.12). Promítání probíhá s daným posunem, změnou měřítka a maticí `homography` daného záznamu, pomocí funkce `warpPerspective`. Potom jsou pixely `imgCand` pokryté nenulovými pixely masky `maskCand` překopírovány do `image`. Pomocí funkce `bitwise_or` jsou sloučeny nenulové pixely z `maskCand` a `mask` do `mask`. Potom, co jsou popsané operace provedeny pro všechny záznamy `myFinal`, je v proměnné `image` výsledné panorama a v proměnné `mask` jeho maska, která se používá například při tvorbě výřezu z panoramatu (výsledné panorama obrázek 5.13).



Obrázek 5.12: Ukázka z jednoho kroku připojování kandidátních (navazujících) snímků. Kandidátní snímek (viz. tabulka 5.3) `yosemite4` promítnutý do proměnné `imgCand` a jeho maska `maskCand` první dva obrázky. Na druhých dvou snímcích prozatímní panorama `image` a jeho maska `mask`. Pixely prvního snímku pokryté maskou budou překopírovány do třetího snímku a masky budou sloučeny.



Obrázek 5.13: Výsledné složené panorama ze snímků (viz. tabulka 5.3) `yosemite` je uloženo v proměnné `image` a jeho maska v `mask`. Protože výsledné panorama bylo složeno prostým překryvem ve snímku jsou jasně viditelné hrany mezi přechody snímků.

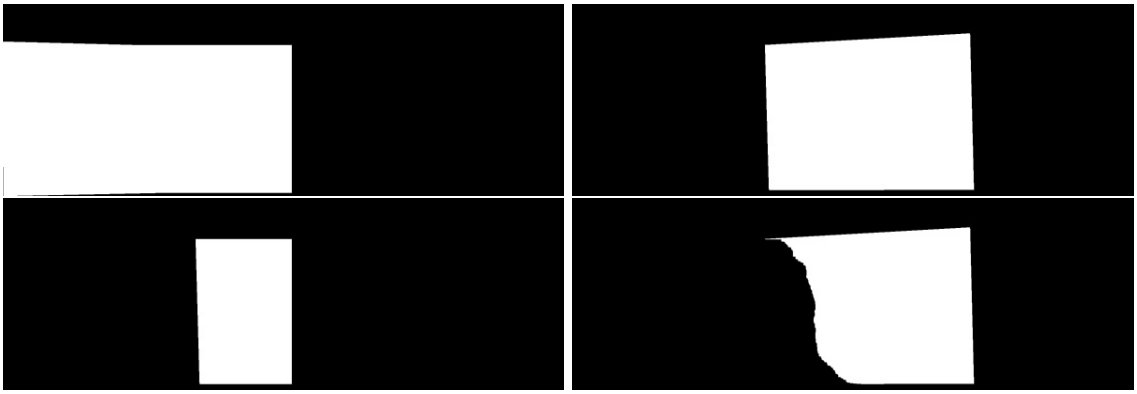
Na obrázku 5.13 je jasně vidět hrana přechodu mezi snímky, viditelnost přechodů budeme odstraňovat pomocí algoritmu Graph Cuts, jehož implementace a použití je popsáno

v následující podkapitole 5.11.

5.11 Spojení snímků pomocí algoritmu Graph Cuts

Spojení snímků s vyhlazením přechodů mezi nimi za použití algoritmu Graph Cuts je implementováno ve metodě *stitchImagesGraphCut* třídy `ImageStitch`. Výpočet velikosti výstupního snímku, matice posunu snímků, matice zmenšení i promítnutí hlavního snímku je absolutně identické s postupem popsaným v předchozí podkapitole . Tvorba výstupu se liší až v části, kde jsou transformovány kandidátské snímky do prostoru hlavního snímku.

Jako v předchozím případě, jsou postupně pro všechny záznamy z `myFinal`, dané snímky určené pomocí `vectorIDCand` a jejich masky promítnuty do proměnných `imgCand` a `maskCand` podle dané homografie, posunu a měřítka. Zde již ale nastává odlišná část výpočtu, jak bylo popsáno v podkapitole 2.8. V překrývající se části snímků je nutné nalézt hranici (linii nejpodobnějších pixelů mezi snímky), která vyjadřuje, které pixely jsou brány z jakého snímku. Proto je nad postupně vytvářenou maskou výstupního snímku `mask` a maskou kandidáta `maskCand` provedena funkce `bitwise_and`, která vypočte masku `maskOver`, což je maska překrývající se oblasti (ukázka na obrázku 5.14).



Obrázek 5.14: Na prvním snímku je postupně vytvářená maska `mask` výsledného panoramatu, na druhém snímku je maska aktuálně připojovaného snímku `candMask` a na třetím snímku je maska překrývající se oblasti předchozích masek `maskOver`. Na posledním snímku je ukázaná výsledná modifikovaná maska `maskCand` funkcí `computeCandMask`, podle které se překopírují obrazové pixely z `imgCand`.

V této překrývající se oblasti je nutné nalézt požadovanou hranici. Implementace nalezení této hranice je realizována metodou `computeCandMask(image, imgCand, mask, maskCand, maskOver)`, která modifikuje masku připojovaného snímku podle vypočítané hranice (čtvrtý snímek obrázku 5.14). Tato funkce masku modifikuje tak, že podle výsledné masky se do obrázku `image` (výsledný snímek) překopírují pouze ty pixely z `imgCand` (připojovaný snímek), které jsou hranicí nalezenou v překryvu označeny jako pixely, které do výsledného panoramatu náleží z připojovaného snímku. Popis výpočtu masky připojovaného snímku je v podkapitole 5.11.1. Následně je kompletní maska výsledného snímky sloučena s maskou připojovaného snímku a pokračuje se připojováním dalšího snímku, pokud takový existuje.

5.11.1 Výpočet hranice v překrývající se oblasti

Hranice je vypočtena pomocí dříve zmíněné funkce `computeCandMask`. Při realizaci metody Graph Cuts se sestrojuje ohodnocený graf a řeší se v něm nalezení minimálního řezu (rozdělení grafu na dvě části pomocí řezu přes co nejméně ohodnocené hrany). Na realizaci tohoto grafu a výpočtu minimálního řezu grafu byla použita pro výzkumné účely volně dostupná C++ knihovna MaxFlow v3.01*.

Funkce `computeCandMask(image, imgCand, mask, maskCand, maskOver)` pracuje následovně. Nejdříve se vytvoří matice `graphIndexes` velikosti výstupního snímku, kde všechny položky této matice mají hodnotu `INT_MAX`. Následně se projde pixel po pixelu maska překryvu snímků `maskOver` a na pozice nenulových pixelů této masky se do `graphIndexes` přiřadí postupně unikátní čísla od 0. Díky tomuto očíslování víme kolik pixelů je v překrývající se oblasti, tudíž kolik uzlů bude mít vytvářený graf. Následně je vytvořen graf `g` viz. zdrojový kód 5.9 a je do něj vytvořen daný počet uzlů. Poté se pro uzly reprezentující sousedící pixely v překrývající se oblasti vypočítá ohodnocení hrany, podle vzorce 2.59, a s tímto ohodnocením je vytvořena hrana mezi danými uzly (`add_edge` ukázka kódu 5.9).

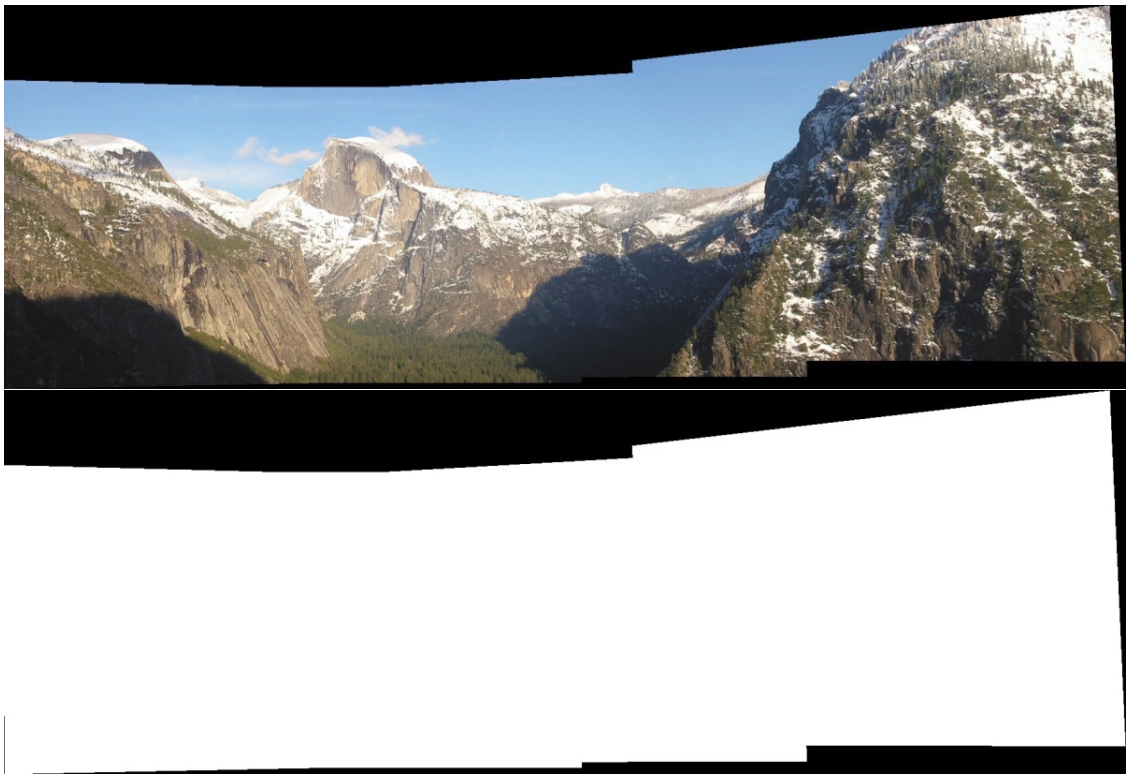
```
//index = pocet pixelu v prekryvajici se oblasti
GraphType *g = new GraphType(index, 4 * index); //vytvoreni grafu
g->add_node(index);                              //pridani daneho poctu
uzlu
//prirazeni hodnoty v1 hrane z uzel1ID do uzel2ID a naopak
g->add_edge(uzel1ID, uzel2ID, v1, v1);
//pripojeni k Source
g->add_tweights(uzel1ID, INT_MAX, 0);
//pripojeni k Sink
g->add_tweights(uzel2ID, 0, INT_MAX);
//vypocet min cut
g->maxflow();
//kontrola zda je uzel pr rozdeleni v oblasti Source nebo Sink
if (g->what_segment(uzel1ID) == GraphType::SOURCE)
    maskCand.at<uchar>(pozice_uzlu1ID) = 0;
else
    maskCand.at<uchar>(pozice_uzlu1ID) = 255;
```

Zdrojový kód 5.9: Ukázka inicializace grafu a samotný výpočet minimálního řezu grafu.

Nyní je nutné určit, které uzly jsou připojeny k uzlu **Source**, a které k uzlu **Sink**. Mezi těmito uzly následně probíhá hledání minimálního řezu. K uzlům **Source** a **Sink** jsou připojovávány pouze uzly nacházející se na hranici překrývající se oblasti, což je oblast, kde nenulové pixely sousedí s nulovými (pomezí černých a bílých pixelů). K uzlu **Source** jsou připojeny ty uzly (pixely) jejichž soused ve čtyř-okolí má nulovou hodnotu, ale tentýž sousedský pixel v `mask` (maska výstupního snímku) je nenulový a zároveň tento sousedský pixel v `maskCand` (maska připojovaného snímku) je nulový. Naopak k uzlu **Sink** jsou připojeny ty pixely (uzly) jejichž soused má nulovou hodnotu, a tento sousedský pixel v `maskCand` je nenulový a v `mask` je nulový. Způsob připojení je ukázán ve zdrojovém kódu 5.9. Na ukázce je také znázorněno jakým příkazem se provede výpočet minimálního řezu a také to, jak se otestuje, který pixel (uzel) z překrývající se oblasti patří do `maskCand`, a který ne. Ve zdrojovém kódu je znázorněno přepsání pixelů masky podle příslušnosti, výstupem je proto

*http://vision.csd.uwo.ca/code/

upravená maska viz. čtvrtý snímek obrázku 5.14.



Obrázek 5.15: Výsledné složené panorama ze snímků (viz. tabulka 5.3) `yosemite` je uloženo v proměnné `image` a jeho maska v `mask`. Skládání bylo provedeno metodou Graph Cuts a je vidět, že přechody mezi jednotlivými snímky nejsou viditelné.

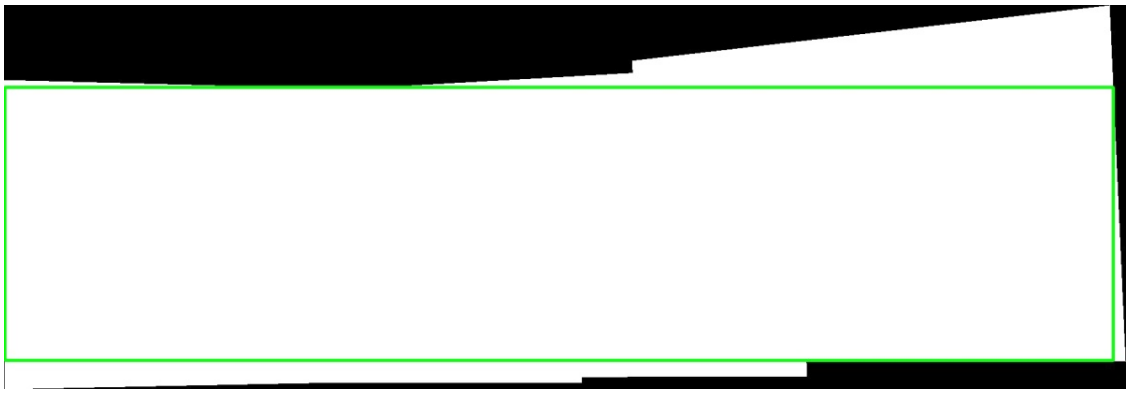
Na obrázku 5.15 je možné vidět výsledné panorama v perspektivní projekci po složení pomocí metody Graph Cuts. Při srovnání s výstupem ukázaným na obrázku 5.13 vidíme absenci rušivých přechodových hran mezi spojovanými snímky. V následující podkapitole 5.12 bude popsán princip vytváření maximálního výřezu obrazu, bez okrajových pixelů, nad zkonstruovaným panoramatem.

5.12 Výřez ze snímku

Pro realizaci možnosti vyříznutí pouze obrazových dat ze sestrojeného panoramatu je ve třídě `ImageStitch` implementována metoda *computeCropImage*, jejímž vstupem je výsledné panorama a jeho maska a výstupem je výřez z tohoto panoramatu. Výpočet je založen na nalezení největší možné obdélníkové oblasti v "bílé" oblasti masky.

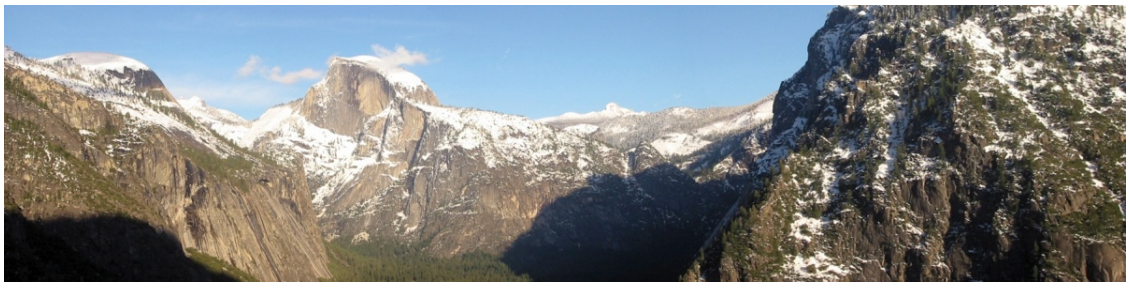
Realizace tohoto výpočtu je provedena následovně. Nejdříve je kvůli snížení výpočetní náročnosti zmenšena velikost (rozlišení) masky na čtvrtinu. Poté je pro masku vytvořena nová proměnná a pro nenulové pixely původní masky je do nové masky vložena nulová hodnota a naopak (inverze masky). Aby bylo možné rychle zjišťovat, zda se v daném kontrolovaném obdélníku nachází pouze obrazové pixely (tj. v nové masce je součet všech pixelů daného obdélníku je nula), je pro novou masku zkonstruován její integrální obraz (viz. 2.2.3). Díky němu jde pomocí čtyř hodnot spočítat součet hodnot pixelů ve vymezeném obdélníku,

pokud je součet nulový obdélník se nachází v prostoru obrazových dat. Následně jsou v nové masce průchodem přes všechny pozice pixelů hledány obdélníky splňující danou podmínku a největší nalezený je vždy uschován. Kvůli tomuto průchodu byla maska zmenšena, aby se nemuselo procházet přes vysoké množství pixelů.



Obrázek 5.16: Na snímku je maska panoramatu s označenou oblastí pro výřez. Maska je z panoramatu na obrázku 5.15.

Potom, co je tento obdélník nalezen, pak musí kvůli předchozímu zmenšení masky být jeho poziční souřadnice i výška a šířka čtyřikrát zvětšeny. Následně je kvůli tomuto zvětšení provedena na daných souřadnicích kontrola, zda souřadnice opravdu sedí do obrazové oblasti masky. Kvůli zvětšení by souřadnice obdélníku mohly o pár pixelů ležet v okrajové oblasti a proto je případně provedena korekce těchto souřadnic. Ukázka nalezeného obdélníku je na obrázku 5.16. Následně je již podle vypočtených souřadnic realizován samotný výřez ze snímku viz. obrázek 5.17.



Obrázek 5.17: Snímek ukazuje samotný výřez z panoramatu na obrázku 5.15. Podle oblasti označené na 5.16.

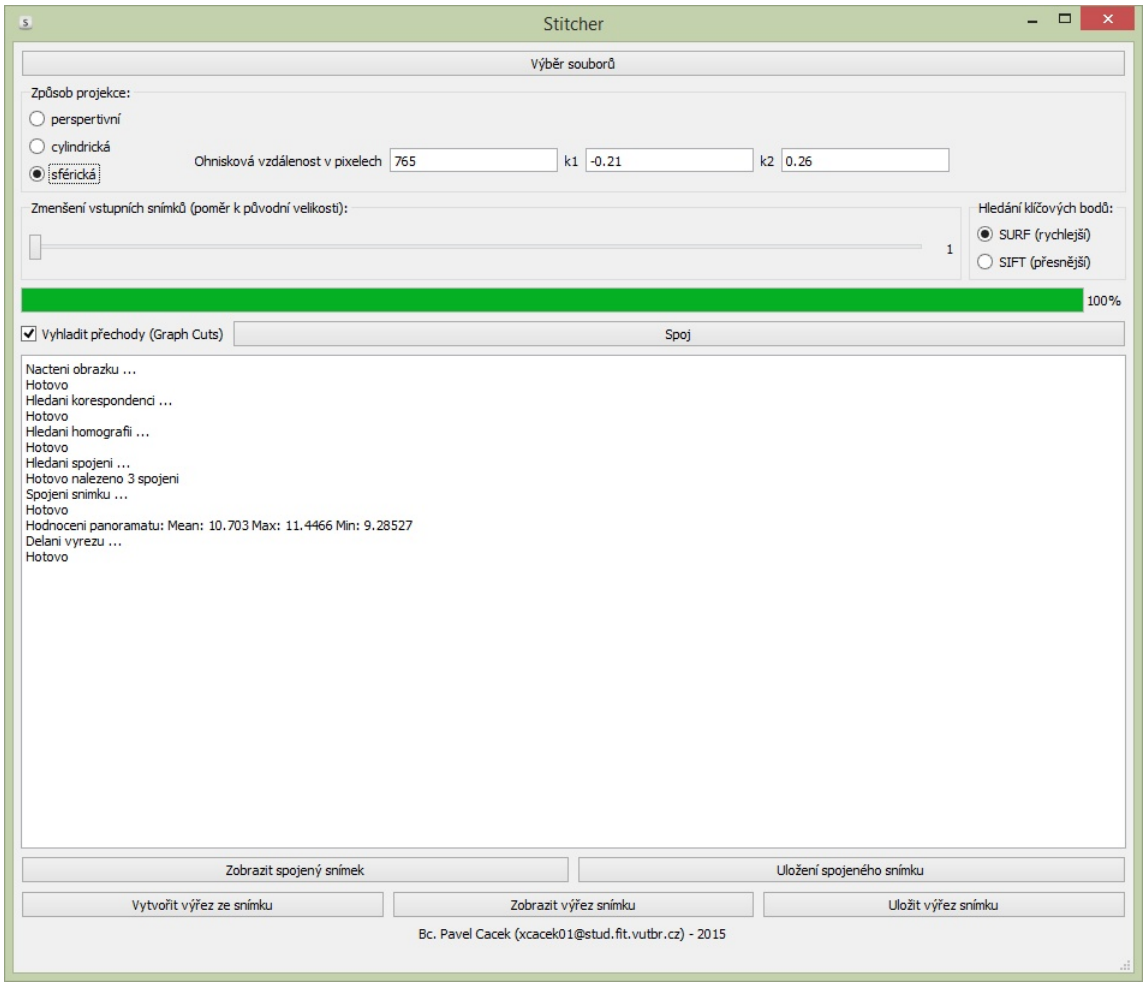
5.13 Implementace navržené metriky

V metodě *computeMetric* třídy `ImageStitch` je implementována metrika popsaná v 4.1. Postupně nad všemi dvojicemi z `myFinal` (S v daných vzorcích) jsou vypočteny hodnoty *mean*, *max* a *min*. Vypočteny jsou tak, že ve snímku s id `vectorIDcand` je náhodně vygenerována pozice pixelu. Souřadnice této pozice jsou pomocí `homographyOld` transformovány do prostoru snímku `vectorIDmain` a pokud transformovaný pixel leží ve snímku `vectorIDmain`, pak je pozice označena jako validní, jinak je generován nový náhodný bod v prostoru

snímku `vectorIDcand`. Nad danými validními pozicemi (hodnotami pixelů) je ve snímcích `vectorIDcand` a `vectorIDmain` vypočtena eulerovská vzdálenost hodnot pixelů na těchto pozicích. Takto je pro každou dvojici z `myFinal` výpočet proveden pro 100 náhodných validních bodů a zprůměrován. Zapamatován je výsledek nad nejlepší dvojicí *min*, nad nejhorší dvojicí *max* a průměr všech dvojic *mean*.

Pro ukázku výstupu metriky má panorama na snímku 5.15 tyto hodnoty $mean = 11.14$, $max = 11.68$ a $min = 10.40$. Tyto hodnoty označují kvalitně spojené panorama, více o vztahu velikosti těchto hodnot v závislosti na vizuální kvalitě výstupu je popsáno v kapitole 6.

5.14 Grafické rozhraní aplikace `Stitcher`



Obrázek 5.18: Grafické prostředí vytvořené aplikace.

Implementované třídy `Stitcher` a `Computation`, a jejich metody realizují řídící a výpočetní podporu pro instanci třídy `ImageStitch`, která, jak bylo zmíněno, obstarává samotnou tvorbu výstupního panoramatu. Třída `Stitcher` realizuje pomocí **Qt** konstrukci grafického rozhraní aplikace (viz. obrázek 5.18). Samotné grafické rozhraní je navrženo po-

mocí nástroje **Qt Designer** jako **Qt** `ui` šablona. Tlačítka z této šablony jsou pomocí funkce `connect` přes *signály* a *sloty* navázány na metody třídy `Stitcher`. Tyto metody, podle své funkce a nastavených parametrů, v položkách grafického rozhraní, spouští metody instance třídy `Computation`. Metody třídy `Computation` řídí přímo konstrukci výsledného panoramatu spouštěním metod třídy `ImageStitch` a ošetřují chyby, které se mohou vyskytnout např. nenalezení žádného spojení mezi vstupními snímky atd. Instance třídy `Computation` běží v samostatném vláknu pomocí **QThread**, aby v ní prováděné náročné výpočty neovlivňovali chování grafického prostředí. Pokud by výpočty nebyly realizovány v samostatném vlákně, pak by grafické rozhraní při konstrukci panoramatu "zamrzlo", dokud by konstrukce panoramatu neskončila.

Grafické rozhraní aplikace zobrazené na obrázku 5.18 se ovládá následovně. Nejdříve uživatel vybere tlačítkem **Výběr souborů** snímky, ze kterých chce vytvořit jedno panorama. Následně zvolí způsob projekce, případně zadá ohniskovou vzdálenost (volitelně k_1, k_2 parametry radiálního zkreslení objektivu), se kterou byly snímky fotografovány. Vybere metodu, kterou budou hledány klíčové body a zvolí, zda budou přechody vyhlazeny, nebo ne. Následně po stisku tlačítka **Spoj** začne samotná konstrukce panoramatu. Informace o průběhu jsou vypisovány do textového pole. Poté je zkonstruovaný výstup zobrazen v samostatném okně. Toto okno může uživatel zobrazit sám po stisku tlačítka **Zobrazit spojený snímek** a tento snímek si může uložit pomocí tlačítka **Uložení spojeného snímku**. Také je po zkonstruování panoramatu možné z něho vytvořit výřez za pomoci tlačítka **Vytvořit výřez ze snímku**. Tento výřez je možné zobrazit tlačítkem **Zobrazit výřez ze snímku** a uložit tlačítkem **Uložit výřez snímku**.

Kapitola 6

Vytvořená panoramata z datových sad a jejich hodnocení

V této kapitole budou ukázána výsledná panoramata vytvořená ze snímků datových sad pomocí navrženého (kapitola 4) implementovaného (kapitola 5) systému pro tvorbu panoramat. Systém byl testován na jednořadých panoramatech, tzn. snímky na sebe horizontálně navazují. V první části budou ukázány některá panoramata spojená v perspektivní obrazové projekci (sekce 6.1), to jsou ty, jejichž snímky zabíraly do 120° horizontálního úhlu záběru.

V další části této kapitoly budou zobrazeny a popsány výsledná panoramata v cylindrické a sférické projekci (sekce 6.2) to jsou většinou tzv. 360°–vá panoramata (zabírají 360° horizontálního úhlu). U těchto panoramat bylo potřeba použít ohniskovou vzdálenost objektivu v pixelech, pokud tato hodnota byla dodána spolu s datovou sadou, pak byla použita (případně mírně upravena na základě metody zkoušení a vizuální kontroly kvality výstupu). U těch datových sad, kde nebyla hodnota ohniskové vzdálenosti uvedena, byla co nejlépe odpovídající hodnota nalezena testováním různých hodnot ohniskových vzdáleností, za pomoci porovnávání hodnocení panoramatu (4.1), vizuální stránky výstupů, až bylo dosaženo uspokojivého výsledku. Pokud byly u datových sad uvedeny parametry radiálního zkreslení objektivu k_1 a k_2, byly tyto hodnoty použity, jinak byly tyto parametry nastaveny jako nulové. Pokud u dané datové sady, bylo dostupné výsledné panorama vytvořené autory sady, bude ukázáno a porovnáno s výstupem realizované aplikace.

Vždy bude ukázán výstup pro metodu nalezení klíčových bodů SURF nebo SIFT podle toho, který se podle metriky a vizuální kvality zdál lepší, a také jeho výřez realizovaný pomocí implementované funkce. Pro všechny výstupní panoramata bude použito spojování pomocí metody Graph Cuts pro vyhlazení přechodů a odstranění artefaktů. Bude možné si všimnout, že pro panoramata skládající se ze snímků, které jsou jasově vyrovnané, tato metoda pracuje velmi dobře (přechody nejsou viditelné), ale pro snímky s různými jasy jsou její výstupy neuspokojivé. Problém u jasově (i barevně) nevyrovnaných snímků spočívá v principu metody Graph Cuts (2.8.1), kdy se ideální hranice počítá jako linie minimálních rozdílů pixelů překrývající se oblasti, ale tím, že se liší jasy (barvy), liší se i pixely, tudíž hranice je nalezena, ale přechod je kvůli vyšším rozdílům jasů (barev) stejně viditelný.

U každé ukázky bude uvedeno z kolika snímků se daná datová sada skládá, a z kolika těchto snímků bylo složeno výsledné panorama. Je totiž možné, že mezi některými snímky nemusí být nalezeny uspokojivé vazby, a proto se výsledné panorama může skládat jen s podmnožiny vstupních snímků. Implementovaná aplikace uživateli vždy sdělí, z kolika spojení se výsledné panorama skládá.

Z důvodu, že výstupní snímky jsou v této diplomové práci s ohledem na tisk a možnosti zobrazení na formátu A4 zmenšeny, jsou v plném rozlišení, jak byly vytvořeny, umístěny na přiloženém DVD.

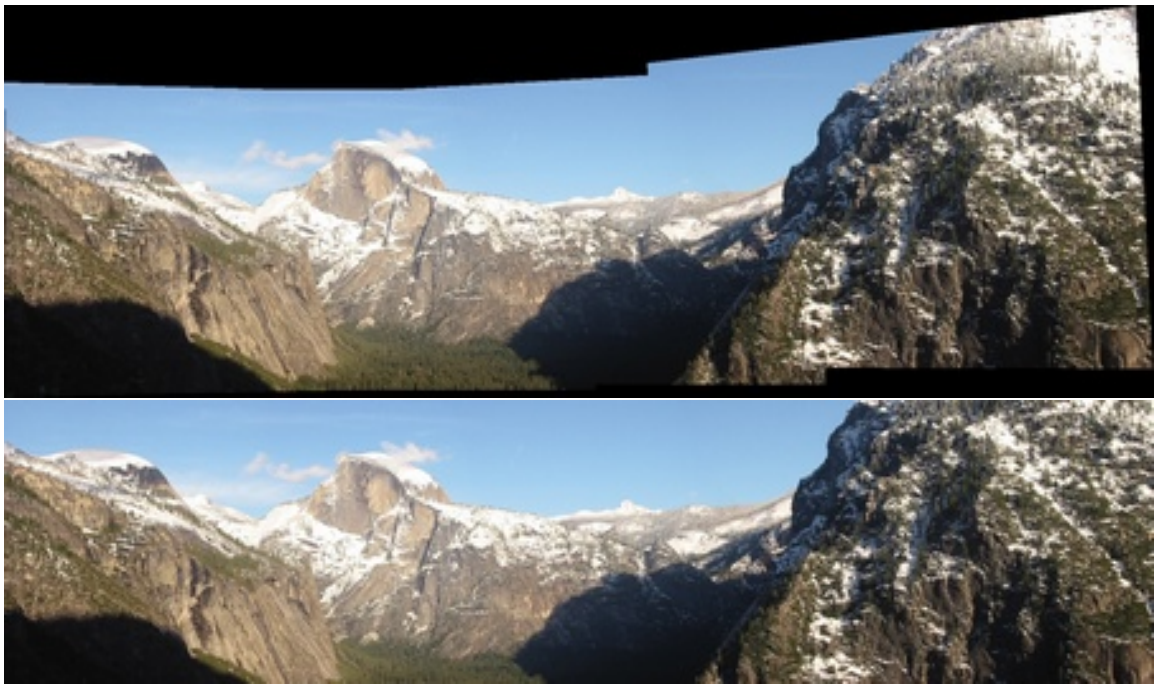
V části 6.3 bude ukázáno, jak vypadají výstupy při nevalidně nastavené ohniskové vzdálenosti, nebo při perspektivním spojování snímků s horizontálním úhlem záběru nad 120°, a také panoramata, u kterých se nepodařilo správně určit homografii mezi dvojicí snímků, ale tyto snímky byly navrženým systémem vyhodnoceny jako sousedící (tzn. chybní sousedé).

V poslední části (sekce 6.4) bude zhodnocena kvalita navržené metody pro hodnocení panoramat (4.1) a ukázáno hodnocení pro různě nastavené parametry ohniskové vzdálenosti.

6.1 Panoramata spojená v perspektivní projekci

V této podkapitole budou ukázány některé datové sady, které byly používány pro testování panoramat spojovaných v perspektivní projekci.

Na první ukázce (obrázek 6.1) je perspektivně spojené panorama snímků `yosemite` z datové sady [26]. Sada `yosemite` se skládá ze 4 snímků, výsledné panorama bylo složeno ze všech 4. Hodnocení tohoto výstupu pro klíčové body nalezené metodou SURF je $mean = 10.06$, $max = 10.33$ a $min = 9.86$ a pro SIFT je $mean = 11.14$, $max = 12.32$ a $min = 10.22$. Hodnocení je lepší pro metodu SURF, a proto je na obrázku 6.1 výstup s použitou metodou na hledání klíčových bodů SURF. Spojení se zdá být dle vizuální kontroly i metriky kvalitní, žádné artefakty nejsou pozorované.



Obrázek 6.1: Perspektivní panorama `yosemite` ze 4 snímků a jeho výřez. Klíčové body metoda SURF. Hodnocení $mean = 10.06$, $max = 10.33$ a $min = 9.86$.

Na druhé ukázce (obrázek 6.2) je výstup snímků `goldengate` datové sady [3]. Sada `goldengate` se skládá ze 6 snímků, výsledné panorama bylo složeno ze všech 6. Hodnocení

pro SIFT bylo $mean = 8.06$, $max = 9.46$ a $min = 5.59$ a pro SURF $mean = 8.25$, $max = 10.87$ a $min = 5.89$, a proto v zobrazené ukázce (obrázek 6.2) je použit výsledek pro nalezené SIFT klíčové body. Na tomto výstupu také nejsou viditelné vady a spojení vypadá dobře.

Obrázek 6.2: Perspektivní panorama `goldengate` z 6 snímků a jeho výřez. Klíčové body metoda SIFT. Hodnocení $mean = 8.06$, $max = 9.46$ a $min = 5.59$.

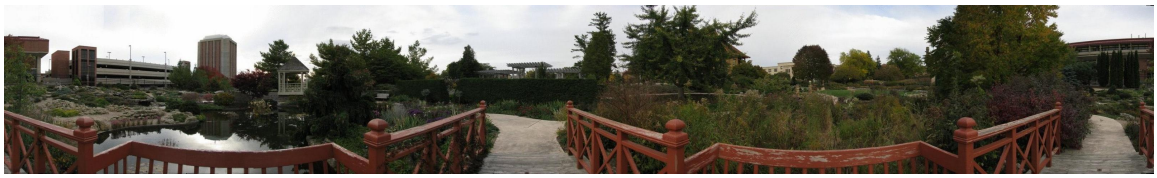
Perspektivní panoramata složená navrženým systémem vypadají vizuálně kvalitní a bez artefaktů.

6.2 Panoramata spojená v cylindrické/sférické projekci

V této podkapitole jsou ukázány výstupy navrženého systému pro tvorbu panoramat, pro všechny datové sady, které byly používány při testování a ladění sférické a cylindrické projekce. Většina výstupů je ukázána pro sférickou projekci, protože se výstupy při ní zdály vizuálně kvalitnější.

Prvním výstupním panoramatem v cylindrické/sférické projekci je sférické panorama ze snímků `park` datové sady [29]. Sada `park` se skládá z 32 snímků a do výsledného panoramatu jsou zahrnuty všechny. Nastavená ohnisková vzdálenost výstupního snímku byla 665px a parametry radiálního zkreslení dle [29] jsou $k_1 = -0.22892$ a $k_2 = 0.27797$. Hodnoty metriky hodnocení pro klíčové body metodou SURF byly $mean = 8.21$, $max = 13.09$ a $min = 5.62$ a pro klíčové body SIFT $mean = 8.52$, $max = 11.88$ a $min = 5.75$. Z důvodu lepšího hodnocení $mean$ u klíčových bodů metody SURF je ukázáno výsledné panorama s tímto nastavením (obrázek 6.3). Všechny spoje vytvořeného panoramatu se zdají správně nalezené, v obloze jsou pouze trochu viditelné přechody mezi snímky (obloha snímků datové sady má mírně proměnlivou barvu). U této sady je k dispozici řešení vytvořené autory této sady [29], to je pro porovnání ukázáno na obrázku 6.4. Výsledné panorama ze systému je referenčnímu velmi blízké.

Obrázek 6.3: Sférické panorama `park` z 32 snímků a jeho výřez. Ohnisková vzdálenost 665px korekce radiálního zkreslení $k_1 = -0.22892$ a $k_2 = 0.27797$. Klíčové body metoda SURF. Hodnocení $mean = 8.21$, $max = 13.09$ a $min = 5.62$.



Obrázek 6.4: Referenční řešení `park` [29].

Druhým sférickým panoramatem je panorama složené ze snímků `cars` datové sady [29]. Datová sada obsahuje 30 snímků a výsledné panorama se skládá z těchto 30. Ohnisková vzdálenost výstupního snímku byla 665px a parametry radiálního zkreslení dle [29] jsou $k_1 = -0.22892$ a $k_2 = 0.27797$. Hodnoty metriky hodnocení pro metodu SURF byly $mean = 8.15$, $max = 13.35$ a $min = 4.95$ a pro klíčové body SIFT $mean = 8.01$, $max = 13.54$ a $min = 3.83$. Z důvodu lepšího hodnocení $mean$ u klíčových bodů metody SIFT je ukázáno výsledné panorama s tímto nastavením (obrázek 6.5). Výsledné panorama vypadá velice dobře, pouze se mírně svažuje, což je pravděpodobně způsobeno posunem fotoaparátu při snímání. U této sady je k dispozici řešení vytvořené autory [29], to je pro porovnání ukázáno na obrázku 6.6.

Obrázek 6.5: Sférické panorama `cars` z 30 snímků a jeho výřez. Ohnisková vzdálenost 665px korekce radiálního zkreslení $k_1 = -0.22892$ a $k_2 = 0.27797$. Klíčové body metoda SIFT. Hodnocení $mean = 8.01$, $max = 13.54$ a $min = 3.83$.

Obrázek 6.6: Referenční řešení `cars` [29].

Třetí ukázka výstupu je na sadě snímků `department` datové sady [27]. Tato sada se skládá ze 17 snímků. Ohnisková vzdálenost je 682px a korekce radiálního zkreslení jsou nulové (nejsou u sady uvedeny). Hodnocení při spojení pomocí SURF je následující $mean = 23.17$, $max = 34.08$ a $min = 17.46$ pomocí SIFT $mean = 22.20$, $max = 30.41$ a $min = 14.27$. Z důvodu kvalitnějšího hodnocení je na obrázku 6.7 ukázán výstup s použitím metody SIFT. Hodnoty metriky jsou o něco málo vyšší než na předchozích ukázkách, je to tím, že předchozí ukázky obsahovaly snímky s velkými překryvy (tzn. byly nalezeny kvalitnější homografie) a také proto, že snímky v tomto panoramatu obsahují silně proměnlivě barevné plochy. I toto hodnocení označuje dobře spojený výstup. Na obrázku 6.8 je referenční řešení spojení daných snímků, ale je možné vidět, že se tam vyskytuje artefakt (červeně označený na obrázku 6.9), který se ve výstupu implementované aplikace nenachází, z důvodu použití metody Graph Cuts. Výstup je možné považovat za vizuálně velmi kvalitní.

Obrázek 6.7: Sférické panorama `department` z 17 snímků a jeho výřez. Ohnisková vzdálenost 682px korekce radiálního zkreslení $k_1 = 0$ a $k_2 = 0$. Klíčové body metoda SIFT. Hodnocení $mean = 22.20$, $max = 30.41$ a $min = 14.27$.

Obrázek 6.8: Referenční řešení `department` [27] s označeným artefaktem.

Obrázek 6.9: Ukázka výřezů mřížky na okně s artefaktem referenční snímek (vlevo) a bez artefaktu spojené navržených systémem (vpravo).

Čtvrtá ukázka výstupu navrženého systému je obrázek 6.10 složený ze snímků `camp` da-

tové sady [27] do sférického panoramatu. Použita je ohnisková vzdálenost 609px, při které výstup vycházel nejlépe. Při této ohniskové vzdálenosti jsou hodnocení panoramatu pro SIFT $mean = 17.68$, $max = 39.21$ a $min = 7.44$ (ale pouze spojení 15 snímků) a pro SURF $mean = 16.81$, $max = 29.36$ a $min = 8.01$ (všech 17 snímků). V hodnocení sice nejsou velké rozdíly, ale hodnota $mean$ je lepší pro SURF, SURF také propojuje všechny snímky sady a proto na obrázku 6.10 je zobrazeno panorama, kde klíčové body byly nalezeny metodou SURF. Na výsledném panoramatu jsou mírně viditelné přechody mezi jednotlivými snímky, ale jinak vypadá velice dobře. V článku [27] bylo ukázáno i řešení autorů (obrázek 6.11). Vytvořené panorama je v podstatě vizuálně schodné s referenčním řešením, kde na odstranění přechodů jasově různých snímků byla použita nějaká metoda rozmazání a vyhlazení (blending), to se projevilo mírnou neostrostí v přechodech.

Obrázek 6.10: Sférické panorama `camp` z 17 snímků a jeho výřez. Ohnisková vzdálenost 609px korekce radiálního zkreslení $k_1 = 0$ a $k_2 = 0$. Klíčové body metoda SURF. Hodnocení $mean = 16.81$, $max = 29.36$ a $min = 8.01$.

Obrázek 6.11: Referenční řešení `camp` [27].

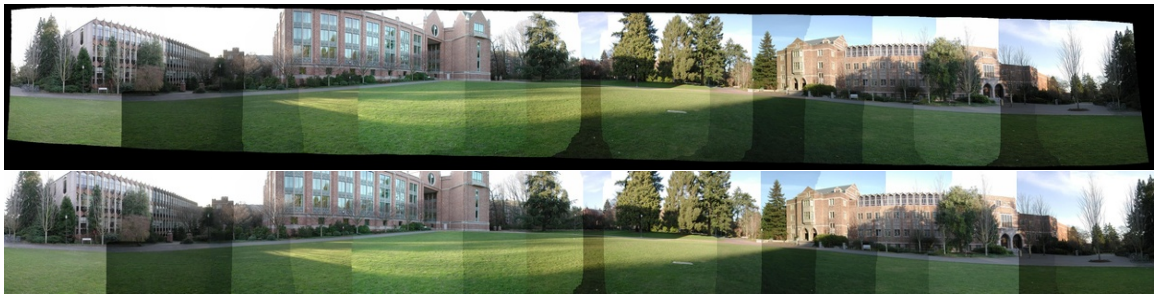
Panorama na páté ukázce (obrázek 6.12) je panorama složené v cylindrické projekci ze snímků `grass` datové sady [27]. Sada `grass` obsahuje 17 snímků a výsledné panorama je z těchto 17 složeno. Snímky sady `grass` byly pravděpodobně snímány s určitou rotací fotoaparátu. Z toho důvodu se navrženému systému nepodařilo snímky spojit úplně do roviny, ale výsledné spojené snímky tvoří oblouk. Z toho důvodu, ani výřez z tohoto panoramatu není nijak kvalitní. Panorama bylo spojováno s ohniskovou vzdáleností 673px, když byly klíčové body hledány pomocí metody SIFT podařilo se dohromady spojit pouze 16 snímků s hodnocením panoramatu $mean = 26.88$, $max = 38.54$ a $min = 15.96$, při využití klíčových bodů metody SURF se podařilo aplikaci spojit všech 17 snímků s hodnocením $mean = 27.97$, $max = 40.51$ a $min = 18.27$. Z důvodu, že za pomocí klíčových bodů metody SURF bylo možné spojit všechny snímky, ačkoliv s mírně nižším hodnocením, je na obrázku 6.12 panorama založené na SURF bodech.

Referenční řešení (obrázek 6.13) od autorů datové sady [27] také není při bližším zkoumání nijak dokonalé. Výřez je z pohledu původních snímků velice úzký a přechody jsou silně rozmazané (blending).

Obrázek 6.12: Cylindrické panorama `grass` z 17 snímků a jeho výřez. Ohnisková vzdálenost 673px korekce radiálního zkreslení $k_1 = 0$ a $k_2 = 0$. Klíčové body metoda SURF. Hodnocení $mean = 27.97$, $max = 40.51$ a $min = 18.27$.

Obrázek 6.13: Referenční řešení `grass` [27].

Šestou ukázkou výstupu z navrženého systému pro tvorbu panoramat je obrázek 6.14 složený ze snímků `campus` datové sady [26]. Jedná se o sférické panorama s nastavenou ohniskovou vzdáleností 750px a s korekcí radiálního zkreslení $k_1 = -0.15$ a $k_2 = 0$. Sada `campus` obsahuje 18 snímků. Nevýhoda této sady je v tom. že snímky jsou různě exponovány (liší se jasem), proto, ačkoliv je výsledné panorama složeno správně, jsou jednotlivé přechody mezi snímky jasně viditelné. Panorama na obrázku 6.14 je složeno za pomoci klíčových bodů metody SURF s hodnocením $mean = 42.32$, $max = 91.23$ a $min = 11.41$, protože hodnocení při spojení pomocí metody SIFT bylo o něco málo horší $mean = 42.69$, $max = 90.50$ a $min = 13.10$. Kromě viditelnosti přechodů, je toto panorama vytvořené pomocí navrženého systému vizuálně správné.

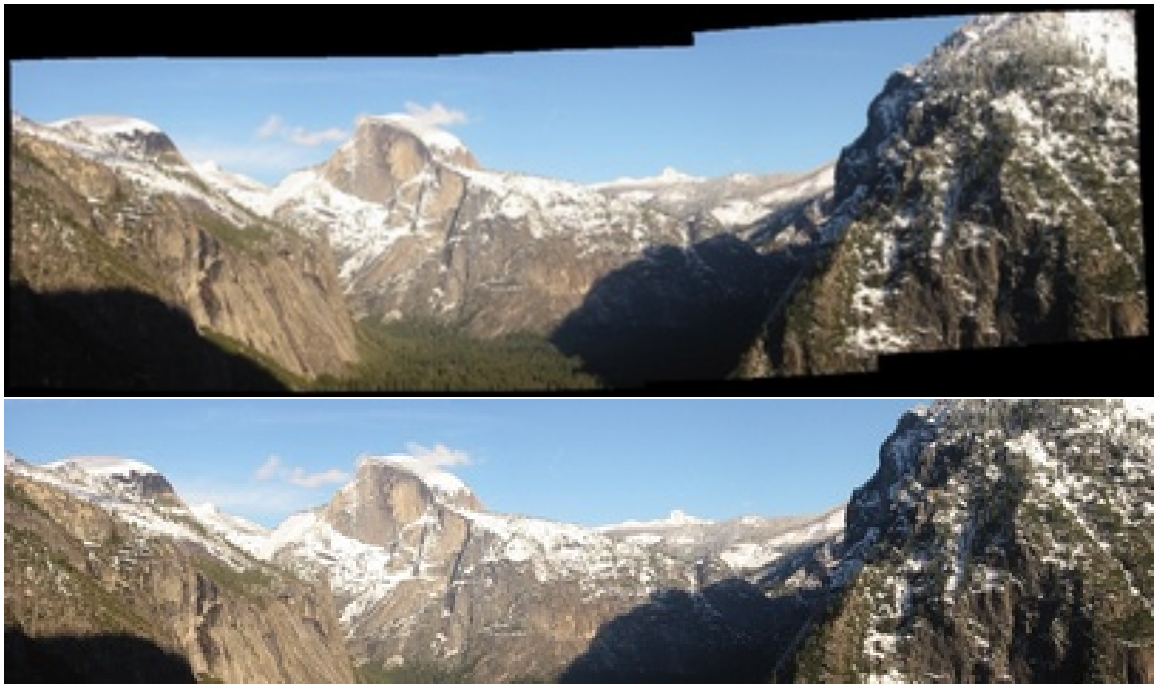


Obrázek 6.14: Sférické panorama `campus` z 18 snímků a jeho výřez. Ohnisková vzdálenost 750px korekce radiálního zkreslení $k_1 = -0.15$ a $k_2 = 0$. Klíčové body metoda SURF. Hodnocení $mean = 42.32$, $max = 91.23$ a $min = 11.41$.

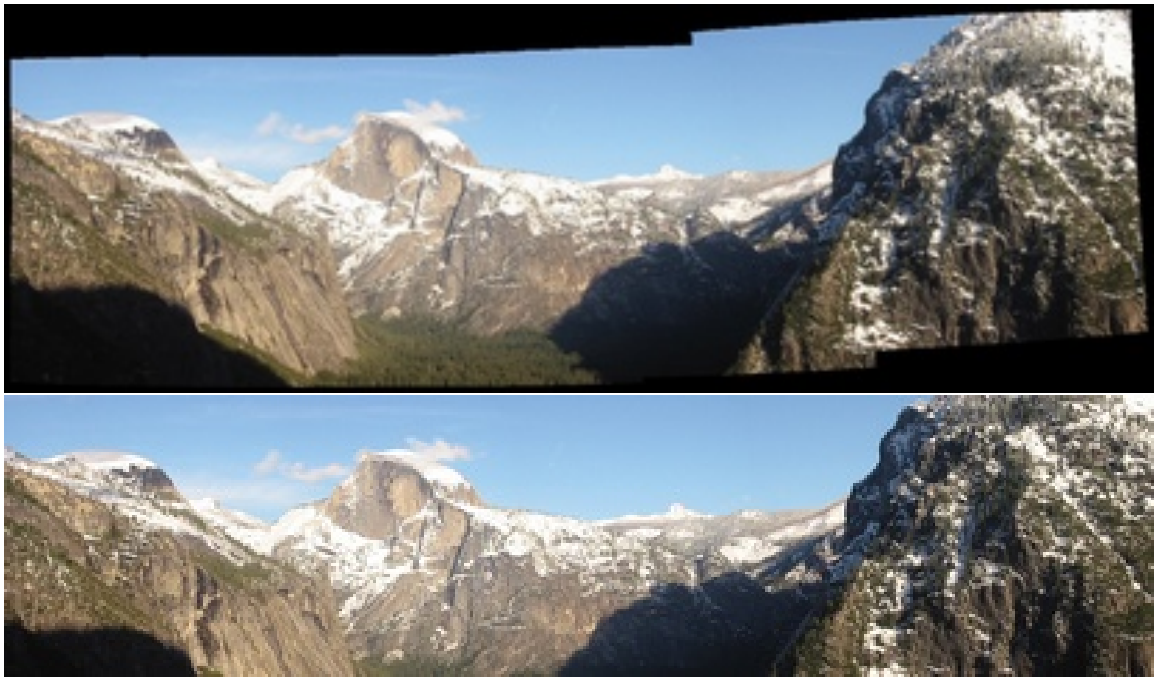
Sedmou a předposlední ukázkou (obrázek 6.15) výstupu realizovaného systému pro tvorbu panoramat, je sférické složení snímků `trees` z datové sady [26]. Datová sada těchto snímků obsahuje 18 a jejich spojení je vysoce problematické kvůli jejich obsahu. Detekce správných korespondenčních klíčových bodů v kvetoucích korunách stromů je vysoce obtížná. Z toho důvodu dělalo spojení těchto snímků navrženému systému velké obtíže, a jako u jediné datové sady se zde výrazně vyskytla vlastnost spojená s algoritmem RANSAC, která je ale pro jeho funkci nutná. Touto vlastností je **náhodný** výběr korespondenčních bodů, které mohou být špatně určené, a následný iterační výpočet co nejlepší homografie je nespolehlivý. Z tohoto důvodu se při konstrukci tohoto panoramatu stává, že při různých bězích algoritmus najde různě kvalitní homografie, které pokud jsou špatné, pak znemožní správné určení sousedních snímků, či správnost výsledného spojení. Také tam hraje určitou roli správnost určení korespondenčních klíčových bodů mezi snímky, kde se také s určitou mírou používá náhodnost. Výsledné nejlepší získané panorama (obrázek 6.15) se skládá z 18 snímků, nastavena byla ohnisková vzdálenost 649px a korekce radiálního zkreslení $k_1 = -0.15$ a $k_2 = 0$. Klíčové body byly hledány metodou SURF a hodnocení panoramatu je $mean = 24.18$, $max = 64.71$ a $min = 9.98$, při hledání klíčových bodů metodou SIFT se aplikaci panorama ze všech snímků nepodařilo sestavit (vyskytl se problém s chybným určením sousedících snímků, které ve skutečnosti sousedící nebyly viz. podkapitola 6.3). Sestavené panorama je vizuálně přesné a je bez viditelných přechodů a artefaktů.

Obrázek 6.15: Sférické panorama `trees` z 18 snímků a jeho výřez. Ohnisková vzdálenost 649px korekce radiálního zkreslení $k_1 = -0.15$ a $k_2 = 0$. Klíčové body metoda SURF. Hodnocení $mean = 24.18$, $max = 64.71$ a $min = 9.98$.

Poslední ukázkou jsou dvě panoramata sestavená ze snímků `yosemite` datové sady [26]. Jedno je sestaveno v cylindrické (obrázek 6.16) a druhé ve sférické (obrázek 6.17) projekci při nastavené ohniskové vzdálenosti 1190px (určená testováním) a korekčních koeficientech radiálního zkreslení $k_1 = -0.21$ a $k_2 = 0.26$ dle [26]. Tyto dva výstupy jednořadého panoramatu s relativně malým vertikálním zorným úhlem jsou takřka identické. Panoramata a jejich hodnocení je možné porovnat s panoramatem na obrázku 6.1, kde jsou spojené stejné snímky, ale v perspektivní projekci.



Obrázek 6.16: Cylindrické panorama `yosemite` ze 4 snímků a jeho výřez. Ohnisková vzdálenost 1190px korekce radiálního zkreslení $k_1 = -0.21$ a $k_2 = 0.26$. Klíčové body metoda SURF. Hodnocení $mean = 8.71$, $max = 9.76$ a $min = 7.87$.



Obrázek 6.17: Sférické panorama `yosemite` ze 4 snímků a jeho výřez. Ohnisková vzdálenost 1190px korekce radiálního zkreslení $k_1 = -0.21$ a $k_2 = 0.26$. Klíčové body metoda SURF. Hodnocení $mean = 9.44$, $max = 10.06$ a $min = 9.03$.

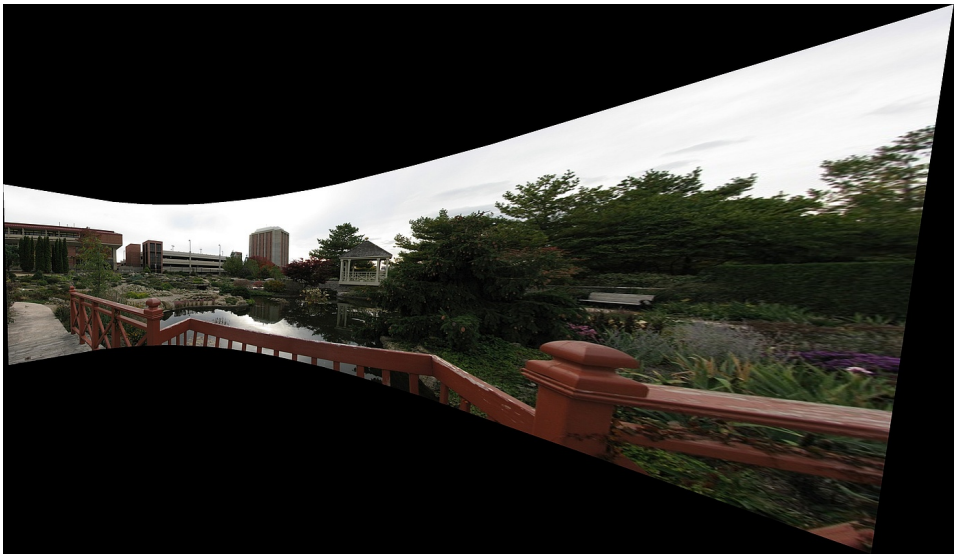
Dle prezentovaných výstupů na datových sadách je možné říci, že implementace navrženého systému je schopna kvalitně spojovat snímky do panoramat. Vizuální kvalitu výstupu nejvíce ovlivňují viditelné přechody u jasově/barevně nevyrovnaných snímků, které algoritmus Graph Cuts není schopen odstranit. Problémem jsou také snímky s malými překryvy, a také snímky, které jsou s různým natočením, potom systém nevytváří rovné panorama. Jinak je systém ve spojování velice schopný a výstupy vypadají vizuálně velice dobře.

6.3 Nekvalitní panoramata a výstupy se špatně nastavenou ohniskovou vzdáleností (nevalidní výstupy)

V této podkapitole budou ukázány výstupy navrženého systému při špatně nastavených parametrech spojovaného panoramatu. Chybná panoramata budou demonstrována na 12 navazujících snímcích `park` z datové sady [29]. Správné spojení těchto snímků je součástí obrázku 6.3.

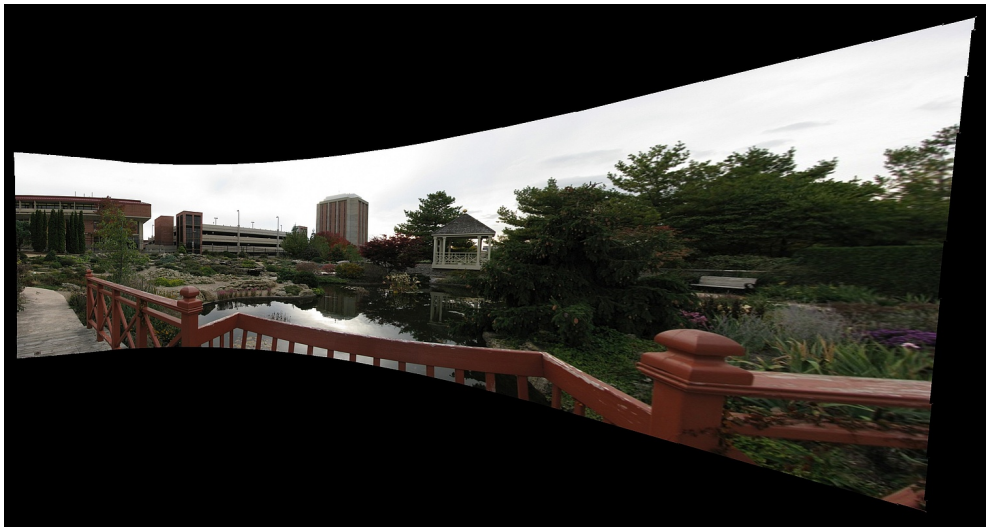
Pokud je vytvářeno perspektivní panorama pro snímky, které ale pokrývají horizontální úhel záběru větší než 120°, je výsledné panorama silně zkresleno (obrázek 6.18). Perspektivní projekce se pro tolik snímků nehodí.

Na další ukázce (obrázek 6.19) je zobrazen výstup při nastavené ohniskové vzdálenosti kratší než je správná. Výstup se podobá snímkům z objektivů typu rybí oko. Dále na obrázku 6.20 je stejný snímek s ohniskovou vzdáleností nastavenou na delší, než je správná a proto vypadá panorama podobně jako by bylo v perspektivní projekci (čím delší ohnisko tím více se podobá perspektivní projekci).



Obrázek 6.18: Perspektivní panorama z 12 snímků `park` z datové sady [29], které ale mají horizontální úhel záběru větší než nad 120°. Výsledek je silně perspektivně zkreslen. Klíčové body metoda SURF. Hodnocení $mean = 9.97$, $max = 14.56$ a $min = 6.82$.

Obrázek 6.19: Sférické panorama z 12 snímků `park` z datové sady [29]. Ohnisková vzdálenost byla nastavena na hodnotu 300px což je dvakrát méně než je správná hodnota. Korekce radiálního zkreslení $k_1 = -0.22892$ a $k_2 = 0.27797$ dle [29]. Výsledek je silně zkreslen připomíná výstupy z objektivů typu rybí oko. Klíčové body metoda SURF. Hodnocení $mean = 16.73$, $max = 23.41$ a $min = 12.48$.



Obrázek 6.20: Sférické panorama z 12 snímků `park` z datové sady [29]. Ohnisková vzdálenost byla nastavena na hodnotu 1200px což je dvakrát více než je správná hodnota. Korekce radiálního zkreslení $k_1 = -0.22892$ a $k_2 = 0.27797$ dle [29]. Výsledek je silně zkreslen připomíná výstupy při perspektivní projekci. Klíčové body metoda SURF. Hodnocení $mean = 8.11$, $max = 11.02$ a $min = 7.70$.

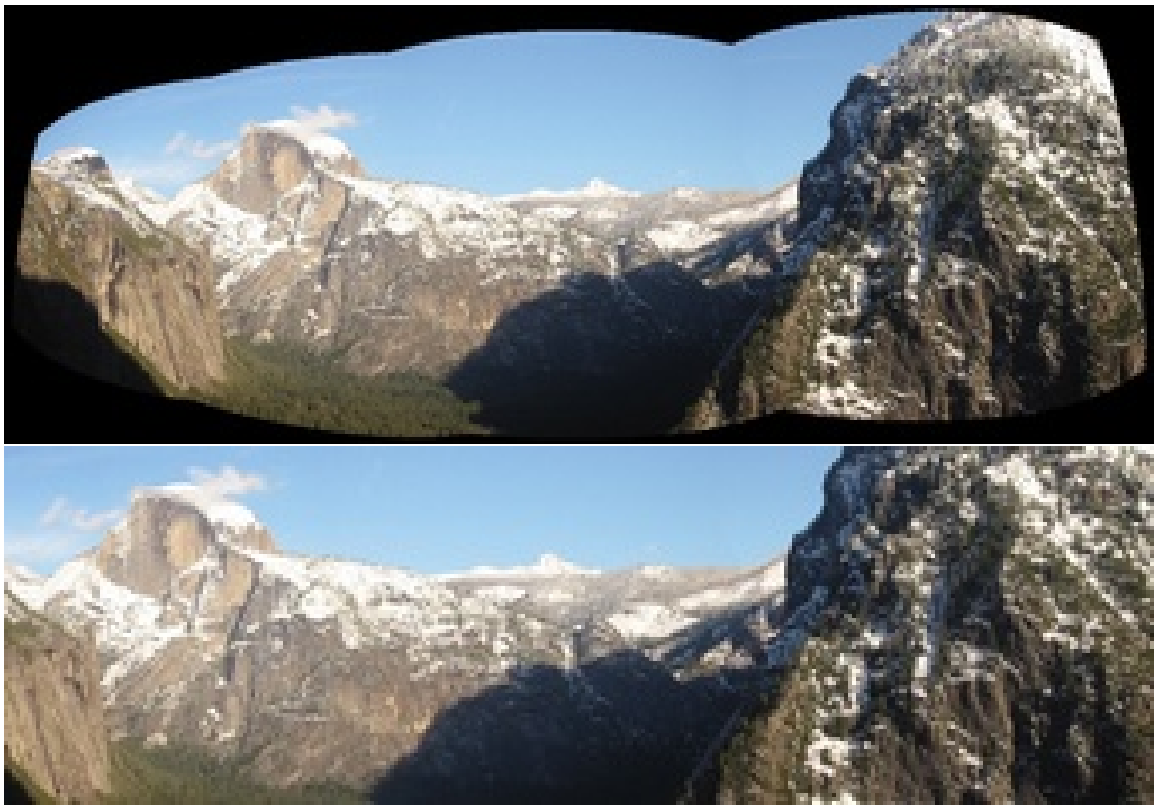
Na obrázku 6.21 je chybně sestavené panorama. Při spojování tohoto panoramatu nastala ta chyba, že navržený systém určil nevalidní homografii mezi snímky jako správnou a označil tyto snímky jako sousedící, a v tomto spojení nastala chyba, která zdeformovala výsledné panorama. Hodnocení je relativně dobré, z důvodu nalezených vazeb mezi snímky. Vizuální kvalita je ale očividně špatná, a pro nápravu je nutné upravit vstupní parametry (ohniskovou vzdálenost atd.) systému při konstrukci tohoto panoramatu, pro kvalitnější sestavení.

Obrázek 6.21: Chybně vytvořené sférické panorama ze snímků `trees` z datové sady [26]. Ohnisková vzdálenost 645px korekce $k_1 = -0.15$ a $k_2 = 0$. Klíčové body metoda SURF. Hodnocení $mean = 25.16$, $max = 44.58$ a $min = 12.25$.

6.4 Hodnocení výsledků navržené metriky

Navržená metrika tedy hodnotí kvalitu spojení jednotlivých dvojic snímků, ze kterých se skládá výstupní panorama. Hodnota jejího výstupu je navázána na vizuální kvalitu a viditelnost artefaktů v panoramatu. Výstupy této metriky byly ukázány u všech snímků v sekcích 6.1 a 6.2. Z uvedených hodnot u těchto snímků vyplývá, že daná metrika je vázána svým hodnocením, na nastavené parametry systému, podle kterých je spojování prováděno. Při volbě parametrů pro dané panorama je možné se podle ní orientovat. Na obrázku 6.22 je sférické panorama `yosemite` s úmyslně špatně nastavenými parametry, ohnisková vzdálenost nastavena na 440px korekce radiálního zkreslení $k_1 = -0.5$ a $k_2 = 0.5$. Hodnocení tohoto panoramatu je $mean = 24.20$, $max = 28.73$ a $min = 20.58$, což je výrazně rozdílné od hodnocení panoramatu ze stejných snímků na obrázku 6.17. Toto panorama se špatně nastavenými parametry vypadá vizuálně relativně dobře, ale výstup obrázku 6.17 s lépe nastavenými parametry se vizuálně zdá být daleko lepší.

Navrženou metrikou není absolutně možné hodnotit panoramata z rozdílných datových sad mezi sebou. Z vypozorovaných výsledků na datových sadách vyplynulo, že panorama, které má hodnotu $mean \leq 25$ většinou vypadá vizuálně velice dobře. Podle hodnoty max si můžeme říci, že v panoramatu existuje dvojice, která není příliš dobře spojena, ale většinou pokud je $max \leq 60$ jsou výstupy v pořádku.



Obrázek 6.22: Sférické panorama `yosemite` ze 4 snímků a jeho výřez, s úmyslně špatně nastavenými parametry pro ukázku hodnocení metriky. Ohnisková vzdálenost 440px korekce radiálního zkreslení $k_1 = -0.5$ a $k_2 = 0.5$. Klíčové body metoda SURF. Hodnocení $mean = 24.20$, $max = 28.73$ a $min = 20.58$.

Nevýhodou navržené metriky je její nerobustnost při spojování snímků s rozdílnou expozicí (jasem). Panorama z takovýchto snímků metrika hodnotí jako nekvalitní, i když kvalita spojení snímků je třeba uspokojující (viz. výsledné panorama obrázek 6.14). U takovýchto snímků bývají nejčastěji problémy s viditelností přechodů, toto se promítne zvýšením hodnocení, a proto je možné metriku základně používat i pro hodnocení vizuální kvality výstupu.

Z pozorování vyplynulo, že navrženou metriku je možné dobře použít při nastavování parametrů spojení daného panoramatu a také pro základní ohodnocení jeho vizuální kvality.

Kapitola 7

Závěr

Cílem této diplomové práce bylo vytvoření přehledu metod, pro vytváření panoramatických snímků z jednotlivých fotografií. Dále návrh a implementace systému pro tvorbu panoramat založeného na daných metodách, získání vhodných datových sad pro testování a hodnocení kvality navrženého systému. Výsledná práce splňuje všechny body zadání.

V kapitole 2 byly podrobně popsány jednotlivé metody a algoritmy, používané pro skládání panoramatických snímků, které byly nastudovány z odkazované literatury. Ze získaných znalostí o používaných algoritmech, popsaných v kapitole 2 a z literatury, byl v kapitole 4 popsán návrh systému, který bude vybrané algoritmy používat pro tvorbu panoramat. Také v této kapitole 4 byl popsán návrh metriky pro hodnocení kvality vytvářených panoramat. V kapitole 3 jsou popsány získané datové sady obrázků, které se používají pro testování realizovaného systému. V následné kapitole 5 je popsána implementace navrženého systému jako aplikace, která se nazývá `Stitcher`. V poslední kapitole 6 jsou ukázána a zhodnocena panoramata vytvořená realizovanou aplikací `Stitcher` na snímcích datových sad.

Vyvinutá aplikace `Stitcher` vytváří ze vstupních snímků panorama, dle principu navrženého systému. Podle uživatelovy volby je možné pomoci ní vytvářet perspektivní, cylindrická či sférická panoramata. U cylindrických a sférických panoramat je největší obtíží určení správné ohniskové vzdálenosti, při které byly snímky fotografovány, aby bylo možné je správně převést do dané obrazové projekce. Uživatel tuto hodnotu ohniskové vzdálenosti musí sám zadat jako vstup do aplikace. Aplikace jinak pracuje plně automatizovaně a výstupní panoramata na datových sadách vypadají vizuálně velmi dobře. Pro odstranění artefaktů v přechodech spojovaných snímků byla použita metoda Graph Cuts, která se ukázala jako velmi platná a artefakty (např. poloviční uřízlé objekty) nejsou viditelné. Jedinou slabou stránkou použité metody Graph Cuts se ukázala neschopnost plynulého spojení jasově nevyrovnaných snímků. Navržená metrika je součástí aplikace a vždy při vytváření panoramatu jsou její hodnoty zobrazeny v aplikaci. Dle zkoumání v kapitole 6 je možné se podle jejich hodnot řídit při posuzování kvality vytvořeného panoramatu.

Jako další možné rozšíření realizované aplikace by mohl být systém pro automatický odhad hodnoty ohniskové vzdálenosti snímků při cylindrických/sférických panoramatech, aby tuto hodnotu nemusel zkoumat a zadávat uživatel. Dále by bylo možné při spojování jednotlivých snímků implementovat metodu vyhlazování přechodů (např. blending), která by si poradila v odstraněním viditelných přechodů i u jasově nevyrovnaných obrázků.

Při práci na této diplomové práci jsem získal přehled o metodách používaných ve zpracování obrazu a počítačovém vidění. Práce a zkušenost z těmito metodami je pro mě velkým přínosem. Myslím, že vytváření panoramatických snímků je velice zajímavá oblast a ještě se bude velmi rozvíjet.

Literatura

[1] Bay, H.; Ess, A.; Tuytelaars, T.; aj.: Speeded-Up Robust Features (SURF). 2006.
URL http://www.vision.ee.ethz.ch/~surf/eccv06.pdf

[2] Blogging, C. R. P.: [RESEARCH] FEATURE DETECTORS AND DESCRIPTORS. [cit. 2014-12-09].
URL http://littlecheesecake.wordpress.com/2013/05/23/research-feature-detectors-and-descriptors/

[3] Brandt, T.: Transform coding for fast approximate nearest neighbor search in high dimensions. roèník IEEE Conf. on Computer Vision and Pattern Recognition, 2010.
URL http://sourceforge.net/adobe/adobedatasets/panoramas/home/Home/

[4] Brown, M.: AutoStitch.
URL http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html

[5] Brown, M.; Love, D. G.: Recognising Panoramas. 2003.
URL http://www.cs.bath.ac.uk/brown/papers/iccv2003.pdf

[6] Brown, M.; Love, D. G.: Automatic Panoramic Image Stitching using Invariant Features. 2007.
URL https://www.cs.bath.ac.uk/brown/papers/ijcv2007.pdf

[7] Brown, M. A.: *Multi-Image Matching using Invariant Features*. Dizertaèní práce, The University of British Columbia, 2005.
URL http://www.cs.bath.ac.uk/brown/papers/phd.pdf

[8] Brunner, C.: Stitching. [cit. 2015-04-20].
URL http://www.acfr.usyd.edu.au/courses/amme4710/Lectures/AMME4710-Chap7-Image%20StitchingA.pdf

[9] CAMBRIDGE in COLOUR: Panoramatic Image Projection. Navštíveno: 2015-04-28.
URL http://www.cambridgeincolour.com/tutorials/image-projections.htm

[10] Gao, J.; Kim, S. J.; Brown, M. S.: Constructing Image Panoramas using Dual-Homography Warping. 2011.
URL https://www.comp.nus.edu.sg/~brown/pdf/cvpr_dualhomography2011.pdf

[11] Harris, C.; Stephens, M.: A Combined Corner and Edge Detector. 1988.
URL http://www.bmva.org/bmvc/1988/avc-88-023.pdf

[12] He, K.; Chang, H.; Sun, J.: Rectangling Panoramic Images via Warping. 2013.
URL http://research.microsoft.com/en-us/um/people/kahe/publications/sig13pano.pdf

[13] Hugin: Hugin manual - Projections. Navštíveno: 2015-04-28.
URL http://hugin.sourceforge.net/docs/manual/Projections.html

[14] Hugin: Panorama photo stitcher.
URL http://hugin.sourceforge.net/

[15] Juan, L.; Gwun, O.: SURF applied in Panorama Image Stitching. 2010.
URL http:
//ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=5586723{&}tag=1

[16] Kolor: Understanding Projecting Modes. Navštíveno: 2015-04-28.
URL
http://www.kolor.com/wiki-en/action/view/Understanding_Projecting_Modes

[17] Komosný, P.: *Detekce odpovídajících si bodů ve dvou fotografiích*. Diplomová práce, FIT VUT v Brně, 2009.

[18] Kršek, P.: *Základy počítačové grafiky, opora IZG*. FIT VUT v Brně, [cit. 2014-12-06].

[19] Kwatra, V.; Schodl, A.; Essa, I.; aj.: Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, roèník 22, è. 3, July 2003: s. 277–286.
URL http://www.cc.gatech.edu/cpl/projects/graphcuttextures/gc-final.pdf

[20] Lowe, D. G.: Object Recognition from Local Scale-Invariant Features. 1999.
URL http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

[21] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. 2004.
URL http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

[22] Muja, M.; Lowe, D. G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. 2009.
URL http://www.cs.ubc.ca/~lowe/papers/09muja.pdf

[23] Navrátil, J.: Transformace a RANSAC, slidy do předmětu POV 2013/2014. [cit. 2014-12-06].

[24] OpenCV: Open Source Computer Vision.
URL http://opencv.org/

[25] Qt: Cross-platform application & UI development framework.
URL http://www.qt.io/

[26] Seitz, S.: CS6670 Computer Vision, Spring 2011.
URL http:
//www.cs.cornell.edu/courses/cs6670/2011sp/projects/p2/project2.html

[27] Su, Y.-F.: Image Stitching.
URL http://mpac.ee.ntu.edu.tw/~sutony/vfx_stitching/pano.htm

[28] Szeliski, R.; Winder, S.; Brown, M.: Multi-Image Matching using Multi-Scale Oriented Patches. 2003.
URL http:
//cs.brown.edu/courses/cs195g/asgn/proj6/resources/ImageMatching.pdf

[29] Verma, C. S.; Mon-Ju: Panoramic Image Mosaic.
URL
http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/imagemosaic.html

[30] Wallace, E.: CS 195-G: Automated Panorama Stitching. [cit. 2014-12-06].
URL http://cs.brown.edu/courses/csci1950-g/results/proj6/edwallac/

[31] Wikipedia: Summed area table — Wikipedia, The Free Encyclopedia. 2014, [cit. 2014-12-08].
URL http://en.wikipedia.org/w/index.php?title=Summed_area_table

[32] Zagaroza, J.; Chin, T.; Brown, M. S.; aj.: As-Projective-As-Possible Image Stitching with Moving DLT. 2013.
URL http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Zaragoza_As-Projective-As-Possible_Image_Stitching_2013_CVPR_paper.pdf

Příloha A

Popis ovládání vytvořeného programu `Stitcher`

Přiložené DVD obsahuje, jak zdrojové kódy vytvářené aplikace, tak její přeloženou binární formu `Stitcher.exe` s přiloženými všemi potřebnými knihovnami pro její běh na systémech Windows. S touto aplikací pro konstrukni panoramat se po jejím spuštění pracuje následovně. Nejdříve uživatel vybere tlačítkem **Výběr souborů** snímky, ze kterých chce vytvořit jedno panorama. Následně se v položce *Způsob projekce* zvolí použitá obrazová projekce při konstrukci panoramatu. Na výběr je `perspektivní`, `cylindrická` nebo `sférická` projekce (výchozí je `perspektivní`). U `cylindrické` a `sférické` projekce musí uživatel zadat ohniskovou vzdálenost objektivu pro správný převod vstupních snímků. Volitelně pokud zná korekční koeficienty radiálního zkreslení daného objektivu k_1 a k_2 může je zadat také, a nebo v těchto políčkách ponechat 0. Dále v položce *Hledání klíčových bodů* uživatel zvolí zda budou klíčové body ve snímcích hledány metodou `SURF`, nebo `SIFT` (ve výchozím nastavení je vybrána metoda `SURF`). Také je možné v položce *Zmenšení vstupních snímků* zvolit pomocí táhla zda mají být vstupní snímky zmenšeny, je to z důvodu menšího výstupu, ale i urychlení výpočtu panoramatu, protože zpracovávané snímky jsou menší. Nakonec je ještě přes zaškrtávací položku *Vyhladit přechody* možné nastavit zda bude při spojování snímků použita metoda Graph Cuts, pro minimalizaci viditelnosti přechodů a artefaktů.

Následně po stisku tlačítka **Spoj** začne samotná konstrukce panoramatu. Informace o průběhu jsou vypisovány do textového pole. Poté je výstup zkonstruován je zobrazen v samostatném okně. Toto okno může uživatel zobrazit sám po stisku tlačítka **Zobrazit spojený snímek** a tento snímek si může uložit pomocí tlačítka **Uložení spojeného snímku**. Také je po zkonstruování panoramatu možné z něho vytvořit výřez za pomoci tlačítka **Vytvořit výřez ze snímku**. Tento výřez je možné zobrazit tlačítkem **Zobrazit výřez ze snímku** a uložit tlačítkem **Uložit výřez snímku**.

Kdykoli, když neprobíhá výpočet, je možné pomocí tlačítka **Výběr souborů** vybrat nové vstupní snímky, nastavit parametry a popsaným způsobem vytvářet panoramata.

Příloha B

Obsah přiloženého DVD

Obsah přiloženého DVD je rozdělen do této adresářové struktury:

- *bin* - obsahuje přeložený program Stitcher.exe se všemi potřebnými knihovnami ke spuštění
- *datasets* - použité datové sady snímků s popisem
- *manual* - návod jak pracovat s realizovanou aplikací Stitcher.exe
- *panorama_out* - výstupní vytvořená panoramata implementovaného programu na datových sadách
- *plakatek* - vytvořený plakátek pro prezentaci diplomové práce
- *plakatek_src* - zdrojové soubory k plakátku (pdf lze otevřit v Adobe Illustrator)
- *src* - obsahuje zdrojové soubory aplikace z programu Microsoft Visual Studio 2013
- *text_pdf* - obsahuje tento text ve formátu PDF (verze s obrázky ve vysokém rozlišení)
- *text_src* - zdrojové soubory textu práce v LaTeXu