VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÁ REKTIFIKACE FOTOGRAFIÍ

AUTOMATIC RECTIFICATION OF PHOTOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — BORIS BURKALO
AUTHOR

VEDOUCÍ PRÁCE — Ing. ROMAN JURÁNEK, Ph.D.
SUPERVISOR

BRNO 2021

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

23284

Student: **Burkalo Boris**
Program: Informační technologie
Název: **Automatická rektifikace fotografií**
Automatic Rectification of Photos
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte metody pro automatickou korekci perspektivního zkreslení.
2. Navrhněte nebo modifikujte existující metodu.
3. Implementujte metodu a vyhodnoťte její přesnost.
4. Vytvořte prezentační materiály
5. Zhodnoťte vaši práci a navrhněte možnosti pokračování

Literatura:

- Chaudhury, Krishnendu, Stephen DiVerdi, and Sergey Ioffe. Auto-rectification of user photos. ICIP 2014
- Xue et al, Learning Attraction Field Representation for Robust Line Segment Detection, CVPR 2019
- Dubska et al, Real Projective Plane Mapping for Detection of Orthogonal Vanishing Points, BMVC 2013

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Juránek Roman, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem této práce je navrhnout způsob, kterým lze provést rektifikaci velkého množství fotografií. Práce se zaměřuje na vysvětlení pojmu rektifikace a popis kroků, které jsou nutné k provedení úspěšné rektifikace fotografie. Zároveň je práce zaměřena na popis jednotlivých metod pro rozpoznávání úseček nebo přímek v obraze a metod pro výpočet úběžníků, jsou-li přímky a úsečky již známé. Dále je v rámci práce implementován automatický rektifikátor fotografií a je zhodnocena přesnost jednotlivých použitých metod.

Abstract

The aim of this thesis is to propose a method for automatic rectification of larger sets of photos. This thesis is first aimed at describing the problem which is posed by rectification of photos and the description of individual steps that need to be taken to successfully rectify an input photograph. After that the thesis describes individual methods for line and vanishing point detection. Next the proposed method for automatic rectificatior is implemented and ultimately a statistic of used methods is created.

Klíčová slova

rektifikace fotografií, rektifikace obrazu, rozpoznání přímek, rozpoznání úseček, rozpoznání úběžníků, automatická rektifikace

Keywords

rectification of photos, line detection, line segment detection, vanishing point detection, automatic rectification

Citace

BURKALO, Boris. *Automatická rektifikace fotografií.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Roman Juránek, Ph.D.

Automatická rektifikace fotografií

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Romana Juránka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Boris Burkalo
9. května 2021

Poděkování

Tímto bych rád poděkoval Ing. Romanu Juránkovi, Ph.D. za rady a informace, které mi poskytl při řešení této práce a za vedení mé bakalářské práce.

Obsah

1 Úvod **2**

2 Rektifikace obrazu **3**
2.1 Metoda 4

3 Studované metody **6**
3.1 Detekce úseček 6
3.2 Detekce přímek 18
3.3 Detekce úběžníků 22
3.4 Datové sady pro rozpoznání čar 26

4 Návrh řešení **29**
4.1 Sjednocení anotací 29
4.2 Nalezení úseček nebo přímek 30
4.3 Nalezení úběžníků 31
4.4 Zpracování dat a výběr použitelných prvků 31
4.5 Finální rektifikace 31
4.6 Vyhodnocení výsledků 32
4.7 Použité technologie 32

5 Výsledky **34**
5.1 Rozpoznání úběžníků v obraze 34
5.2 Rektifikace vstupních fotografií 38

6 Závěr **43**

Literatura **45**

Kapitola 1

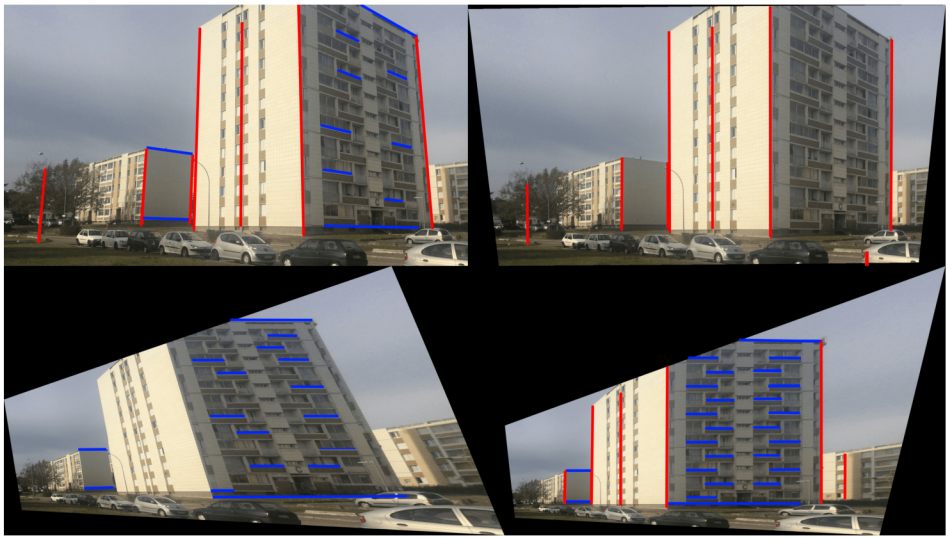
Úvod

Již dlouhodobě je zpracování obrazu důležitým problémem informačních technologií a informačních služeb. Ať už se jedná o kosmetickou manipulaci vstupního obrazu, nebo detekci konkrétních obrazců a předmětů v rámci sledovaného vstupního obrazu. Tato práce pokryje oba zmíněné aspekty zpracování obrazu, popíše proces rektifikace fotografie a navrhne postup pro vytvoření automatického rektifikátoru fotografií.
Proces rektifikace se snaží o to, aby výsledná fotografie měla všechny čáry v obraze rovnoběžné. Samotná rektifikace je hojně využívaná v *post-processingu* profesionálními nebo hobby fotografy a často je součástí programů pro úpravu fotografií. Detailněji je tento proces popsán v kapitole 2. K úspěšné rektifikaci fotografie je ale zapotřebí nalézt a vypočítat úběžníky fotografie. Ty budou v rámci práce vypočítávány z úseček nebo přímek nalezených v obraze. Dalším cílem této práce je tedy analyzovat a popsat jednotlivé metody pro nalezení přímek, úseček a úběžníků v obraze.
Nacházení čar v obraze je velice komplexní úkon, kterým se oblast počítačového vidění zabývá již velice dlouho. Proto existuje velké množství metod, které se liší v mnoha aspektech. Tyto metody, které jsou popsané v kapitole 3, se dělí podle jejich specifických postupů způsobu detekce. Mohou být rozděleny podle obrazce, který rozpoznávají, dle tohoto kritéria by mohly být metody popisované v této práci rozděleny na ty, které hledají v obraze úsečky a ty, které hledají přímky. Dalším kritériem pro rozdělení metod by mohlo být, zda je při nacházení obrazců používáno strojové učení, což jsou například metody HAWP [21], AFM [20], PPGNet [22] a Wireframe [15], nebo strojové učení k detekci nepoužívají, jako metody LSD [10], PCLines [8] a EDLines [1].
Práce očekává již nějaké předchozí znalosti na téma zpracování obrazu a pokročilé znalosti geometrie a matematiky. V kapitole 2 je nejdřív nastíněn způsob, jakým je možné rektifikovat vstupní fotografii a následně je uveden matematický postup, kterým lze vypočítat matice homografie pro afinní transformaci obrazu takovou, aby zajistila rovnoběžnost vertikálních čar ve vstupním obraze. V kapitole 3 jsou popsány jednotlivé metody, které je možné použít pro nalezení přímek nebo úseček a z nich i úběžníků. Při popisu těchto metod se práce drží pouze důležitých informací a základních postupů, pro specifické informace je možné nahlédnout do citovaných článků. Dále jsou v kapitole 3 přiblížena data, tedy fotografie, nad kterými bude navrhované řešení pracovat a která budou použita pro finální evaluaci rektifikátoru. Kapitola 4 se zabývá navrhovaným řešením, které bylo následně i implementováno. Zbývající kapitola 5 se zabývá výsledky dosaženými pomocí implementovaného rektifikátoru.

Kapitola 2

Rektifikace obrazu

Už déle se oblast počítačového vidění zabývá transformacemi obrazu za cílem zlepšení vstupní fotografie. Rektifikace obrazu není výjimkou a jejím hlavním cílem je transformovat obraz tak, aby všechny hrany, které byly v původní reálné scéně rovnoběžné, byly rovnoběžné i na výstupní fotografii. Výsledkem procesu rektifikace je fotografie, která má buď vertikální, horizontální nebo oba směry čar rovnoběžné a základní typy rektifikace mohou být podobně rozděleny na vertikální, horizontální a kombinaci těchto dvou. Příklad rektifikovaných fotografií je demonstrován na obrázku 2.1, těchto fotografií bylo dosaženo za pomocí webového nástroje[1]. Tato práce je však zaměřena pouze na rektifikaci ve vertikálním směru z důvodu lepšího využití na reálných fotografií.
Důležitým aspektem rektifikace fotografií a obrázků je také korektní rozpoznání přímek nebo úseček v obraze a výpočet úběžníků, tedy bodů ve kterých se přímky nebo úsečky v obraze scházejí. Metody pro provedení těchto kroků jsou uvedeny v následujících kapitolách.



Obrázek 2.1: Příklad jednotlivých typů rektifikace. Fotografie byly získány pomocí webového nástroje na rektifikaci fotografií. **Nalevo:** Fotografie po provedení rektifikace ve vertikálním směru. **Uprostřed:** Fotografie po provedení rektifikace v horizontálním směru. **Napravo:** Fotografie po provedení vertikální a horizontální rektifikace.

[1] www.imagerectifier.net

2.1 Metoda

Jsou-li tedy ze vstupní fotografie získány úsečky nebo přímky, z kterých jsou následně jedním z níže uvedených způsobů vypočítány úběžníky, je možné provést rektifikaci vstupního obrazu navrhovaným postupem, který je inspirován způsobem popisovaným v článku [5]. Tento postup tedy provede nad vstupním obrazem vertikální rektifikaci, čímž nastolí rovnoběžnost všech vertikálních čar.

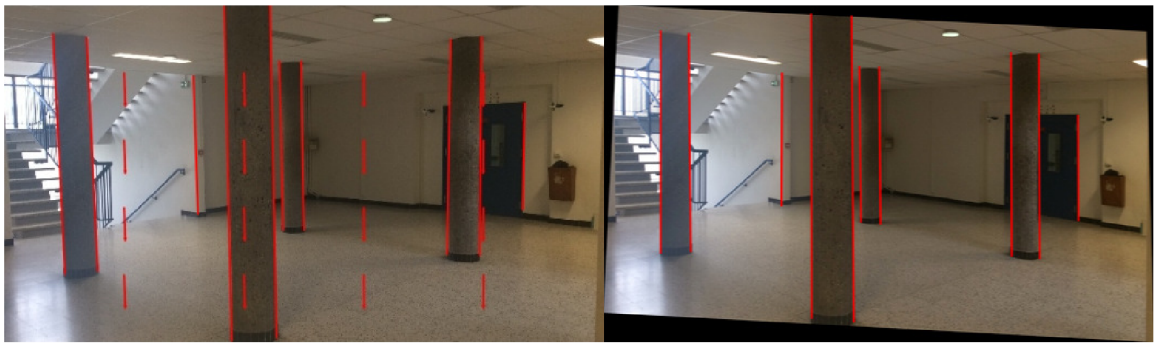
Nastolení rovnoběžnosti vertikálních čar: Jsou-li ve vstupním obraze nalezeny až tři úběžníky a souřadnice těchto úběžníků jsou převedeny do homogenních souřadnic, je možné pokračovat v hledání homografie, která je potřebná k transformaci vstupní fotografie.
Podle metody dokumentované v článku [5] musí pro hledanou homografii platit, že převede použitý úběžník zpět do nekonečna a tak nastolí rovnoběžnost všech vertikálních čar v obraze.
Je nutno podotknout základní charakteristiku pro bod a přímku v nekonečnu v systému homogenních souřadnic - bod $p = \begin{bmatrix} x & y & z \end{bmatrix}^T$ v nekonečnu v homogenních souřadnicích se vyznačuje tím, že má složku z nulovou a přímkou v nekonečnu je tedy přímka následujícího tvaru $l = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.
Nejdříve je pro sestrojení homografie nutné zjistit, který z nalezených úběžníků v_1, v_2, v_3 je úhlově nejblíž ose Y. Tyto nalezené úběžníky převedeme do homogenních souřadnic a vznikne nám vektor $\vec{v_i} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$. Osa Y je reprezentována vektorem $\vec{Y} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ a může být použit vzorec $\theta_i = \arccos(\frac{\vec{v_i} \cdot \vec{Y}}{|\vec{v_i}| * |\vec{Y}|})$, pomocí kterého lze získat úhel, který svírají dva vektory. Vektor úběžníku s nejmenším úhlem θ_i je hledaným úběžníkem, který je vybrán pro proces rektifikace. Následně je vypočítán $\vec{v}_h = \begin{bmatrix} -\vec{v}_{v,y} & \vec{v}_{v,x} & 0 \end{bmatrix}^T$, což je pomocný bod v nekonečnu, který je kolmý na nalezený úběžník v_v.
Jsou-li $\vec{v}_v$ a $\vec{v}_h$ známé, je potřeba sestrojit vektor přímky, která tyto dva body spojuje. V homogenních souřadnicích je tato přímka definována vektorovým součinem vektorů dvou bodů, tedy $\vec{l}_v = \vec{v}_v \times \vec{v}_h$. Spojením všech získaných informací je možné sestrojit matici demonstrovanou rovnicí 2.1. Pomocí této matice se nastolí fronto-paralelita vstupní fotografie.

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vec{l}_x & \vec{l}_y & \vec{l}_z \end{bmatrix} \tag{2.1}$$

Zarovnání vertikálních čar s osou Y: Je-li nalezena matice H, $\vec{v}_v$ a jemu příslušící úhel θ_i, je možné sestrojit rotační matici, která zajistí otočení transformované fotografie tak, aby všechny, teď již paralelní, vertikální čáry, byly také paralelní s osou Y. Tuto rotaci zajistí matice 2.2.

$$R = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Výslednou homografii pro transformaci vstupní fotografie získáme součinem dvou výše nalezených matic a výsledná matice tedy má tvar $T = RH$. Příklad vertikálně rektifikované fotografie je demonstrován na obrázku 2.2.



Obrázek 2.2: Demonstrace procesu rektifikace fotografie. **Nalevo:** Původní fotografie, ve které jsou vyznačeny vertikální úsečky (pro jednoduchost jsou vyznačené ručně a nejsou vyznačené všechny) a šipky, které směřují k vertikálnímu úběžníku v_v. **Napravo:** Rektifikovaná fotografie s rovnoběžnými vertikálními úsečkami (vyznačeny jsou také pouze některé vertikální úsečky).

Kapitola 3

Studované metody

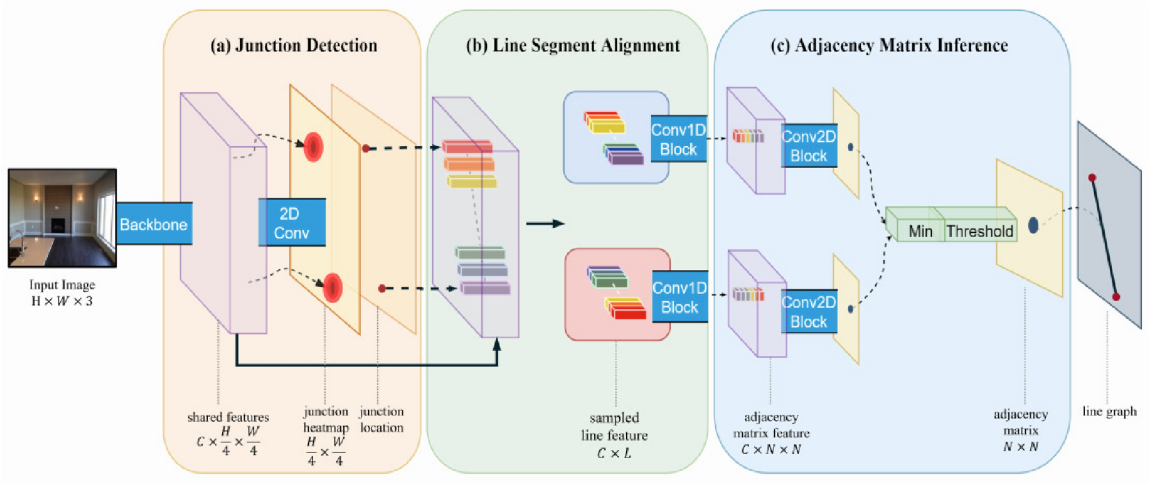
Tato kapitola se bude věnovat jednotlivým metodám, které byly studovány pro účely práce a v rámci navrhování řešení pro nalezení úseček nebo přímek v obraze a úběžníků v obraze. Nacházení a rozpoznání obrazců, v tomto případě úseček a přímek, ve fotografiích a obrázcích je v oblasti počítačového vidění často probírané a problematické téma. Existuje velké množství metod, které jsou uzpůsobené k nacházení přímek, úseček nebo úběžníků ve vstupních fotografiích a obrázcích, v této kapitole budou vysvětleny principy jejich fungování. Je také důležité poznamenat, že pro standardizované porovnání těchto metod je nutné použít standardizovaná data, o těchto datech se zmíním v podkapitole 3.4.

3.1 Detekce úseček

Jedním z nejcharakterističtějších znaků jak rozdělit metody pro detekci čar v obraze je podle obrazce, který rozpoznávají. Tato podkapitola se bude zabývat rozpoznáním úseček, tyto metody jsou zpravidla složitější něž metody, které hledají přímky a často mají větší nároky na zdroje. Metody pro rozpoznání úseček by se daly rozdělit podle dvou typických charakteristických znaků, tedy metody využívající strojového učení k dosažení výsledků, těmi se zabývá první část podkapitoly, a metody které strojové učení nevyužívají, tyto metody jsou popsány ve zbývající části podkapitoly.

3.1.1 PPGNet

Tato metoda vyvinuta a implementována Zhang et al. [22] je metodou pro rozpoznání úseček v obraze. Nezaměřuje se ale pouze na úsečky, věnuje se také rozpoznání průsečíků mezi jednotlivými úsečkami. Dala by se zařadit do první z kategorie popisovaných metod tím, že využívá strojové učení - neuronových sítí - pro rozpoznávání prvků v obraze.
Celkový návrh metody je možné rozdělit na čtyři části. První částí je páteřní síť metody. Popis této části bude však vynechán, jelikož není pro pochopení metody nutný. Druhou částí je modul pro detekci průsečíků úseček, třetí část zastává pozici rozpoznání všech úseček v obraze a čtvrtá se zaměřuje na nacházení spojení jednotlivých průsečíků. Na obrázku 3.1 je znázorněna architektura metody PPGNet [22].



Obrázek 3.1: Architektura metody PPGNet [22]. Převzato z článku [22].

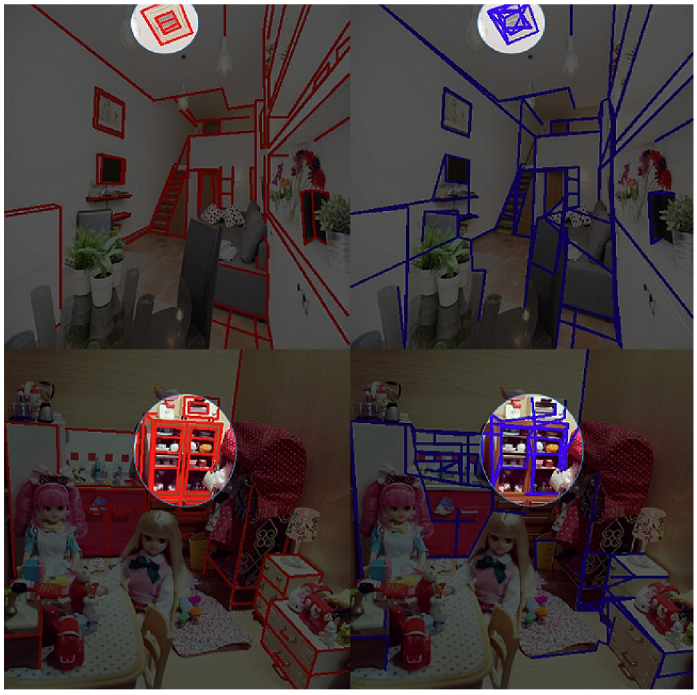
Modul pro detekci průsečíků zajistí rozpoznání uzlových bodů v obraze. Průsečíky dostane tak, že regresuje teplotní mapu průsečíků ze získaného prvku pomocí páteřní sítě a následně určí všechny body, u kterých je pravděpodobnost, že jsou validními průsečíky větší, než práh τ a zároveň je pravděpodobnostní hodnota největší v rámci osmi sousedních bodů. Následně dojde ke sdružení bodů do shluků, ve kterých platí, že každé libovolně vybrané body mají mezi sebou menší vzdálenost než ϵ (podle článku [22] $\epsilon = 3$). Body s největší pravděpodobností v rámci shluku jsou označeny jako průsečíky.

Modul pro rozpoznání úseček pro každý pár průsečíkových bodů a mapu prvků vygeneruje tenzor prvků o velikosti $C \times L$, kde C je počet kanálů mapy prvků a L je prostorová délka prvku úsečky. Konkrétně tento modul vygeneruje L bodů o rovnoměrné vzdálenosti od jednoho nalezeného průsečíku, ke druhému. V rámci článku [22] je L fixně nastaveno na hodnotu 64, takže každý průsečný pár generuje tenzor o velikosti $C \times 64$.

Modul pro odvození matice přiléhavosti předpoví spojení každé kombinace průsečíkových bodů v rámci vstupní fotografie. Čerpá z prvků poskytnutých předchozím modulem a používá konvoluční strukturu pro určení pravděpodobnosti, že jsou dva průsečíky spojeny.

Výsledky dosažené metodou PPGNet [22] můžeme vidět na obrázku 3.2. Autoři se také v článku [22] zmiňují o známých chybách, kterých jejich metoda dosahuje. Typicky tyto chyby nastávají pro malé obdélníky ve vstupním obraze, nebo pokud se v obraze nachází velice blízké kolineární úsečky. Tyto chyby jsou demonstrované na obrázku 3.3.

Obrázek 3.2: Nalezené úsečky pomocí metody PPGNet [22]. Převzato z [22]. **První řada:** Anotované hodnoty pro Wireframe [15] datovou sadu. **Druhá řada:** Metodou nalezené hodnoty pro Wireframe [15] datovou sadu. **Třetí řada:** Anotované hodnoty pro York Urban Database [6]. **Čtvrtá řada:** Metodou nalezené hodnoty pro York Urban Database [6].



Obrázek 3.3: Demostrace chyb metody PPGNet [22] při blízkých kolineárních úsečkách nebo malých obdélnících ve fotografii. Převzato z [22]. **Červeně vyznačené:** Anotované úsečky. **Modře vyznačené:** Nalezené úsečky.

3.1.2 Hledání úseček pomocí Attraction Field Map

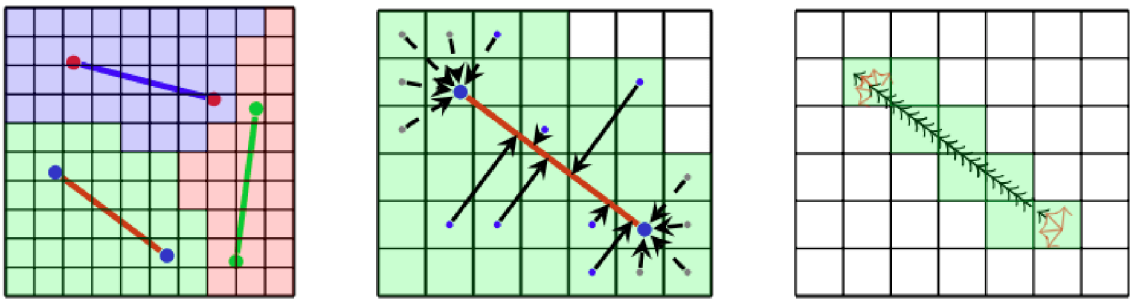
Tato metoda (dále jen AFM) vyvinuta a implementována Xue et al. [20] je jednou z dalších metod pro rozpoznání úseček v obraze. Podobně jako metoda předchozí také používá strojové učení pro dosažení co nejpřesnějších výsledků. Je ideální pro použití v mnou řešeném problému, avšak problémem této metody je, že potřebuje již anotované hodnoty k rozpoznání dalších hodnot.
Máme-li tedy anotované úsečky, můžeme každou úsečku reprezentovat svým geometrickým modelem, který je vypočítán z jejích koncových bodů. Pro reprezentaci úsečky navrhovanou metodou jsou potřeba tři kroky. V prvním je vstupní obraz rozdělen na oblasti na základě anotovaných úseček. Druhým krokem je sestavení tzv. *attraction field map*. Posledním krokem metody je sestrojení tzv. *squeeze* modulu.

Rozdělení vstupního obrázku na oblasti je provedeno tak, že každý pixel vstupního obrazu je přidělen vždy a pouze jen jedné úsečce. Pixely přidružené k jednotlivým úsečkám tvoří jednotlivé oblasti, tyto oblasti se nepřekrývají. Tímto způsobem vytvoří mapu oblastí, která je znázorněna na obrázku 3.4. Zjednodušený postup je možné vidět na obrázku 3.5.

Sestrojení attraction field map tkví v nalezení projekčního bodu na úsečce pro každý pixel ležící v oblasti úsečky, využívá tedy informace získané v předchozím komponentu a aplikuje je. Každý pixel má na úsečce pouze jeden projekční bod, nalezneme-li tento bod, můžeme každý pixel reprezentovat jeho projekčním vektorem, ležícím mezi pixelem a projekčním bodem. Attraction field map můžeme vidět na 3.4, zjednodušená verze pro pouze jednu úsečku je na 3.5.

Obrázek 3.4: Reprezentace úseček pomocí oblastí. Převzato z [20]. **Nalevo:** Původní reprezentace úseček. **Uprostřed:** Reprezentace úseček pomocí oblastí. **Napravo:** Attraction field map

.



Obrázek 3.5: Zjednodušená reprezentace úseček podle popisované metody. Převzato z [20]. **Nalevo:** Výpočet oblastí. **Uprostřed:** Výpočet *attraction* vektorů. **Napravo:** Squeeze modul.

Squeeze modul využívá informací získaných v předchozích krocích a pomocí nich nalezne úsečky co nejpodobnější těm, které byly v procesu na začátku.
Máme-li tedy A (odpovídající attraction field map), převrátíme hodnoty vypočítáním reálné hodnoty průmětu pro každý pixel p a pro jeho korespondující diskrétní bod. Poté je nutno vypočítat mapu s návrhem korespondujících úseček, ve které jsou ke každému pixelu q vybrány vektory z attraction field množiny, jejichž diskretizované body odpovídají q.
Zjednodušeně lze algoritmus popsat tak, že je náhodně vybrán pixel q a jeden vektor $a(p)$ z jeho pole kandidátních *attraction* vektorů. Následně je prohledáno pole o velikosti 3×3 kolem pixelu q, zda-li se v něm nachází nějaký pixel, jehož vektor z attraction field vektorů by byl úhlově zarovnaný s $a(p)$ tak, že úhel mezi nimi je menší než práh τ (pro článek [20] $\tau = 10°$). Pokud takovýto pixel není nalezen, je $a(p)$ odstraněn z pole kandidátních vektorů (je-li po odstranění pole prázdné, je odstraněn i pixel q). Pokud je nalezen, prohledáváme dále pixel q a průměrujeme jeho směr pomocí nově nalezeného vektoru. Zarovnané *attraction* vektory jsou označené jako použité. Tento algoritmus je tímto způsobem rekurzivně aplikován. Jestliže jsou vyčerpány všechny pixely, dostaneme navrhovanou úsečku, která je ověřena a následně označena jako validní. Příklad lze vidět na obrázku 3.5.

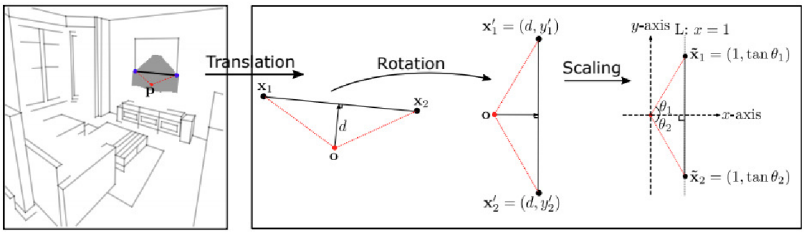
Výsledky rozpoznání úseček lze vidět na obrázku 3.6. Metoda dosahuje velice přesných výsledků, avšak často dochází ke shluku náhodných nesmyslných úseček.

Obrázek 3.6: Rozpoznání úseček pomocí metody AFM [20]. **Nalevo:** Vzorek z datové sady Wireframe [15]. **Uprostřed:** Vzorek z York Urban Database [6]. **Napravo:** Vzorek z Toulouse Vanishing Point Dataset [2].

3.1.3 Holistically-Attracted Wireframe Parsing

Je metoda (dále jen HAWP) pro rozpoznání úseček a jejich průsečíků v obraze od Xue et al. [21]. Podobně jako předchozí metody, HAWP používá strojového učení za cílem rozpoznání úseček a průsečíků. Tato metoda se skládá ze tří základních kroků. Těmito kroky jsou: rozpoznání úseček a průsečíků, přiřazení průsečíků k úsečkám a verifikace detekovaných prvků.
Důležitými složkami, které tato metoda využívá jsou pole A, J, F a O.
Pole A neboli *holistic attraction field map*, je pole o velikosti $V \times S$ (V značí výšku vstupního obrazu, S značí šířku) obrázku I, kde jsou všechny úsečky převedeny do 4-D tvaru. Tohoto 4-D tvaru dosáhneme tak, že nad každým pixelem v podpůrné oblasti úsečky (na obrázku 3.7 vyznačena šedě) v obraze provedeme tři afinní transformace. Těmito transformacemi jsou: translace, rotace a změna velikosti úsečky způsobem znázorněným na obrázku 3.7. Každý pixel $p(\ddot{l})$ je tedy potom parametrizován v rámci této podpůrné oblasti pomocí 4-D vektoru 3.1, kde d je vzdálenost mezi pixelem $\mathbf{p}$ a úsečkou $\ddot{l}$ v podpůrné oblasti úsečky a zbylé souřadnice odpovídají proměnným uvedeným na obrázku 3.7. Tyto vektory jsou následně normalizovány a dohromady tvoří pole A. Pole J a O obsahují informace o koncových bodech a průsečících úseček.

$$p(\ddot{l}) = (d, \theta, \theta_1, \theta_2) \tag{3.1}$$



Obrázek 3.7: Ilustrace převedení úseček do 4-D formy.

HAWP [21] používá pro extrakci prvků ze vstupních obrázků Hourglass Network [17], což je konvoluční neuronová síť používaná pro rozpoznání pozic lidského těla, je však také často používána pro rozpoznání rohových oblastí v obraze. Tato neuronová síť pro vstupní snímek nalezne prvky, které jsou následně použity pro rozpoznání úseček a průsečíků. Prvky jsou vyznačeny v poli F o stejné velikosti jako předchozí zpracovávaná pole.

Predikce úseček a průsečíků začíná predikcí 4-D AFM pole z pole F získaného pomocí neuronové sítě. Je-li tedy $\hat{A}$ predikované 4-D pole podle výše popsaného postupu, tak navrhované úsečky dostaneme obrácením normalizace a afinních transformací. Průsečíky jsou navrhnuty pomocí vytvoření polí J a O z pole F.

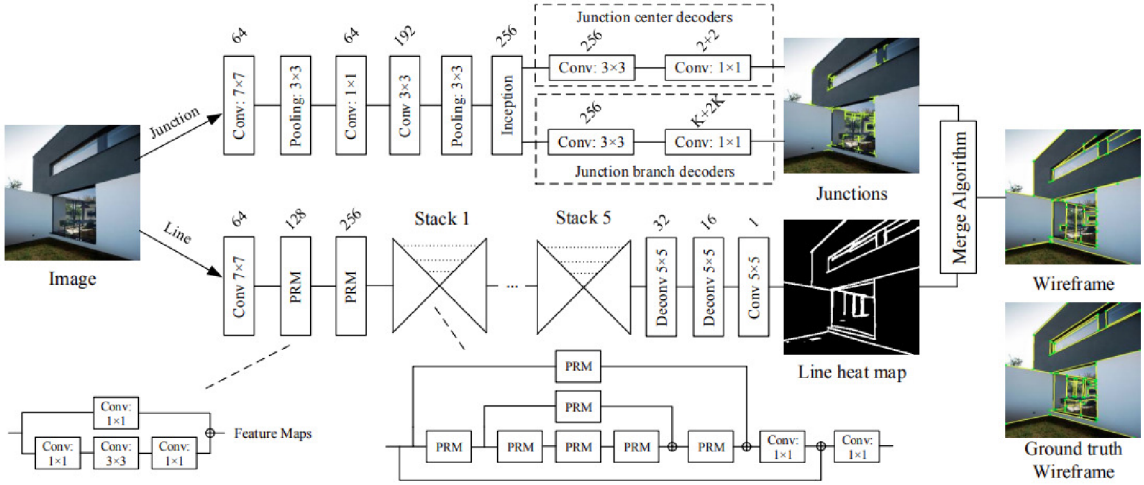
Přiřazení průsečíků k úsečkám probíhá pomocí extrakce různých informací získaných v předchozím kroku. Úsečka je ponechaná pouze v případě, je-li možné přiřadit její dva koncové body ke dvěma nalezeným průsečíkům na základě jejich Euklidovské vzdálenosti s předdefinovaným prahem τ (podle článku [21] $\tau = 10$). Naopak průsečík je ponechán pouze v případě, kdy je možné ho přiřadit k nějakým z ponechaných úseček.

Výsledky detekce úseček lze pozorovat na obrázku 3.8. Je vidět, že v porovnání s předchozí metodou od stejných autorů, HAWP [21] dosahuje o poznání lepších výsledků. Také se ve zpracovaných obrázcích nevyskytují náhodné shluky úseček, které by kazily následné zpracování.

Obrázek 3.8: Rozpoznání úseček pomocí metody HAWP [21]. **Nalevo:** Vzorek z datové sady Wireframe [15]. **Uprostřed:** Vzorek z York Urban Database [6]. **Napravo:** Vzorek z Toulouse Vanishing Point Dataset [2].

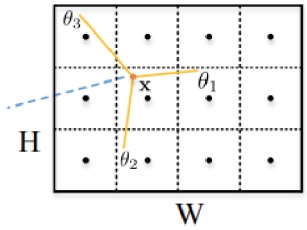
3.1.4 Wireframe parser

Tato metoda od Huang et al. [15] je další z metod, které používají neuronové sítě pro rozpoznání úseček v obraze. Kromě úseček je tato metoda také schopná rozpoznat průsečíky úseček v obraze stejně jako metoda PPGNet [22]. Na obrázku 3.9 je znázorněna architektura metody. Zároveň byla pro zhodnocení této metody vytvořena velice obsáhlá datová sada, která je zmíněna v podkapitole 3.4.



Obrázek 3.9: Architektura systému. Převzato z [15]. **Nahoře:** Detekce průsečíků. **Dole:** Detekce přímek.

Za rozpoznání průsečíků je zodpovědná plně konvoluční síť (anglicky *fully convolutional network* nebo také *FCN*), která zkoumá celý vstupní obraz a dokáže v něm tedy lépe průsečíky najít. Implementovaná konvoluční síť rozdělí vstupní obraz na pole o velikosti $H \times W$ (H je výška a W je šířka vstupního obrazu), jak je demonstrováno na obrázku 3.10. Za průsečík je tedy zodpovědná ta buňka, do které průsečík spadá. Každá ij-tá buňka vyprodukuje skóre c_{ij}, kterým značí, jak moc si je jistá, že do ní spadá nějaký průsečík.



Obrázek 3.10: Schematická ukázka pole vstupního obrazu. Převzato z [15].

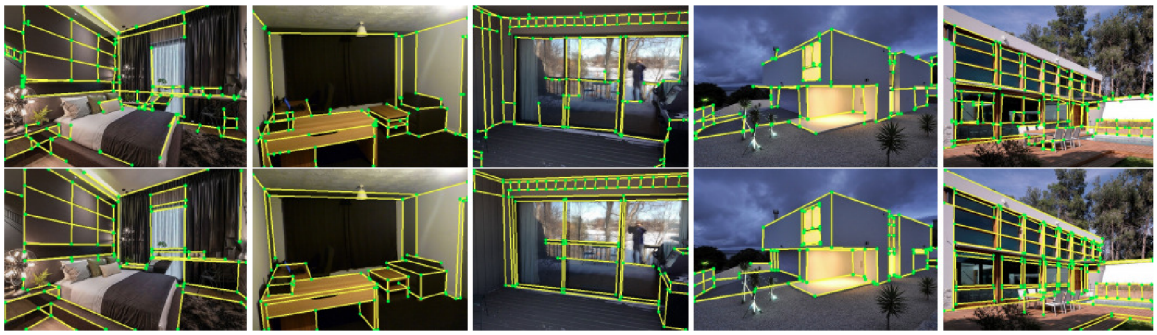
Pro rozpoznání úseček byla vytvořena konvoluční neuronová síť, která je schopná získat informace o úsečkách z RGB obrázků. Neuronová síť předpoví pro každý pixel, zda leží na nějaké přímce l. Pro potlačení lokálních hran, krátkých úseček nebo křivek je předpovězená hodnota $h(p)$, která bude zaznamenaná do teplotní mapy, nastavena na délku úsečky, ke které patří daný pixel p. Máme-li tedy obrázek s úsečkami L, tak výsledné hodnoty $h(p)$ jsou definovány jako:

$$h(p) = \begin{cases} d(l) & p \text{ je na přímce } l \text{ v } L \\ 0 & p \text{ není na přímce } l \text{ v } L \end{cases} \tag{3.2}$$

kde $d(l)$ je délka úsečky l.

Spojení nalezených průsečíků a úseček a vytvoření finálního W (W = *wireframe*) pro obrázek složený z průsečíků P, které jsou spojeny úsečkami z množiny úseček L, jsou posledními kroky systému.
Je-li tedy nalezena množina průsečíků $\{p_i\}_{i=1}^{N}$ a teplotní mapa úseček h, aplikujeme práh w pro převedení h do binární mapy $\mathcal{M}$. *Wireframe* W je sestrojen tak, že jsou náhodně vybrány dva průsečíky z množiny P. Tyto dva průsečíky jsou spojeny přímkou $l = (p, q)$ pouze, pokud jsou umístěny přímo na, nebo velice blízko u té samé větve (větev je v kontextu této metody úsečka, pro kterou je bod p průsečík s nějakou jinou úsečkou, bod p může mít tedy více větví). Pokud dojde k nalezení více průsečíků na jedné větvi průsečíku p, je ponechána pouze úsečka s nejmenší délkou, aby nedocházelo k překrývání.
Pro jakoukoliv větev průsečíku p, pro kterou nebyl nalezen druhý průsečík, je hledána alternativní úsečka pomocí $\mathcal{M}$. Nejprve je nalezen nejzazší pixel $q_{\mathcal{M}}$, který vyhovuje podmínce $\mathcal{M}(q) = 1$ a zároveň leží na paprsku, který začíná v průsečíku p a směřuje podél popisované větve. Potom jsou nalezeny všechny průsečíky $\{q_1, \cdots, q_S\}$ úsečky (p, q_m) s existujícími úsečkami v L. Je-li tedy $q_0 = p_i$ a $q_{S+1} = q_{\mathcal{M}}$, tak je vypočítána proměnná $\mathcal{K}$ úsečky. $\mathcal{K}$ značí poměr počtu přímkových pixelů vůči celkové délce úsečky. Pokud je $\mathcal{K}$ větší než práh 0.6, tak přidáme úsečku do L a průsečík do P.

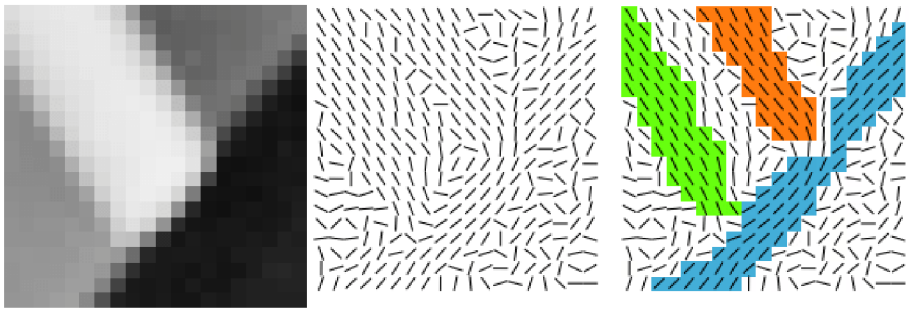
Výsledky metody Wireframe [15] je možné pozorovat na obrázku 3.11.



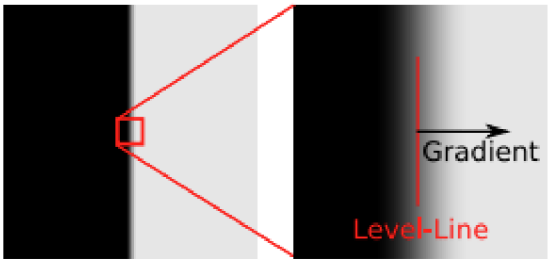
Obrázek 3.11: Výsledky dosažené pomocí metody Wireframe [15]. Převzato z [15]. **První řada:** Výsledné úsečky. **Druhá řada:** Anotované hodnoty.

3.1.5 LSD

Cílem LSD od Gioi et al. [10] je rozpoznání kontur v obrázcích. Kontury v tomhle případě značí hrany, mezi kterými dochází k rychlejší změně odstínu barvy, jak demonstrováno na obrázku 3.12.
Prvním krokem pro dosažení úseček v obraze je, že je vypočítán vektor k hladinové linii pro každý pixel tak, že vznikne *Level-line* pole - tedy pole jednotkových vektorů takové, že v něm jsou všechny vektory tečné k hladinové linii procházející jejich počátkem jak je ukázáno na 3.12.
Následně je toto pole seskupeno do oblastí vektorů se stejným hladinovým úhlem v rozmezí nějaké tolerance τ. Oblasti takto seskupených vektorů se nazývají *Line-support* oblasti.

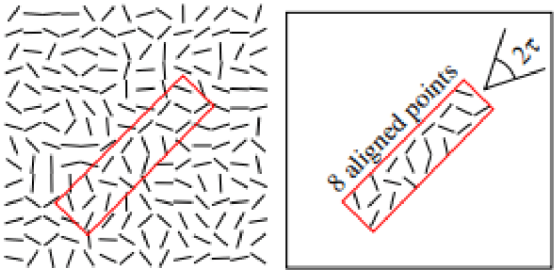


Obrázek 3.12: Detekce hran z obrazu. Převzato z [10]. **Nalevo:** Původní obrázek. **Uprostřed:** *Level-line* pole. **Napravo:** *Line-support* oblasti.



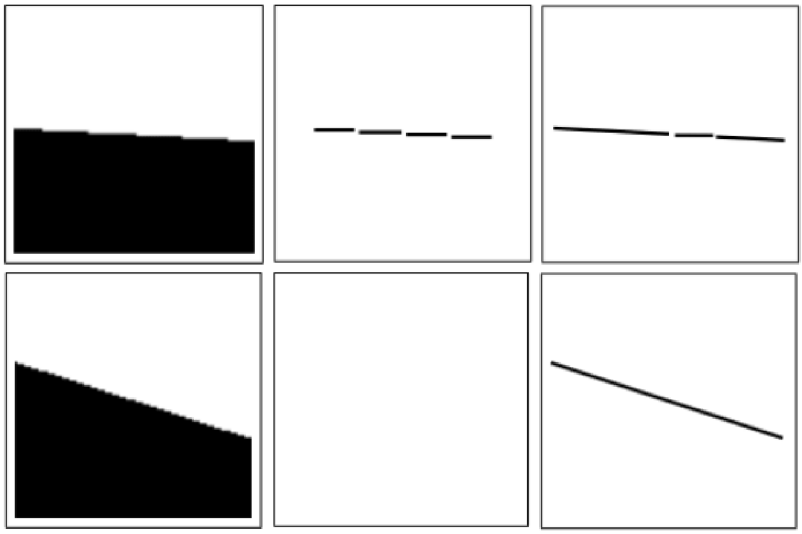
Obrázek 3.13: Ilustrace hladinové linie. Převzato z [10].

Každá z těchto oblastí (množin pixelů) je kandidátem na úsečku a musí jí být připsán náležitý geometrický útvar. V popisované metodě se jedná o obdélník. Každý z obdélníků musí být validován. Pixely v obdélnících jsou seskupeny do tzv. sdružených bodů, tyto sdružené body můžeme pozorovat na obrázku 3.14. Celkový počet pixelů v obdélníku je označen jako n a celkový počet sdružených bodů jako k. Tyto body jsou potom spočítány a použity k validaci, zda je segment přímky validní či nikoliv.



Obrázek 3.14: Příklad sdružených bodů. Převzato z [10].

Algoritmus metody je rozdělen na sedm podúloh. Je-li na vstupu obraz ve stupních šedi, je nutné obraz zmenšit. Tím, že se vstupní obraz zmenší na 80% jeho původní velikosti, jsou eliminovány potenciální chyby, které by nastaly v důsledku aliasingu nebo kvantování. Stejný efekt by mělo i rozmazání vstupního obrazu. Rozdíly detekce ilustruje obrázek 3.15.



Obrázek 3.15: Rozdíly při detekci hran. Převzato z [10]. **Nalevo:** Původní obraz. **Uprostřed:** Výsledky při vynechání zmenšení. **Naprav:** Výsledky při provedení zmenšení.

Dále je potřeba vypočítat gradient podle vzorců navrhnutých v článku [10].
Pokud jsou vypočítané gradienty pro jednotlivé pixely, je nutno tyto gradienty nějak uspořádat. Jelikož je LSD [10] hladovým algoritmem, je důležité, v jakém pořadí se nalezené gradienty jednotlivých pixelů zpracovávají. Z tohoto důvodu jsou pixely rozděleny do 1024 stejně velkých intervalů podle hodnoty jejich gradientu. Metoda LSD takto zpracovává pixely po vytvořených intervalech od největšího intervalu po nejmenší.
Jsou-li gradienty seřazené, je nutné zjistit, které navrhované gradienty jsou validní a které

ne. Je tedy nastaven práh ρ a pokud má pixel menší velikost gradientu než ρ, tak je vyřazen. Dále je na každý nepoužitý pixel z uspořádaného seznamu pixelů aplikován algoritmus pro formování line-support oblasti. Rekurzivně jsou vyhodnocovány nepoužité sousední pixely pixelů v line-support oblasti a ty, které spadají pod práh τ kolem oblastního úhlu θ jsou přidány do oblasti. Pokaždé, když je nějaký pixel do oblasti přidán, je hodnota úhlu θ aktualizovaná.
Dále je potřeba aproximovat obdélníky, které odpovídají jednotlivým oblastem. Tento obdélník je aproximován jako nejmenší obdélník, který je schopný pokrýt celou oblast. Zbývá už jen validace vybraných obdélníků a vyřazení obdélníků, které nejsou validní.

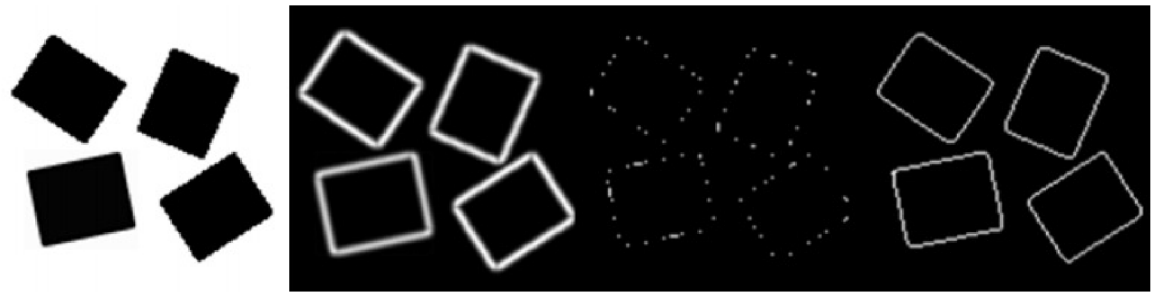
Úsečky nalezené pomocí metody LSD [10] je možné vidět na obrázku 3.16. V porovnání s předchozími, více komplexními metodami je evidentní, že tato metoda není tolik přesná a často nalezne úsečky, které by lidské oko za úsečku neoznačilo. Její výhodou však je její malá náročnost na zdroje a její rychlost.

Obrázek 3.16: Nalezené úsečky pomocí metody LSD [10]. **Nalevo:** Vzorek z datové sady Wireframe [15]. **Uprostřed:** Vzorek z York Urban Database [6]. **Napravo:** Vzorek z Toulouse Vanishing Point Dataset [2].

3.1.6 EDLines

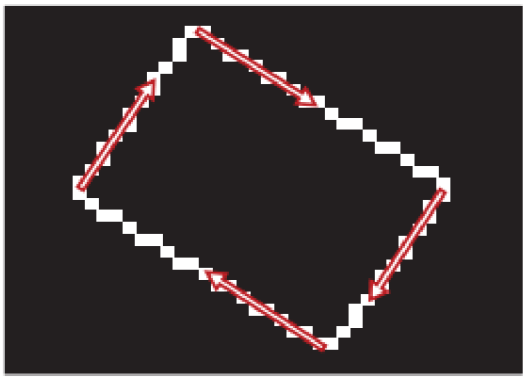
Tato metoda od Akinlar et al. [1] je založená na detekci hran v obraze, z kterých následně utvoří úsečky. V rámci této metody byl vyvinut i nový typ detekce hran v obraze. Výhodou této metody je, že je relativně rychlá a nenáročná. K dosažení úseček dojde již po dvou krocích, kterými jsou: detekce hran a nalezení úseček. Po těchto dvou krocích následuje už jen ověření, zda jsou úsečky validní.

Detekce hran pomocí kreslení hran (z názvu metody *Edge Drawing*) je prvním krokem k nalezení úseček. Na vstupní obraz ve stupních šedi je nejdříve aplikován Gaussův filtr pro potlačení šumu a vyhlazení obrazu. Následně je pro každý pixel vypočítána velikost a směr gradientu, kterých lze dosáhnout pomocí libovolného operátoru (např. Sobelova [18]). Dále dojde k výpočtu pixelů, u kterých je největší pravděpodobnost, že jsou hranové. Jsou to tedy pixely, které po aplikaci hranových operátorů dosahují největších hodnot. V posledním kroku dojde ke spojení všech vybraných hranových pixelů. Demonstrace jednotlivých kroků je zobrazena na obrázku 3.17.



Obrázek 3.17: Jednotlivé kroky metody EDLines [1] pro nalezení hran. Převzato z [1]. **1. zleva:** Původní obrázek ve stupních šedi. **2. zleva:** Mapa gradientu. **3. zleva:** Vybrané hranové body s nejvyšší pravděpodobností. **4. zleva:** Výsledné nalezené hrany.

Nalezení úseček probíhá analyzací sousedních hranových pixelů. Hledání úseček je realizováno tak, že se prochází celá sekvence sousedních hranových pixelů a pomocí metody nejmenších čtverců určujeme, zda pixel ještě patří na úsečku, nebo jsme našli úsečku novou. Stanovíme si tedy práh a pokud je tento práh překročen, sousední pixel se nachází na nové úsečce, pokud překročen není, pixel se nachází na úsečce aktuální.



Obrázek 3.18: Ilustrace hledání úseček. Převzato z [1].

Výsledky dosažené pomocí EDLines jsou znázorněny na obrázku 3.19. Z obrázků je evidentní, že metoda není tou nejpřesnější z prezentovaných metod, její velikou výhodou je však rychlost, jelikož zvládá zpracovávat vstupní obrázky až 11x větší rychlostí než metoda LSD [10].

Obrázek 3.19: Výsledky dosažené pomocí metody EDLines [1]. Převzato z [1].

3.2 Detekce přímek

V této podkapitole bude popsáno fungování metod pro detekci přímek v obraze. Tyto metody jsou však zpravidla nejefektivnější na vstupních obrázcích, které jsou jednoduché a dosahují nepřesných výsledků na složitějších fotografiích z reálného světa. I přesto pro větší robustnost práce byly některé z těchto metod použity pro implementaci práce.

3.2.1 Houghova transformace

Tato metoda je jednou z nejstarších metod (její originální forma) pro rozpoznání přímek v obraze. Původní verze této metody navrhnuta Paulem Houghem [14] používá standardní rovnici lineární funkce 3.3. Tato metoda však pro pozdější použití nebyla vhodná a byla tedy modifikována. V modifikované verzi používané dodnes a navržené Duda et al. [9] je využíváno parametrické nebo-li normálové vyjádření přímky podle rovnice 3.4, které vyjadřuje přímku pomocí úhlu θ její normály a pomocí hodnoty ρ, značící vzdálenost od počátku. Tato metoda tedy transformuje hodnoty v kartézském $x - y$ prostoru do parametrického prostoru $\theta - \rho$.

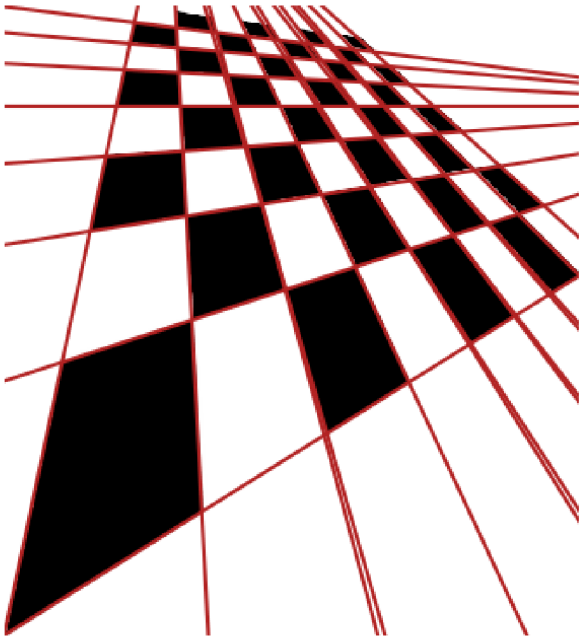
$$y = a * x + b \tag{3.3}$$

$$x \cos(\theta) + ysin(\theta) = \rho \tag{3.4}$$

Pokud omezíme úhel ρ na interval $[0, \pi\rangle$, budou normálové parametry pro přímku unikátní a tedy každá přímka v prostoru $x - y$ bude odpovídat jednomu bodu v prostoru $\theta - \rho$.
V praxi je používána Houghova transformace pro detekci přímek v kombinaci s nějakou z metod pro detekci hran v obraze - jmenovitě Sobelovou [18] nebo Cannyho [4]. Tyto

metody najdou hranové body v obraze a nalezené body jsou použity pro Houghovu transformaci tak, že jsou převedeny do jejich sinusoidní podoby v $\theta - \rho$ prostoru pomocí rovnice 3.4.
Je potom zřejmé, že křivky náležící bodům mají společný průsečný bod. Tento bod v $\theta - \rho$ prostoru definuje přímku v prostoru $x - y$, která protíná dva kolineární body.

Výsledky detekce přímek pomocí Houghovy transformace [9] převážně dávají smysl pro jednoduché obrázky jako je například 3.20. Pro fotografie reálného světa ale často nalezne spoustu přímek na místě, kde by byla pouze jedna. Fotografie z reálného světa zpracované touto metodou můžeme vidět na 3.21.



Obrázek 3.20: Ukázka Houghovy transformace [9] na jednoduchém obrázku. Původní obrázek převzat z [11].

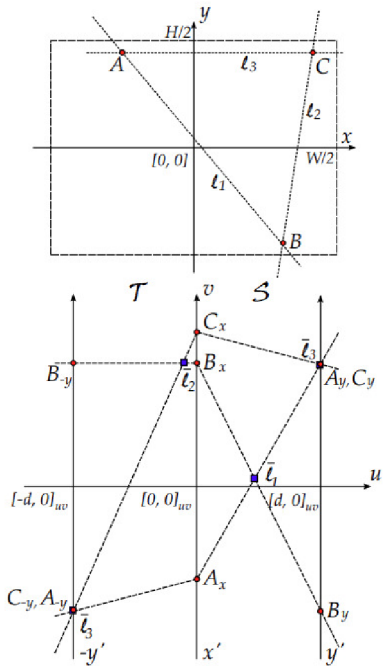
Obrázek 3.21: Ukázka Houghovy transformace [9] pro fotografie z reálného světa. **Nalevo:** Vzorek z Toulouse Vanishing Point Dataset [2]. **Uprostřed:** Vzorek z York Urban Database [6]. **Napravo:** Vzorek z datové sady Wireframe [15].

Progresivní pravděpodobnostní Houghova transformace popsaná J. Matasem et al. [16], je modifikací Houghovy transformace [9], která má za cíl zmenšení časové náročnosti původní metody.
V rámci modifikace dochází k náhodnému výběru jednoho pixelu ze vstupního obrazu, který následně volí do akumulátoru hlasů. Dále je odstraněn ze vstupního obrázku a následuje kontrola, zda-li nejvyšší hodnota v akumulátoru hlasů přesahuje práh $thr(N)$, pokud ano, metoda pokračuje dál, pokud ne, jsou tyto kroky zopakovány dokud nebude podmínka splněna. Pokud podmínka splněna byla, je nalezena nejdelší čára, která je spojitá, nebo jejíž mezery nepřekračují práh. Pixely ležící na úsečce jsou odstraněny ze vstupního obrazu a z akumulátoru hlasů jsou odstraněny hlasy pro nejvyšší hodnotu. Pokud je úsečka delší než stanovené minimum, je označena jako platná. Proces se opakuje dokud obraz není prázdný. Na obrázku 3.22 je možné vidět fotografie zpracované pomocí této metody.

Obrázek 3.22: Úsečky nalezené pomocí metody Progresivní pravděpodobnostní Houghovy transformace [16]. **Nalevo:** Vzorek z Toulouse Vanishing Point Dataset [2]. **Uprostřed:** Vzorek z York Urban Database [6]. **Napravo:** Vzorek z datové sady Wireframe [15].

3.2.2 PCLines

Metoda PCLines od Herout et al. [8] je založená na dvou již existujících metodách pro práci s počítačovou grafikou. Těmi jsou Houghova transformace [9] a systém rovnoběžných souřadnic [13].
Systém rovnoběžných souřadnic reprezentuje vektorový systém pomocí os, které jsou na sebe vzájemně rovnoběžné a každý vektor o N dimenzích je reprezentován $N-1$ přímkami, které spojují osy.
V kontextu rovnoběžných souřadnic je přímka ve tvaru $\ell : y = mx + b$ (v $u - v$ prostoru) reprezentována jako $\overline{\ell} = (d, b, 1-m)_{\mathbb{P}^2}$, kde d je vzdálenost mezi rovnoběžnými osami x' a y'. Reprezentace přímky $\overline{\ell}$ je mezi osami x' a y' pouze pokud $-\infty < m < 0$. Pro $m = 1$ je $\overline{\ell}$ ideální bod (bod v nekonečnu). Pro $m = 0$ leží $\overline{\ell}$ na ose y', pro svislé linie ($m = \pm\infty$), $\overline{\ell}$ leží bod na ose x'.
Mimo tento prostor rovnoběžných souřadnic x', y' (dále zvaný rovný prostor $\mathcal{S}$), je také použit pokroucený prostor $x', -y'$ (neboli $\mathcal{T}$), který je identický prostoru $\mathcal{S}$, pouze osa y je převrácená. V prostoru $\mathcal{T}$ leží $\overline{\ell}$ mezi osami $x', -y'$ pouze, pokud $0 < m < \infty$. Kombinací těchto dvou prostorů nám vznikne rovina $\mathcal{TS}$, kterou můžeme vidět na obrázku 3.23.



Obrázek 3.23: Znázornění jednotlivých prostorů. Převzato z [8]. **Nahoře:** Původní $x - y$ prostor. **Dole:** PClines reprezentace odpovídající $\mathcal{TS}$ prostoru.

Každá přímka $\ell = mx + b$ je tedy reprezentována buď jako $\overline{\ell}_{\mathcal{S}}$ v rovné rovině, nebo jako $\overline{\ell}_{\mathcal{T}}$ v pokroucené rovině.

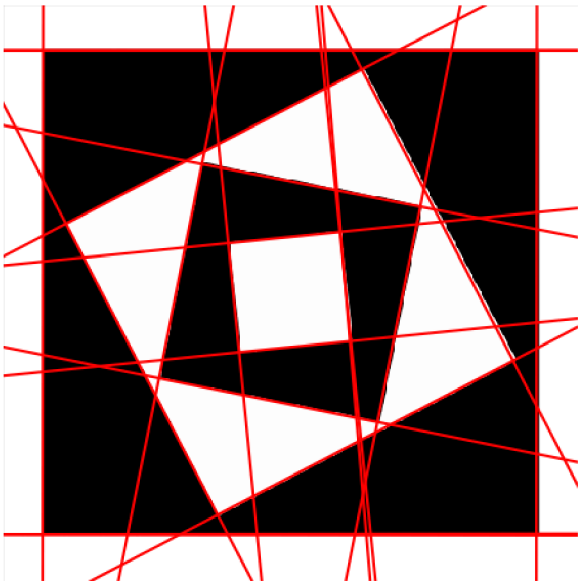
$$\overline{\ell}_{\mathcal{S}} = (d, b, 1 - m)_{\mathbb{P}^2}, -\infty \leq m \leq 0, \tag{3.5}$$

$$\overline{\ell}_{\mathcal{T}} = (-d, -b, 1 + m)_{\mathbb{P}^2}, 0 \leq m \leq \infty \tag{3.6}$$

Každá přímka ℓ tedy má přesně jeden obraz $\overline{\ell}$ v prostoru $\mathcal{TS}$ (vyjma přímek u kterých $m = 0$ a $m = \pm\infty$ kdy $\overline{\ell}$ leží v obou prostorech na y' nebo x'). Toto tedy dovoluje prostorům $\mathcal{T}$ a $\mathcal{S}$, aby byly spojeny. Na obrázku 3.23 můžeme vidět prostor spojený podle osy x'.
Pro detekci přímek je použitá standardní Houghova transformace, $u-v$ prostor je uniformně diskretizován do akumulační matice. Vstupní obrázek je zpracován a pro všechny jeho pixely mimo rozsah je podmnožina akumulátorů inkrementována. V případě metody PCLines [8] jsou rastrovány dvě přímky pro každý vstupní pixel - jedna v prostoru $\mathcal{S}$ a druhá v prostoru $\mathcal{T}$. Vodorovné souřadnice koncových bodů přímek jsou fixní $\{0, d, -d\}$, svislé souřadnice jsou přímo x, y nebo $-y$ původního bodu.
Pro detekci přímek v PClines prostoru, můžeme lehce vypočítat průsečíky s $x - y$ hranou obrazu následovně:

$$v = \pm\frac{1}{2}\left(\pm\frac{W + H}{d}u + W\right). \tag{3.7}$$

Metoda PCLines [8] je díky své parametrizaci ideální pro méně složité obrázky a je schopná rychle rozpoznat rovnoběžné přímky v obraze, proto je tedy vhodná pro sken čárových kódů nebo textu. Není však ideální na rozpoznání přímek v reálných prostranstvích.



Obrázek 3.24: Rozpoznání přímek pomocí PCLines [8] na triviálním obrázku. Původní obrázek převzat z [12].

3.3 Detekce úběžníků

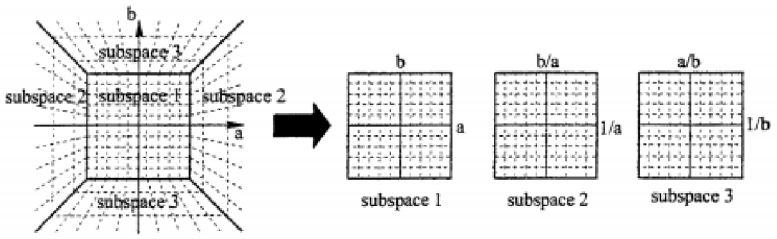
Tato část kapitoly se zabývá nacházením úběžníků ve vstupním obraze. Úběžník je bod nacházející se v nebo mimo vstupní obraz, do kterého se sbíhá větší množství přímek vstupního obrazu. Zpravidla má každá fotografie aspoň tři úběžníky, který každý náleží jednomu směru přímek v obraze. Většina popisovaných metod je však schopna nalézt úběžník bez předchozí znalosti úseček nebo přímek v obraze, pokud má k dispozici vstupní fotografii nebo obrázek dále zpracovaný nějakou z funkcí pro detekci hran (například Cannyho [4]). Avšak předchozí znalost přímek nebo úseček v obraze zajistí mnohem větší přesnost nalezených výsledků.

3.3.1 Kaskádová Houghova transformace

V původní Houghově transformaci [14] jsou přímky reprezentovány pomocí parametrů (a,b) podle vzorce $ax + b + y = 0$. Pomocí této (a, b) parametrizace, je pár hranových souřadnic (x, y) transformován do přímky v prostoru parametrizace (a, b). Podobně bod se souřadnicemi (a, b) v Houghově prostoru odpovídá přímce v prostoru s parametrizací (x, y). Z toho tedy vyplývá, že přímky mohou být reprezentovány jako body v jednom prostoru a naopak. Tato skutečnost umožňuje aplikovat metodu navrženou Tuytelaars et al. [19], tedy opakovat Houghovu transformaci na výstupu předchozí Houghovy transformace. Aby ale mohla být Houghova transformace provedena opakovaně, je nutné upravit neohraničený parametrový prostor Standardní Houghovy transformace na prostor, který je ohraničený.

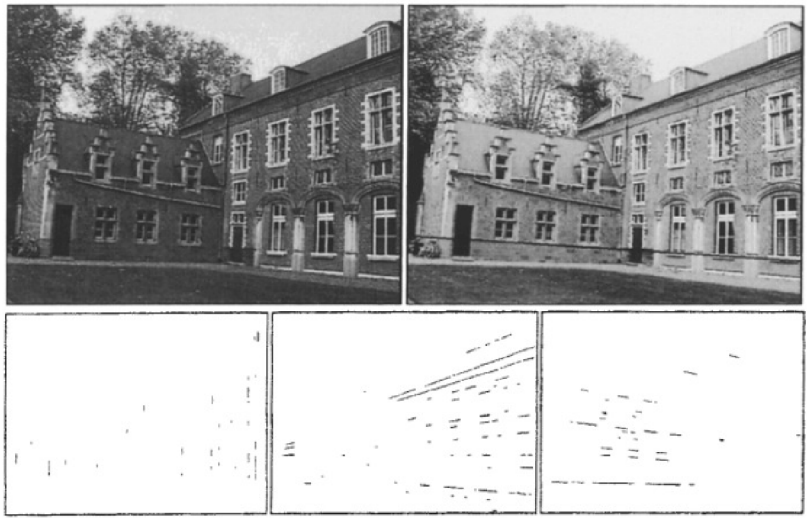
Parametrový prostor pro Kaskádovou Houghovu transformaci musí být ohraničený na rozdíl od prostoru Houghovy transformace [14]. Abychom dosáhli ohraničeného prostoru, rozdělíme originální neohraničený (a, b) prostor do tří ohraničených podprostorů. První prostor má také souřadnice a a b, ale pouze pro $|a| \leq 1$ a $b \leq 1$. Pokud je $a > 1$ a $b \leq a$, tak se bod objeví v druhém podprostoru se souřadnicemi $1/a$ a b/a. Pokud $|b| > 1$ a $|a| < |b|$ tak je použit třetí podprostor se souřadnicemi $1/b$ a a/b. Takto je tedy

původní neohraničený prostor převeden do tří ohraničených podprostorů. Body v těchto podprostorech mají souřadnice omezeny na interval $[-1, 1]$.
Aby toto bylo stále dodrženo, je stejné rozdělení prostoru aplikováno i na (x, y) obrazový prostor.



Obrázek 3.25: Rozdělení neohraničeného prostoru do tří ohraničených. Převzato z [19].

Je-li parametrový prostor ohraničený, k nalezení úběžníků už je zapotřebí pouze provést dvě Kaskádové Houghovy transformace nad vstupními obrazovými daty. Třetí Houghova transformace zajistí nalezení horizontu ve fotografii. Výsledky je možné vidět na obrázku 3.26.



Obrázek 3.26: Výsledky dosažené pomocí Kaskádové Houghovy transformace [19]. Převzato z [19]. **První řada, vlevo:** Původní obraz. **První řada, vpravo:** Nalezený horizont. **Druhá řada:** Nalezené přímky příslušící jednotlivým úběžníkům.

3.3.2 RANSAC

RAndom SAmple Consensus je iterativní metoda pro odhad parametrů matematického modelu z množiny dat, která obsahuje outliery.
Pro tuto práci byla použita RANSAC metoda na výpočet úběžníků podle článku od Chaudhury et al. [5]. Pro tuto konkrétní implementaci metody RANSAC je v prvním kroku důležité nalezení *edgeletů* (Edgelet = abstraktní entita, která je přiřazená ke každému hranovému bodu vstupního obrazu). Edgelet je popsán pomocí tří složek - polohou hran, směrem hrany a sílou hrany - edgelet je tedy definován jako $E = \{\vec{x}, \vec{d}, s\}$. V rámci článku [5] jsou

edgelety vypočítány pomocí Harrisonova detektoru rohů, v této práci však jsou edgelety tvořeny z již nalezených přímek nebo úseček.

Odhad úběžníků probíhá po zvolený počet iterací. Čím více iterací, tím přesnější by měl výsledek být. S větším počtem iterací však stoupá i časová náročnost. Na začátku každé iterace dojde k náhodnému výběru edgeletů z množiny edgeletů, z optimalizačních důvodů je však vybrán první edgelet E_1 z horních dvaceti percentilů a druhý edgelet E_2 z horních padesáti percentilů (předpoklad tedy je, že jsou edgelety seřazeny podle síly jejich hrany od nejvyšší k nejnižší). Jakmile jsou vybrány modelové dva edgelety, jsou vybrány přímky nebo úsečky $\vec{l_1}$ a $\vec{l_2}$ náležící těmto edgeletům. Jelikož jsou oba vektory $\vec{l_1}, \vec{l_2}$ v homogenních souřadnicích, lze z nich získat jejich průsečík a tedy modelový úběžník pro aktuální iteraci vektorovým součinem $\vec{v_m} = \vec{l_1} \times \vec{l_2}$. Může se stát, že nalezený průsečík je v nekonečnu (má tedy složku z nulovou), nebo jsou náhodně vybrané edgelety stejné, v těchto případech je úběžník zahozen a proces je opakován. Pokud však vypočítaný úběžník je validní, je potřeba pro něj vytvořit hodnocení. Pro každý edgelet je vůči modelovému úběžníku vypočítáno hodnocení podle vzorce 3.8, kde θ značí úhel mezi úsečkou $\vec{l_i}$ edgeletu E_i a vektorem spojující jeho směr $\vec{x_i}$ s modelovým úběžníkem. Pokud není aktuálně žádný vybraný nejideálnější úběžník nebo je suma hlasů pro aktuální úběžník větší než pro nejlepší vybraný úběžník, je vybrán aktuální úběžník jako nejideálnější.
Jakmile je vypočítán první úběžník, jsou odstraněny všechny *inlier* edgelety (tedy edgelety, které byly použity pro nalezení prvního úběžníku) z množiny edgeletů a je možné pokračovat k detekci druhého úběžníku úplně stejným způsobem. Tento proces může být opakován dokud stále existují edgelety, které je možné použít.

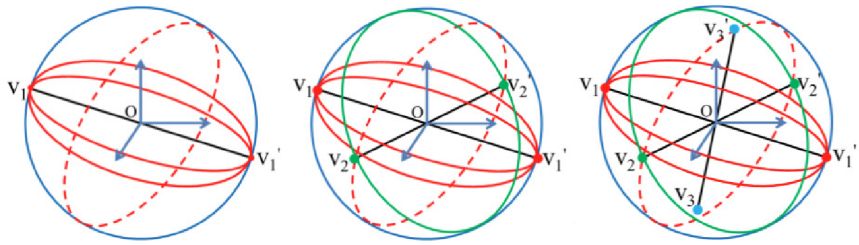
$$vote(E_i, M(E_1, E_2)) = \begin{cases} \frac{1-e^{-\lambda \cos^2 \theta}}{1-e^{-\lambda}} & \text{if} \leq 5°; \\ 0 & \text{jinak;} \end{cases} \tag{3.8}$$

3.3.3 3-line RANSAC

Tato metoda je jednou z mnoha modifikací metody RANSAC [5] pro získání úběžníků z obrazu. 3-line RANSAC podle Bazin et al. [3] je založen na algoritmu běžného RANSAC, proto budou v popisu určité kroky přeskočeny a budou popsány rozdíly této metody oproti metodě předchozí.

Princip metody a rozdíl oproti běžné RANSAC metodě je v počtu čar, ze kterých se odhadují modelové úběžníky. Zatímco metoda RANSAC zmíněná v článku [5] vybírá na začátku každé své iterace dvě čáry, popisovaná metoda vybere naprosto náhodně úsečky tři. Tyto vybrané úsečky jsou v rámci metody 3-Line RANSAC [3] reprezentovány sféricky.

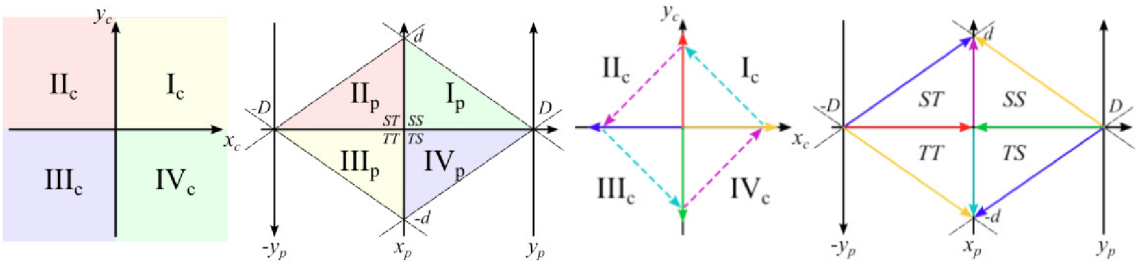
Jsou-li tedy úspěšně vybrané tři přímky stejným způsobem jako v metodě popisované článkem [5], je možné nalézt první z úběžníků , tedy v_1 a jeho opačný bod v_1', které jsou průsečíky první a druhé přímky. Přímky jedna a dva jsou reprezentovány na obrázku 3.27 jako červené kružnice. Tento krok je možné vidět na obrázku 3.27.
Body v_2 a v_2' lze nalézt podobným způsobem, tento krok je demonstrován na obrázku 3.27 uprostřed. Tyto body jsou průsečíky zelené kružnice (značící třetí vybranou čáru) a červené čárkované kružnice, která znázorňuje kružnici pro normálu k v_1.
Třetí úběžník, tedy bod v_3 a v_3' lze snadno vypočítat jako $v_3 = v_1 \times v_2$.



Obrázek 3.27: Názorná ukázka detekce úběžníků pomocí metody 3-Line RANSAC [3]. Převzato z [3]. **Vlevo:** První krok metody, detekce v_1 a v_1'. **Uprostřed:** Druhý krok metody, detekce v_2 a v_2'. **Vpravo:** Třetí krok, detekce v_3 a v_3'

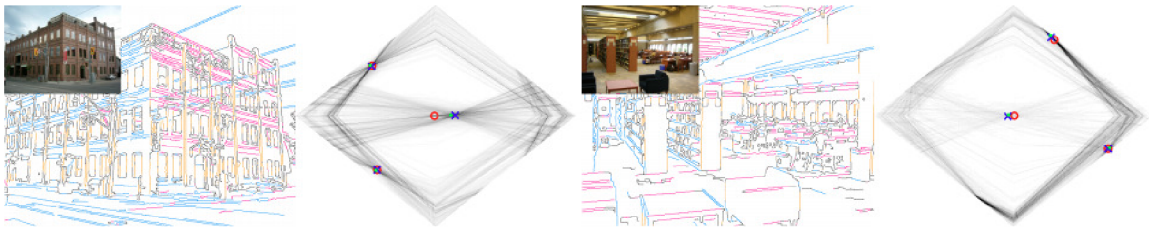
3.3.4 Diamond Space

Tato metoda od Herout et al. [7] staví na jejich předchozích pracích v oblasti počítačového vidění. Čerpá z metody PCLines [8], Kaskádové Houghovy transformace [19] a kombinuje zmíněné metody takovým způsobem, aby nebylo nutné použití více nových prostorů.
Spojením čtyř prostorů do jednoho omezeného prostoru můžeme reprezentovat čtyři Kaskádové Houghovy transformace, spojením těchto čtyř prostorů vznikne prostor, který má podobu diamantu tedy tzv. *diamond space*, znázorněný na obrázku 3.28. V tomto prostoru však už přímka není reprezentována rovnou čarou, ale křivkou, jejíž počet částí záleží na počtu kvadrantů, kterýma v kartézském prostoru prochází.



Obrázek 3.28: Jednotlivá mapování. Převzato z [7]. **1. zleva:** Kvadranty původního neomezeného kartézského prostoru. **2. zleva:** Kvadranty prostoru paralelních souřadnic. **3. zleva:** Po kaskádování dvou mapování jsou cílové kvadranty ohraničené a trojúhelníkové. **4. zleva:** Tyto trojúhelníkové oblasti mohou být propojeny jednotlivými hranami, které reprezentují stejné body v původním prostoru.

Detekce ortogonálních úběžníků je dosažena za pomoci edgeletů, které byly získány předchozím zpracováním vstupního obrazu. Tyto edgelety jsou akumulovány podle správného postupu do Diamond Space. Nejvyšším vrcholem v tomto prostoru je nejdominantnější úběžník v obraze, všechny edgelety, které pro tento úběžník volily jsou odstraněny a tento proces je opakován, dokud je možné nějaký další úběžník nalézt.
Použitím vzorce a parametrů fotoaparátů je bod zobrazen zpět do souřadnic původního obrazu.



Obrázek 3.29: Dva příklady detekce úběžníků pomocí Diamond Space [7]. Převzato z [7]. **Červené kroužky:** Neortogonální úběžníky. **Modré křížky:** Ortogonální trojice úběžníků. **Zelené tečky:** Anotované úběžníky.

3.4 Datové sady pro rozpoznání čar

K rozpoznání přímek, úseček a úběžníků bylo zapotřebí získat fotografie s anotovanými daty, na kterých lze jednoduše porovnávat efektivitu jednotlivých metod. Pro dosažení co nejpřesnějších a nejrozsáhlejších statistických dat byly využity tři datové sady. Z těchto vybraných datových sad jsou dvě velice hojně používané pro porovnávání jednotlivých metod. Třetí byla vybrána za účelem porovnání výkonu na datové sadě, na které nebyla ani jedna z použitých metod testována.

3.4.1 Wireframe dataset

Tato datová sada od Huang et al. [15] je nejobsáhlejší z použité trojice datových sad. Obsahuje 5462 fotek vnitřních nebo venkovních prostorů. 5000 z těchto fotek byly určeny na trénování, zbývající fotky byly určeny pro testování. Anotace jak pro úsečky, tak pro jejich průsečíky byly ručně vyznačeny autory. Tato datová sada je ideální pro potřeby této práce, jejímž cílem je zaměření se na rektifikaci vnitřních a venkovních prostorů. Pro samotnou evaluaci jsou použité jak trénovací, tak testovací fotografie. V této datové sadě jsou však anotovány pouze úsečky. V rámci práce tedy bylo nutno nalézt z těchto anotovaných úseček úběžníky pomocí postupu navrhovaného v podkapitole 4.1.

Obrázek 3.30: Ilustrace anotovaných hodnot pro datovou sadu Wireframe [15]. **První řada:** Anotované úsečky. **Druhá řada:** Nalezené úběžníky z anotovaných úseček.

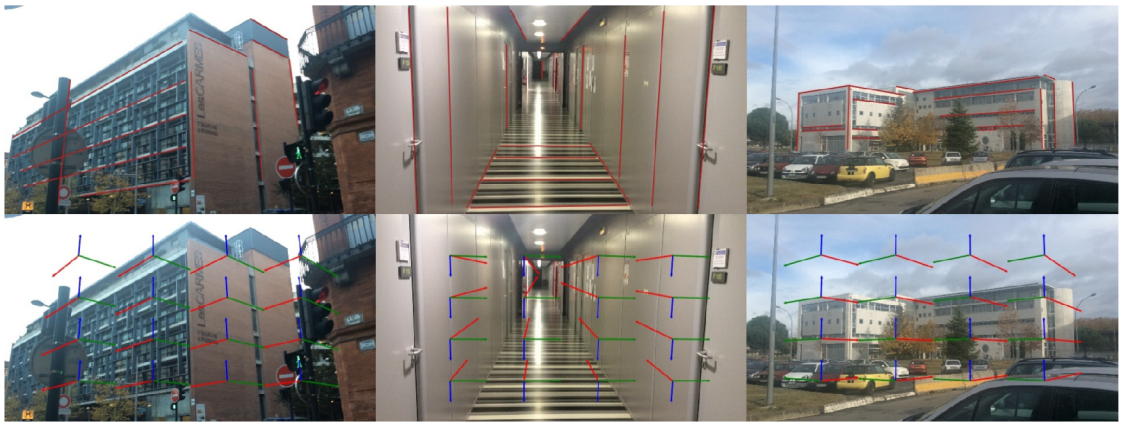
3.4.2 York Urban Database

York Urban Database je druhou z datových sad, které byly v minulosti hojně používány na porovnání metod detekce úseček. Tato datová sada od Denis et al. [6] obsahuje 112 fotografií pořízených fotoaparátem Panasonic Lumix DMC-LC80. Je rozdělena na 2 podmnožiny - 10 fotografií, které byly použity pro kalibraci fotoaparátu a 102 fotografií (45 z vnitřních prostorů, 57 z venkovních prostorů), které byly pořízené pro evaluaci a trénování algoritmů. Anotace dat byla uskutečněna pomocí autory vytvořeného MATLAB programu, který umožňuje vyznačit úsečky v obraze a přiřadit je k jednomu ze tří úběžných směrů. Datová sada mimo anotace úseček obsahuje i anotace, ke kterým úběžníkům přísluší jednotlivé anotované úsečky a tudíž není problém z těchto informací získat jednotlivé úběžníky.

Obrázek 3.31: Ilustrace anotovaných hodnot pro York Urban Database [6]. **První řada:** Anotované úsečky. **Druhá řada:** Anotované úběžníky.

3.4.3 Toulouse Vanishing Point Dataset

Jak již vyplývá z názvu, datová sada od Angladon et al. [2] obsahuje fotografie s anotacemi k úsečkám a úběžníkům. Na obrázku 3.32 je však možné vidět, že oproti předchozím dvěma datovým sadám jsou zaznamenané anotované hodnoty poněkud více strohé a méně detailní. Výhodou použití této datové sady však je, že nebyla zatím použitá pro evaluaci metod popsaných v předchozích kapitolách, čímž přispěje k zvýšení různorodosti vytvářené statistiky. Tato datová sada obsahuje 114 fotografií (40 vnitřních a 74 vnějších prostorů) pořízených pomocí iPad Air 1. Podobně jako předchozí datové sady, je tato datová sada ideální pro náplň mé práce - tedy rektifikace vnějších i vnitřních prostorů.

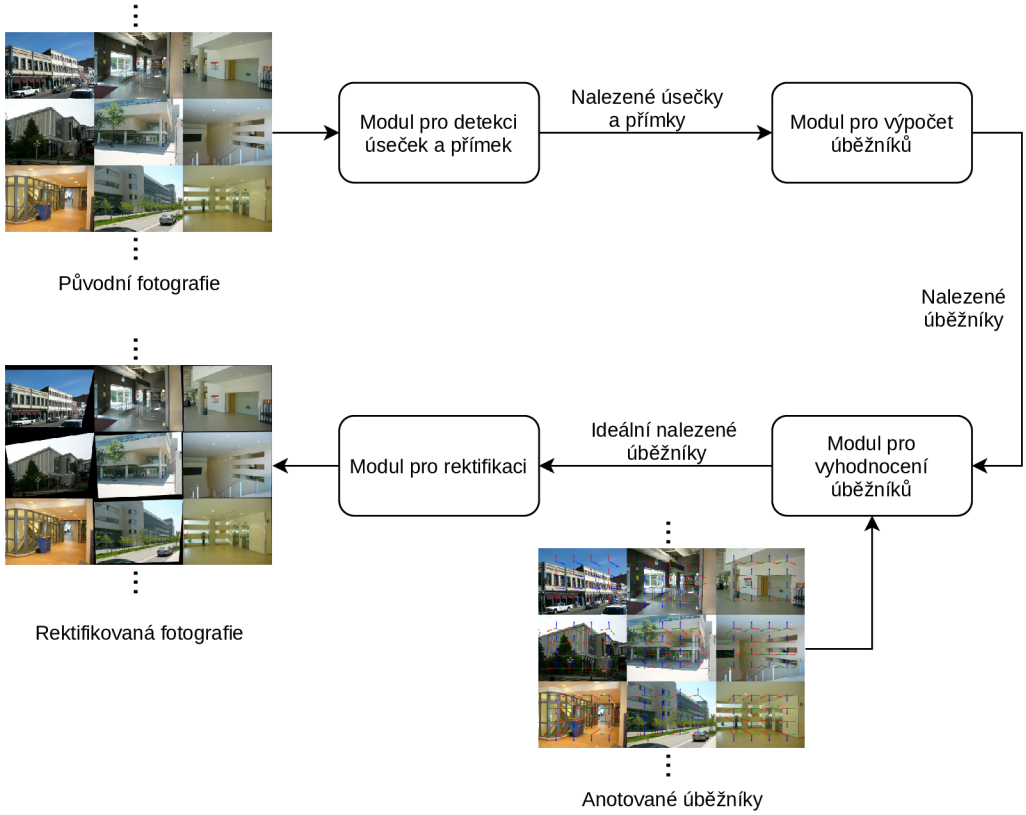


Obrázek 3.32: Ilustrace anotovaných hodnot pro Toulouse Vanishing Point Dataset [2]. **První řada:** Anotované úsečky. **Druhá řada:** Anotované úběžníky.

Kapitola 4

Návrh řešení

V této kapitole je představen návrh pro získání dat a vytvoření finálního rektifikovaného obrazu. Jednotlivé kroky a moduly jsou demonstrovány v obrázku 4.1.

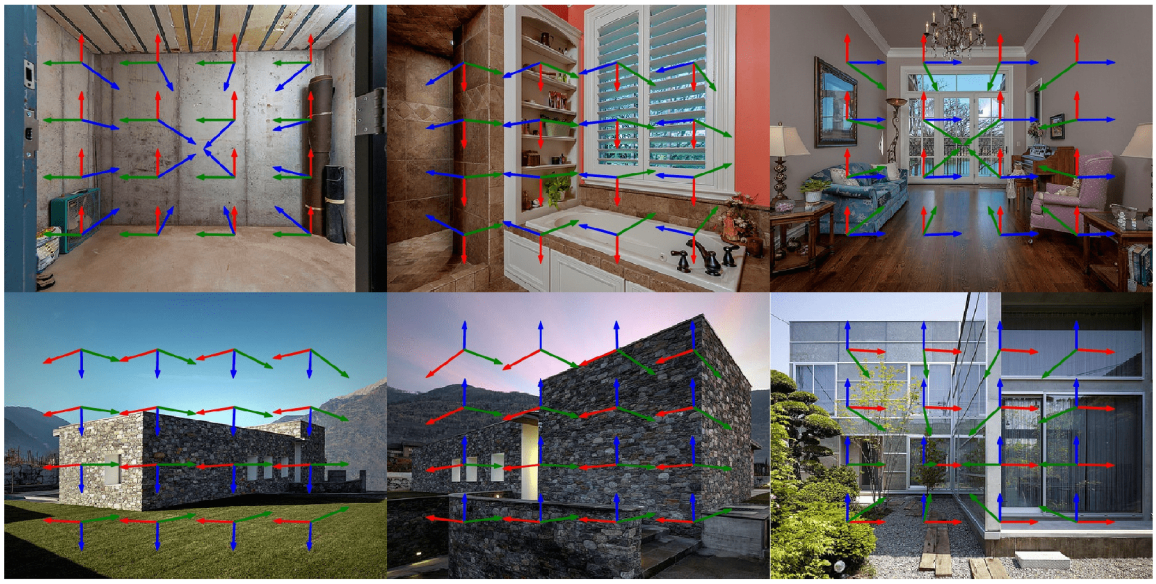


Obrázek 4.1: Diagram ilustrující jednotlivé moduly a kroky navrhovaného řešení.

4.1 Sjednocení anotací

V první řadě je nutno upravit a sjednotit data, která budou hodnocena a rektifikována. Jak už bylo zmíněno v kapitole 3.4 v datových sadách Toulouse Vanishing Point Dataset [2] a York Urban Database [6] jsou úsečky seskupeny do množin podle úběžníků, do kterých se sbíhají. Samotné úběžníky však v anotacích chybí a je tedy nutné, aby byly nalezeny pro

následnou referenci při rozpoznávání úběžníků jednotlivými kombinacemi metod. Pro jejich nalezení je v rámci práce použita metoda RANSAC [5], která je spuštěna s velkým počtem iterací nad každou skupinou úseček náležících jednomu úběžníku, čímž je zajištěna větší přesnost.
Stejný postup byl použit i pro nalezení úběžníků v rámci datové sady Wireframe [15], ve které však nejsou specifikovány informace o úběžnících. Metoda RANSAC [5] je tedy spuštěna i nad touto datovou sadu s počtem 100 000 iterací pro každou fotografii, aby došlo k zajištění co největší přesnosti dat, která se budou používat jako anotovaná. Výsledky tohoto kroku jsou demonstrovány na obrázku 4.2.
Data jsou zaznamenána, zkoprimována, uložena a slouží pro čerpání informací k evaluaci.



Obrázek 4.2: Úběžníky nalezené pomocí RANSAC [5] z anotovaných úseček v datové sadě Wireframe [15].

4.2 Nalezení úseček nebo přímek

Pro získání co nejpřesnějších a nejrůznorodějších výsledků byly vybrány co nejodlišnější ze studovaných metod. V rámci řešení práce jsou tedy využity jak metody, které používají strojového učení pro detekci úseček, tak nějaké z rychlejších a triviálnějších analytických metod, které je nepoužívají.
Z metod stavěných na strojovém učení byly vybrány metody AFM [20] a HAWP [21], jelikož dle jednotlivých studií by metoda HAWP [21] momentálně měla být jednou z nejefektivnějších a nejúčinnějších metod pro detekci úseček v obraze a samotná staví na poznatcích získaných při vývoji metody AFM [20]. Z metod rychlejších byla vybrána metoda LSD [10], která disponuje jednoduchostí, stejně jako metoda Houghovy transformace [9] a Progresivní pravděpodobnostní Houghovy transformace [16]. Dále v rámci práce byla zvažována metoda PCLines [8], avšak po snaze ji aplikovat na fotografie z datových sad, které pro implementaci byly použity bylo uznáno, že metoda není vhodná pro charakter této práce a nebyla tedy použita.
V této fázi tedy dojde ke spuštění metod, nad každou fotografií, ze zvolené datové sady. V rámci ušetření času, jelikož metody HAWP [21] a AFM [20] mají větší časovou náročnost,

jsou pro pozdější zhodnocení pro každou metodu a datovou sadu uloženy komprimované Python slovníky obsahující data pro jednotlivé vstupní fotografie.

4.3 Nalezení úběžníků

Získání úběžníků z již nalezených přímek a úseček je skalní částí pro provedení následující rektifikace a pro správné zpracování vstupního obrazu. Pro tuto úlohu bylo žádoucí, aby byla vybrána jedna z metod pracující s Houghovou transformací. Proto byla vybrána metoda Diamond Space [7], která je dostatečně rychlá a dostatečně přesná pro požadované účely. Jako další byla vybrána metoda RANSAC [5], která je výpočetně a časově náročnější, ale měla by být přesnější. Také byla vybrána její zjednodušená verze 3-Line RANSAC [3].
Podobně jako v předchozím kroku tedy dojde ke spuštění všech metod nad daty získanými při nalezení úseček a přímek. Tato data jsou také zaznamenána do slovníků, zkomprimována a uložena.

4.4 Zpracování dat a výběr použitelných prvků

Po nalezení všech potřebných prvků je nutné zpracovat a vybrat optimální úběžníky. Je nutné spočítat úhel θ mezi jednotlivými anotovanými úběžníky a mezi nalezenými úběžníky pro každou kombinaci metod. Úhel θ dostaneme po převedení obou bodů, anotovaného a nalezeného, do homogenních souřadnic. Máme-li vektor $\vec{v_{gt}}$ značící anotovaný úběžník v homogenních souřadnicích a bod $\vec{v_d}$ značící bod nalezený jednou z kombinací metod v homogenních souřadnicích, je možné použít vzorec $\theta = \arccos(\frac{\vec{v_d} \cdot \vec{v_{gt}}}{|\vec{v_d}| * |\vec{v_{gt}}|})$. Úhel θ je úhlem, který svírají dva vektory a tedy hledaný úhel mezi těmito body. Každý nalezený úběžník je tedy podle úhlu θ přiřazen k nejbližšímu anotovanému úběžníku. Pokud by však nějaká z metod našla více úběžníků, než je anotovaných, docházelo by k chybnému přiřazování úběžníků, z tohoto důvodu je tedy anotovanému úběžníku přiřazen vždy pouze jeden nalezený úběžník a to pouze pokud spadá jeho úhel θ do prahu t (v této práci $t = 30°$), jinak je vyřazen.
Dále je nutné vybrat ze všech metod optimální úběžníky odpovídající anotovaným hodnotám. V tomto kroku dojde k vybrání úběžníků s nejmenšími úhly. Zde se však práce nezaměřuje pouze na jednu metodu. Optimální úběžníky se hledají napříč všemi metodami, tímto způsobem jsou nalezena opravdu nejvhodnější data pro finální rektifikaci obrazu. Aby nedocházelo k výběru nesmyslných dat, je i zde nastaven práh t_{max} (v mé práci $t_{max} = 10°$), pod který musí úhel θ nalezeného úběžníku spadat, pokud tuto podmínku nesplňuje, je úběžník označený jako nenalezený.
Může se samozřejmě stát, že pro fotografii nebyl nalezen dostatečný počet úběžníků. V tomto případě nelze fotografii rektifikovat, jelikož neexistují všechna data, která jsou k výpočtu finální homografie potřebná. Nastane-li tento případ, je fotografie vyřazena z procesu a rektifikace se pro ni neprovede.

4.5 Finální rektifikace

Jestliže jsou určeny všechny možné úběžníky pro každou fotografii, pro kterou bylo úběžníky možné najít, je možné přesunout se na samotnou rektifikaci. Pro každou fotografii je tedy potřeba nalézt homografii H způsobem, který byl popsán v kapitole 2. Řešení se tedy zabývá pouze rektifikací vertikální, jelikož je modul navrhnut jako pomůcka pro realistické zobrazení a při vertikální rektifikaci dosáhneme menšího zkreslení obrazu.

Po získání finální homografie je aplikována matice na pole reprezentující obrazová data a je obnovena rovnoběžnost vertikálních čar v obraze.

4.6 Vyhodnocení výsledků

Mimo rektifikace obrazu je potřeba také vyhodnotit výsledky pro jednotlivé kombinace metod. V kroku 4.4 došlo k ohodnocení jednotlivých nalezených úběžníků na základě úhlové vzdálenosti od anotovaných úběžníků v poskytnutých datech. Díky těmto úhlovým vzdálenostem je možné kombinaci metod úspěšně vyhodnotit. Všechna data jsou ukládána spolu s informacemi, zda byly jednotlivé úběžníky použity, jaká byla velikost jejich chyby, jaké fotografii náleží a kombinací jakých metod byl úběžník získán. Jsou-li tedy tyto informace pro všechny použité metody a všechny vstupní fotografie získány, je možné úspěšně každou metodu ohodnotit pro fotografie z datové sady. Data jsou seskupena podle kombinací metod, kterými byly získány a jsou nad nimi provedeny operace tak, aby bylo možné pozorovat procentuální pravděpodobnost chyby s jakou kombinace metod nachází úběžníky. Následně je pro každou datovou sadu vytvořena skupina grafů, které znázorňují zmíněná data.

4.7 Použité technologie

Pro implementaci byl použit programovací jazyk Python[1]. Konkrétně byla použita verze Python 3.6 z důvodu konkrétních požadavků metod AFM [20] a HAWP [21].
Dále je nutno zmínit i různé knihovny, které byly použity za účelem zjednodušení práce, popřípadě bylo nutné tyto moduly použít. V celé implementaci programu bylo využito knihovny NumPy[2], což je knihovna pro snadnější a rychlejší práci s poli a daty v rámci jazyka Python. Tato knihovna také obsahuje široké spektrum matematických funkcí, které je možno provádět nad celými poli a tedy výrazně ulehčuje operace nad daty a výpočty většího počtu dat.
Knihovna Pandas[3], která usnadnila vyhodnocení finálních dat a jejich ukládání. Umožňuje také lehčí operace nad databázovými daty a efektivnější práci s databázovými daty.
Pro zpracování vstupního obrazu byly využity knihovny OpenCV[4] a scikit-image[5], které umožňují převod obrázků různých typů do NumPy polí, pomocí kterých lze s obrázkem snadno manipulovat. Dále tyto knihovny byly použity k pokroucení finálního rektifikovaného obrázku pomocí matice homografie a pro finální vizualizaci výstupních obrázků.
Pro zobrazení úběžníků, úseček, přímek a vytvoření finálních histogramů byla použita knihovna Matplotlib[6], která umožňuje snadnou vizualizaci dat a obrázků.
Z hardwarových zdrojů byly pro evaluaci metod použity mé osobní stroje, těmi jsou dva notebooky značky Dell (jeden z nich vybavený grafickou kartou NVIDIA GeForce 1050 Ti). Notebook s výkonnější grafickou kartou byl použit pro získání úseček pomocí metody AFM [20], která vyžaduje grafickou kartu značky NVIDIA. Pro metody AFM [20] a HAWP [21] bylo také nutno použít CUDA Toolkit 10.0[7].

[1] https://www.python.org/
[2] https://numpy.org/
[3] https://pandas.pydata.org/
[4] https://opencv.org/
[5] https://scikit-image.org/
[6] https://matplotlib.org/
[7] https://developer.nvidia.com/cuda-toolkit

Pro výzkum a počáteční trénování metod bylo využito služeb Metacentra[8], které poskytuje výpočetní techniku české akademické obci pro výzkumné účely.
Je nutné dodat, že v rámci řešení této práce nebyly metody pro rozpoznání úseček, přímek nebo úběžníků implementovány. Do všech těchto metod bylo zasahováno a byly poupraveny, avšak pro dosažení co nejlepších výsledků, byla většina z metod převzata. Pro detekci úseček pomocí metod AFM [20] a HAWP [21] byly použity již implementované metody, které vyvinuli a implementovali samotní autoři zmiňovaných vědeckých článků. Implementace AFM[9] a HAWP[10] jsou obě volně dostupné z github repozitářů a také byly použity v souladu s licenčními podmínkami. Metoda LSD [10] je volně dostupná jako knihovna pro programovací jazyk Python. Krátký kód pro implementaci PYLSD[11] je dostupný přímo na stránce s informacemi o balíčku a tento kód byl použit. Implementace pro Progresivní pravděpodobnostní Houghovu transformaci [16] a Houghovu transformaci [9] je použitá ze samotných stránek dokumentace pro Python knihovnu OpenCV[12]. Pro metody RANSAC [5] a 3-Line RANSAC [3] byla použita implementace dostupná z repozitáře Image-rectification[13]. Pro metodu Diamond Space [7] byla využita stejnojmenná knihovna diamond_space[14] a program pro nalezení úběžníků v obraze pomocí Diamond Space[15].

[8] https://metavo.metacentrum.cz/
[9] https://github.com/cherubicXN/afm_cvpr2019
[10] https://github.com/cherubicXN/hawp
[11] https://pypi.org/project/pylsd-nova/
[12] https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html
[13] https://github.com/chsasank/Image-Rectification
[14] https://pypi.org/project/diamond-space/
[15] https://colab.research.google.com/drive/1ISOp7mSgSoIXbPLUlED3FL9vdSZLPeWf?usp=sharing

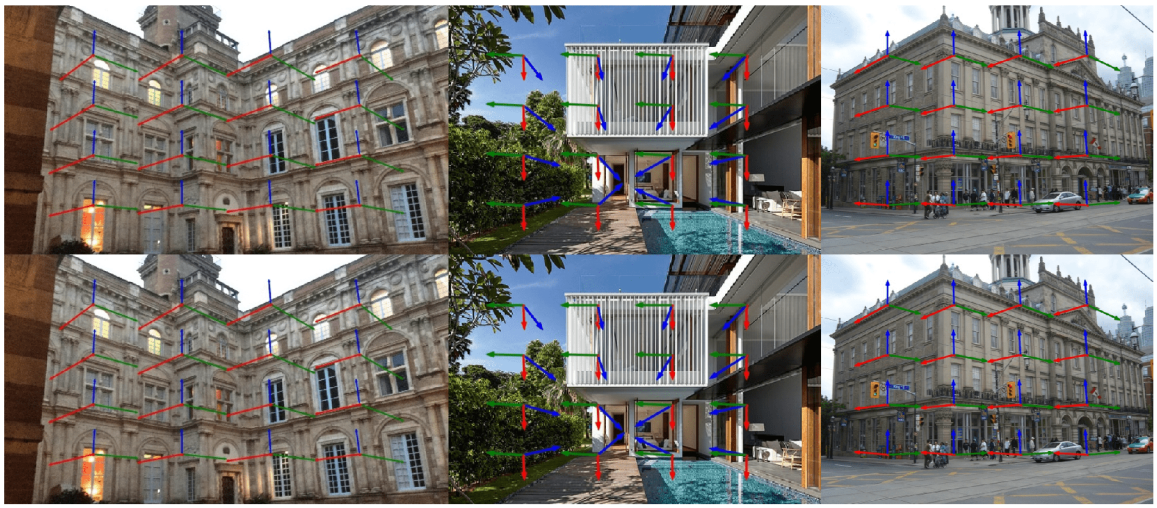
Kapitola 5

Výsledky

Tato kapitola se zabývá zkoumáním výsledného vyhodnocení a shrnutím dosažených výsledků pomocí jednotlivých metod. Statistická data pro porovnání byla dosažena způsobem uvedeným v sekci 4.6. Porovnání je zaměřeno na zhodnocení jednotlivých kombinací metod a jejich efektivitu nad jednotlivými datovými sadami z kapitoly 3.4. V druhé sekci této kapitoly je zhodnocena podoba rektifikovaných fotografií podle jednotlivých datových sad a jsou zmíněny známé chyby, ke kterým dochází při procesu rektifikace.

5.1 Rozpoznání úběžníků v obraze

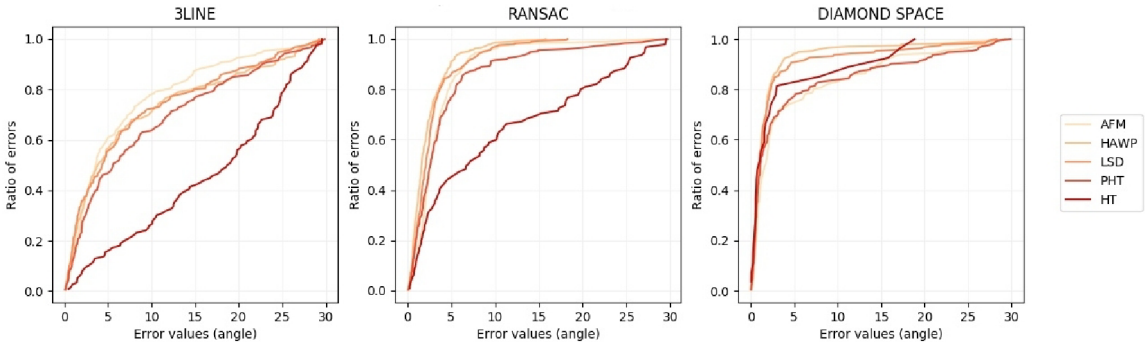
Výsledky při nalézání úběžníků ve vstupním obraze jsou uspokojivé, pro zpracovávané fotografie dosahují metody velice dobrých výsledků a najdou korektní úběžníky. Příklad nalezených úběžníků je demonstrován na obrázku 5.1.
Pro porovnání metod byl použit kumulativní histogram, který se sčítá do 1. Datové sady Toulouse Vanishing Point Dataset [2] a York Urban Database [6] mají každá trojici kumulativních histogramů pro úběžníkové metody RANSAC [5], 3-Line RANSAC [3] a Diamond Space [7]. Datová sada Wireframe [15] má dvojici histogramů pro metody RANSAC [5] a Diamond Space [7]. Dále je přiložena tabulka dosažených výsledků pro každou hodnocenou datovou sadu, ve které je shrnuto, kolik nalezených úběžníků se objevilo pod hranicí 5 stupňů, kolik bylo celkem nalezeno úběžníků a kolik procent úběžníků nalezených kombinací daných metod bylo použito pro výslednou rektifikaci.



Obrázek 5.1: Výsledky rozpoznávání úběžníků. **První řada:** Nalezené úběžníky. **Druhá řada:** Anotované hodnoty.

5.1.1 York Urban Dataset

Z histogramů 5.2 a tabulky 5.1 je zřejmé, že pro tuto datovou sadu je optimální metodou pro rozpoznání úseček metoda HAWP [21], která v kombinaci s metodou Diamond Space [7] najde ve více jak 94 procentech případů úběžník s chybou do 5 stupňů od anotovaných hodnot. V rámci metody Diamond Space [7] je však nutné podotknout, že u všech kombinací s touto metodou došlo k chybě a pro některé fotografie nebyly nalezeny žádné úběžníky (znázorněno v tabulce 5.1 ve sloupci 6) a v případě kombinace s metodou Houghovy transformace [9] chybovost dosahuje až 54.9 procent. Můžeme tedy říct, že metoda HAWP [21] je jednoznačně nejlepší volbou z metod pro nalezení úseček pro tuto datovou sadu, a v kombinaci s metodami RANSAC [5] nebo Diamond Space [7] je i nejúspěšnější pro nalezení použitelných úběžníků. Je také zřejmé, že nejméně ideální metodou pro nalezení hran je metoda Houghovy transformace [9], v kombinaci s metodou Diamond Space [7], jelikož dosahuje nevyhovujících výsledků a velké chybovosti.



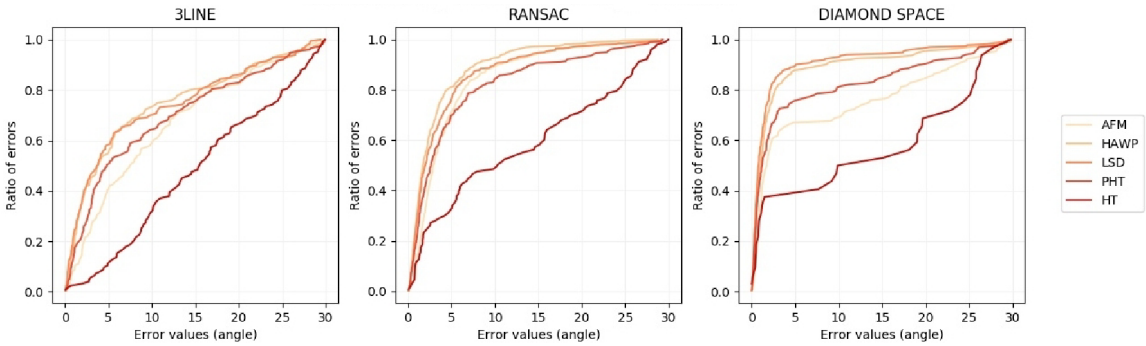
Obrázek 5.2: Kumulativní histogram shrnující přesnost jednotlivých metod. **Nalevo:** výsledky dosažené metodou 3-Line RANSAC [3]. **Uprostřed:** výsledky dosažené metodou RANSAC [5]. **Napravo:** výsledky dosažené metodou Diamond Space [7].

Metoda úběžníků	Metoda čar	< 5.0	celkem	použito (%)	nenalezeno
RANSAC	AFM	82.49%	297	6.0%	0%
	HAWP	91.19%	295	13%	0%
	LSD	85.95%	299	11.67%	0%
	PHT	76.68%	253	6.33%	0%
	HT	44.21%	95	0.33%	0%
3LINE	AFM	60.47%	172	1.67%	0%
	HAWP	56.54%	191	2.67%	0.98%
	LSD	55.26%	190	3.67%	0%
	PHT	46.75%	154	3.67%	0%
	HT	15.52%	116	0.0%	0%
Diamond Space	AFM	74.75%	198	7.33%	2.94%
	HAWP	94.07%	236	18.33%	0.98%
	LSD	90.62%	192	12.33%	0.98%
	PHT	77.01%	174	11.67%	0.98%
	HT	81.48%	27	1.33%	54.9%

Tabulka 5.1: Statistická data dosažená navrhovaným způsobem. **1. sloupec:** Metoda použitá pro rozpoznání úběžníků. **2. sloupec:** Metoda použitá pro rozpoznání úseček nebo přímek. **3. sloupec:** Procento značící, kolik z nalezených úběžníků má rozdíl úhlů s anotovanými hodnotami menší než 5 stupňů. **4. sloupec:** Počet nalezených úběžníků kombinací metod. **5. sloupec:** Procento značící, kolik úběžníků by bylo použito k finální rektifikaci. **6. sloupec:** Procento značící, u kolika fotografií kombinace metod nenalezla žádné úběžníky.

5.1.2 Toulouse Vanishing Point Dataset

Pro tuhle datovou sadu jsou výsledky obdobné v porovnání s datovou sadou York Urban Database [6]. Lze je pozorovat v obrázku 5.3 a tabulce 5.2. Pro celou trojici kombinací metod pro nalezení úběžníků dosahuje průměrně nejlepších výsledků metoda HAWP [21], nejpřesnější kombinací metod je však metoda LSD [10] s metodou Diamond Space [7]. Pro tuto datovou sadu je tedy optimální metodou na detekci úběžníků kombinace metod LSD [10] a Diamond Space [7]. Jednoznačně nejhorších výsledků dosahuje znovu metoda Houghovy transformace [9] v kombinaci s jakoukoliv ze tří metod pro detekci úběžníků.



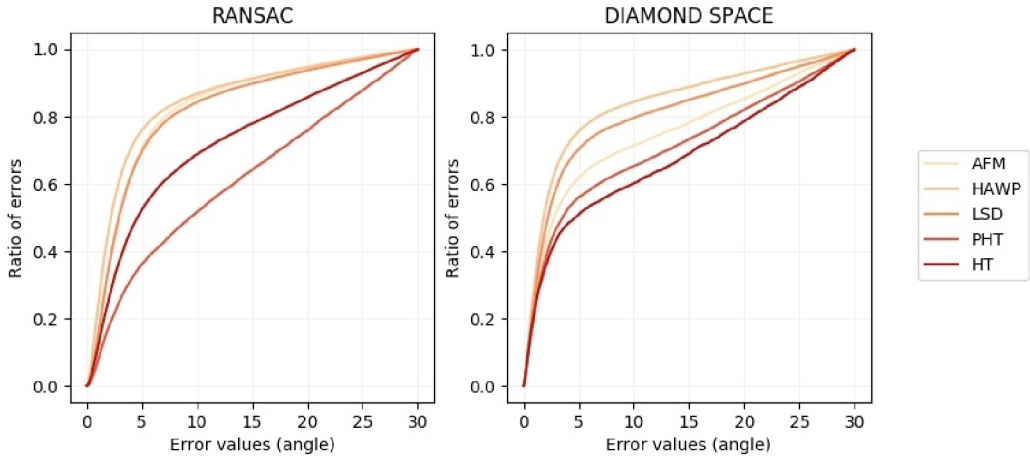
Obrázek 5.3: Kumulativní histogram shrnující přesnost jednotlivých metod. **Nalevo:** výsledky dosažené metodou 3-Line RANSAC [3]. **Uprostřed:** výsledky dosažené metodou RANSAC [5]. **Napravo:** výsledky dosažené metodou Diamond Space [7].

Metoda úběžníků	Metoda čar	< 5.0	celkem	použito (%)	nenalezeno
RANSAC	AFM	71.3%	324	4.41%	0%
	HAWP	80.97%	331	16.76%	0.88%
	LSD	76.11%	339	12.35%	0%
	PHT	69.86%	282	8.24%	0%
	HT	32.04%	103	0.29	0%%
3LINE	AFM	40.35%	171	0.59%	0%
	HAWP	55.15%	194	4.71%	1.75%
	LSD	58.08%	198	5.0%	0%
	PHT	50.33%	153	1.47%	0%
	HT	11.29%	124	0.59%	0.88%
Diamond Space	AFM	67.06%	170	5.88%	3.5%
	HAWP	87.94%	257	14.71%	0.88%
	LSD	89.36%	235	18.24%	0%
	PHT	75.6%	168	6.18%	3.5%
	HT	37.5%	32	0.59%	38.6%

Tabulka 5.2: Statistická data dosažená navrhovaným způsobem. **1. sloupec:** Metoda použitá pro rozpoznání úběžníků. **2. sloupec:** Metoda použitá pro rozpoznání úseček nebo přímek. **3. sloupec:** Procento značící, kolik z nalezených úběžníků má rozdíl úhlů s anotovanými hodnotami menší než 5°. **4. sloupec:** Počet nalezených úběžníků kombinací metod. **5. sloupec:** Procento značící, kolik úběžníků by bylo použito k finální rektifikaci. **6. sloupec:** Procento značící, u kolika fotografií kombinace metod nenalezla žádné úběžníky.

5.1.3 Wireframe dataset

Pro poslední datovou sadu, která je nejobsáhlejší, dopadly výsledky podobně jako u předchozích sad. Nejlepší metodou pro nalezení úseček je opět metoda HAWP [21], která na této datové sadě jednoznačně dosahuje nejlepších výsledků. V rámci přesnosti vychází statistiky pro jednotlivé metody na detekci úběžníků velice podobně, avšak metoda RANSAC [5] nalezne více úběžníků celkově a je tedy větší šance, že nalezne ten správný úběžník.



Obrázek 5.4: Kumulativní histogram shrnující přesnost jednotlivých metod. **Nalevo:** výsledky dosažené metodou RANSAC [5]. **Napravo:** výsledky dosažené metodou Diamond Space [7].

Metoda úběžníků	Metoda čar	< 5.0	celkem	použito (%)	nenalezeno
RANSAC	AFM	70.47%	19053	16.66%	0%
	HAWP	75.86%	18616	17.95%	0.06%
	LSD	69.64%	18607	13.07%	0.18%
	PHT	52.32%	14487	6.12%	0.71%
	HT	36.08%	8162	2.59%	0.073%
Diamond Space	AFM	61.56%	12709	9.48%	1.04%
	HAWP	75.6%	15995	17.81%	0%
	LSD	70.23%	13200	9.42%	0.55%
	PHT	55.92%	9988	4.84%	0.64%
	HT	50.84%	4359	2.06%	35.56%

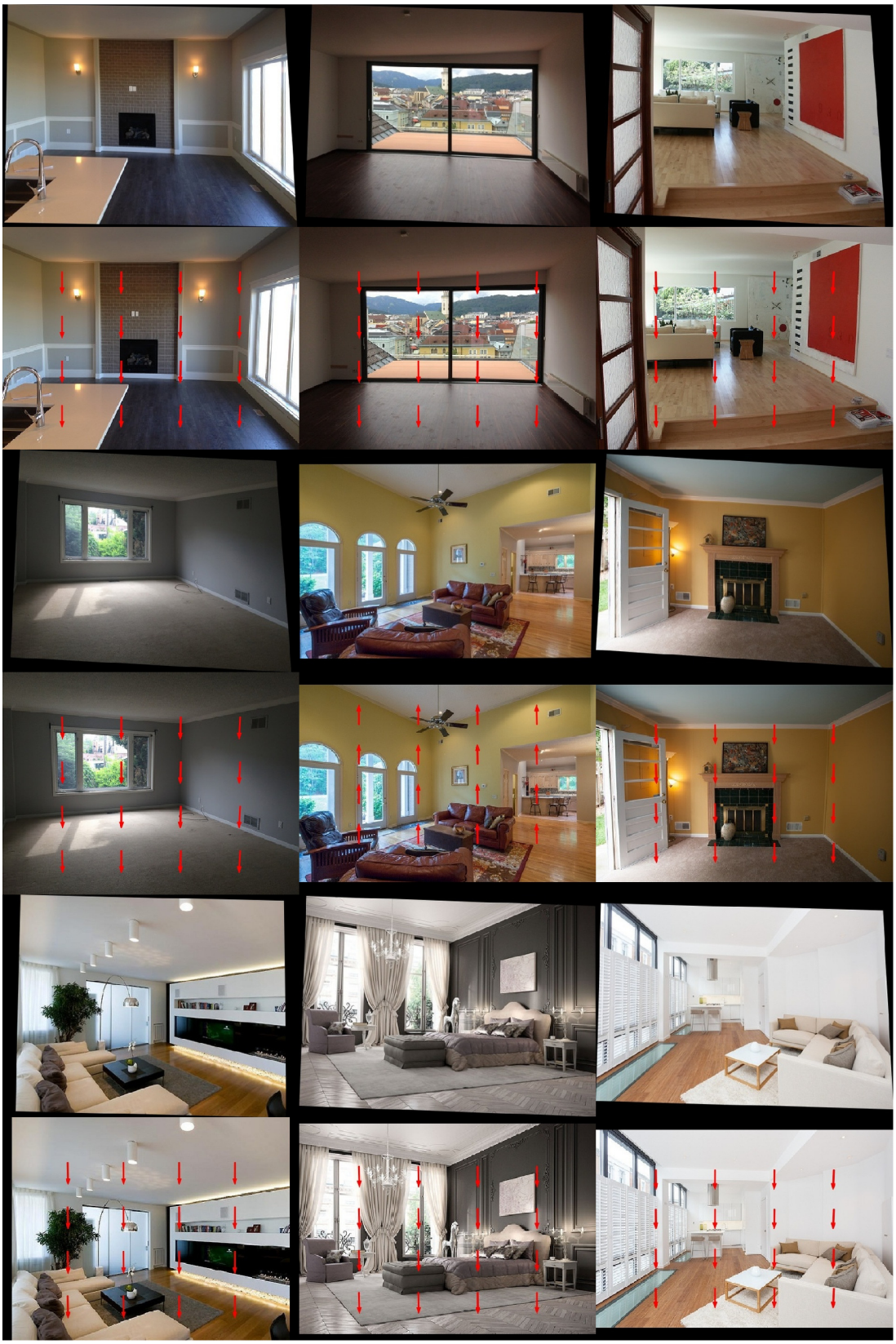
Tabulka 5.3: Statistická data dosažená navrhovaným způsobem. **1. sloupec:** Metoda použitá pro rozpoznání úběžníků. **2. sloupec:** Metoda použitá pro rozpoznání úseček nebo přímek. **3. sloupec:** Procento značící, kolik z nalezených úběžníků má rozdíl úhlů s anotovanými hodnotami menší než $5°$. **4. sloupec:** Počet nalezených úběžníků kombinací metod. **5. sloupec:** Procento značící, kolik úběžníků by bylo použito k finální rektifikaci. **6. sloupec:** Procento značící, u kolika fotografií kombinace metod nenalezla žádné úběžníky.

5.2 Rektifikace vstupních fotografií

Následně budou prezentovány hlavní výsledky této práce. Analýza výsledků rektifikace bohužel není snadná. Proto bylo nutné všechny výsledné rektifikované fotografie zkontrolovat manuálně. Při prohlédnutí všech fotografií se zdá, že navrhovaná metoda dosahuje s nalezenými ideálními úběžníky velice dobrých výsledků a z čar ve vertikálním směru úspěšně udělá navzájem rovnoběžné čáry. Na obrázcích 5.5, 5.6 a 5.7 se nachází příklady výsledných fotografií.

Obrázek 5.5: Porovnání výsledků pro York Urban Database [6].**První řada:** výsledek rektifikace. **Druhá řada:** původní podoba fotografií s vyznačeným vertikálním úběžníkem.

Obrázek 5.6: Porovnání výsledků pro Toulouse Vanishing Point Dataset [2].**První řada:** výsledek rektifikace. **Druhá řada:** původní podoba fotografií s vyznačeným vertikálním úběžníkem.



Obrázek 5.7: Porovnání výsledků pro datovou sadu Wireframe [15].**První řada:** výsledek rektifikace. **Druhá řada:** původní podoba fotografií s vyznačeným vertikálním úběžníkem.

Je nutné také zmínit, že při tak velkém množství fotografií dochází i k nalezení úběžníků, podle kterých nelze fotografii korektně rektifikovat. Zpravidla to jsou úběžníky nacházející se uvnitř fotografie a úběžníky, které leží mimo fotografie, avšak jsou velice blízko hranicím fotografie. Pokud máme k dispozici pouze tyto úběžníky, rektifikace se nemusí zdařit a vstupní fotografie je transformována nesmyslně nebo je vyřazena z procesu rektifikace. Některé známé nedostatky také vychází z nedostatečné přesnosti anotovaných fotografií. Jedná se hlavně o nepřesné vyznačení hran v anotovaných fotografiích, nebo o nedostatečný počet vyznačených hran, což může vést k nalezení nepřesného úběžníku.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit program, který je schopný rektifikace velkého množství fotografií za pomocí kombinace několika metod takovým způsobem, aby byl výsledný rektifikovaný obrázek co nejpřesnější.
Tato práce se nejdřív zaměřila na zkoumání různých metod pro rozpoznání přímek nebo úseček v obraze. U těchto metod byl nastíněn způsob, jakým dokážou nelézt hledané přímky nebo úsečky v obraze. Podle různých kritérií bylo několik z popsaných metod vybráno pro praktickou část práce.
Následně byly vybrány a popsány metody pro nalezení úběžníků v obraze a byl uveden postup, pomocí kterého lze úspěšně provést rektifikaci vstupního obrazu, jsou-li úběžníky získané nějakou z popsaných metod.
Jelikož k provedení navrhovaného postupu je zapotřebí větší datové sady obrázků nebo fotek, byly vybrány dvě vhodné datové sady, na kterých již byla většina metod pro rozpoznání přímek nebo úseček vyzkoušena. Pro získání obsáhlejších statistik byla přidána ještě třetí datová sada, na které žádná z metod v rámci předchozích studií testována nebyla. Všechny datové sady zobrazují reálné prostory a je tedy možné na nich otestovat, zda se rektifikace úspěšně provede na fotografiích z reálného světa.
Součástí této práce bylo implementováno navrhované řešení a byl úspěšně vytvořen program, který je schopný vertikální rektifikace, jsou-li dostupná anotovaná data pro všechny vstupní fotografie. Současně v rámci programové implementace bylo vytvořeno statistické ohodnocení zkoumaných metod a výsledky byly vizualizovány pomocí grafů. Postupným zkoumáním bylo zjištěno, že ve většině případů jsou pro nalezení úseček ideální metody používající strojové učení (konkrétně metoda HAWP [21]). Tato metoda následně v kombinaci s jednou z metod RANSAC [5] nebo Diamond Space [7] dosahuje ve většině případů nejpřesnějších výsledků.
Výsledné rektifikované fotografie byly zkontrolovány ručně a kromě určitého procenta fotek, u kterých buď nebyl nalezen vhodný vertikální úběžník ($< 1\%$ z celkového počtu obrázků) nebo výsledný obrázek nevypadá uspokojivě, byly všechny zbývající fotografie úspěšně rektifikovány.
Práce by se dále mohla rozšířit o další možné metody pro rozpoznání úseček nebo přímek v obraze, čímž by se mohlo dosáhnout ještě obsáhlejšího zhodnocení metod a pravděpodobně by bylo dosaženo ještě přesnějších výsledků v rámci statistiky jednotlivých metod. Stejně tak by bylo možné analyzátor rozšířit o další metody pro rozpoznání úběžníků a různé datové sady, popřípadě zpřesnit anotovaná data v již použitých datových sadách, což by též přispělo k dosažení detailnějších výsledků. Proces rektifikace by se mohl rozšířit o možnost výběru uživatele, jaký typ rektifikace si přeje - tedy zda-li chce vertikální, hori-

zontální nebo kombinaci obou. Také by bylo možné práci kompletně rozšířit a ze získaných poznatků vytvořit automatický rektifikátor, který by nepotřeboval anotované hodnoty pro rektifikaci vstupních fotografií.

Literatura

[1] Akinlar, C. a Topal, C. Edlines: Real-time line segment detection by Edge Drawing (ed). In: *2011 18th IEEE International Conference on Image Processing*. IEEE, 2011, s. 2837–2840. DOI: 10.1109/ICIP.2011.6116138. ISBN 978-1-4577-1303-3.

[2] Angladon, V., Gasparini, S. a Charvillat, V. The Toulouse Vanishing Points Dataset. In: *Proceedings of the 6th ACM Multimedia Systems Conference*. New York, NY, USA: Association for Computing Machinery, 2015, s. 231–236. MMSys '15. DOI: 10.1145/2713168.2713196. ISBN 9781450333511. Dostupné z: https://doi.org/10.1145/2713168.2713196.

[3] Bazin, J.-C. a Pollefeys, M. 3-line RANSAC for orthogonal vanishing point detection. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, s. 4282–4287. DOI: 10.1109/IROS.2012.6385802. ISBN 978-1-4673-1736-8.

[4] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986, PAMI-8, č. 6, s. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[5] Chaudhury, K., DiVerdi, S. a Ioffe, S. Auto-rectification of user photos. *2014 IEEE International Conference on Image Processing, ICIP 2014*. Leden 2015, s. 3479–3483. DOI: 10.1109/ICIP.2014.7025706.

[6] Denis, P., Elder, J. H. a Estrada, F. J. Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery. In: Forsyth, D., Torr, P. a Zisserman, A., ed. *Computer Vision – ECCV 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 197–210. ISBN 978-3-540-88688-4.

[7] Dubská, M. a Herout, A. Real Projective Plane Mapping for Detection of Orthogonal Vanishing Points. In: *Proceedings of BMVC 2013*. The British Machine Vision Association and Society for Pattern Recognition, 2013, s. 1–10. DOI: 10.5244/C.27.90. ISBN 1-901725-49-9. Dostupné z: https://www.fit.vut.cz/research/publication/10400.

[8] Dubská, M., Herout, A. a Havel, J. PClines - Line Detection Using Parallel Coordinates. In: *Proceedings of CVPR 2011*. IEEE Computer Society, 2011, s. 1489–1494. ISBN 978-1-4577-0393-5. Dostupné z: https://www.fit.vut.cz/research/publication/9488.

[9] Duda, R. a Hart, P. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*. 1972, sv. 15, s. 11–15.

[10] Gioi, R., Jakubowicz, J., Morel, J.-M. a Randall, G. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE transactions on pattern analysis and machine intelligence*. Duben 2010, sv. 32, s. 722–32. DOI: 10.1109/TPAMI.2008.300.

[11] *Diamond_space/chess.png at master · MarketaJu/diamond_space* [online]. GitHub, 2020 [cit. 2021-05-07]. Dostupné z: https://github.com/MarketaJu/diamond_space/blob/master/chess.png.

[12] *Pclines-python/test.png at master · RomanJuranek/pclines-python* [online]. GitHub, 2020 [cit. 2021-05-07]. Dostupné z: https://github.com/RomanJuranek/pclines-python/blob/master/doc/test.png.

[13] Heinrich, J. a Weiskopf, D. State of the Art of Parallel Coordinates. In: Sbert, M. a Szirmay Kalos, L., ed. *Eurographics 2013 - State of the Art Reports*. The Eurographics Association, 2013. DOI: 10.2312/conf/EG2013/stars/095-116. ISSN 1017-4656.

[14] Hough, P. V. METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS. Prosinec 1962.

[15] Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S. et al. Learning to Parse Wireframes in Images of Man-Made Environments. In: *CVPR*. červen 2018, s. 626–635. DOI: 10.1109/CVPR.2018.00072.

[16] Matas, J., Galambos, C. a Kittler, J. Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*. 2000, sv. 78, č. 1, s. 119–137. DOI: https://doi.org/10.1006/cviu.1999.0831. ISSN 1077-3142. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1077314299908317.

[17] Newell, A., Yang, K. a Deng, J. *Stacked Hourglass Networks for Human Pose Estimation*. 2016.

[18] Sobel, I. a Feldman, G. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*. Leden 1973, s. 271–272.

[19] Tuytelaars, T., Proesmans, M. a Gool, L. J. V. The Cascaded Hough Transform as Support for Grouping and Finding Vanishing Points and Lines. In: *Proceedings of the International Workshop on Algebraic Frames for the Perception-Action Cycle*. Berlin, Heidelberg: Springer-Verlag, 1997, s. 278–289. AFPAC '97. ISBN 3540635173.

[20] Xue, N., Bai, S., Wang, F., Xia, G.-S., Wu, T. et al. Learning Attraction Field Representation for Robust Line Segment Detection. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, s. 1595–1603. DOI: 10.1109/CVPR.2019.00169.

[21] Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.-S. et al. Holistically-Attracted Wireframe Parsing. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, s. 2785–2794. DOI: 10.1109/CVPR42600.2020.00286.

[22] Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J. et al. PPGNet: Learning Point-Pair Graph for Line Segment Detection. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, s. 7098–7107. DOI: 10.1109/CVPR.2019.00727.