VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY

INSTITUTE OF MATHEMATICS

METODY MAPOVÁNÍ TEXTUR NA MRAČNA BODŮ

THE METHODS OF TEXTURE MAPPING ON POINT CLOUD

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAL KVĚTNÝ**

AUTHOR

VEDOUCÍ PRÁCE — **Mgr. JANA PROCHÁZKOVÁ, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. Michal Květný**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Metody mapování textur na mračna bodů

Stručná charakteristika problematiky úkolu:

Při získávání prostorových skenů je výsledný 3D objekt zobrazen pomocí triangulační sítě nebo pomocných plošek vytvořených z mračna bodů. Pro reálnou vizualizaci je vhodné připojit i skutečné barvy, které lze jednoduše získat pomocí fotografií. Matematickým problémem je nalezení korelace mezi prostorovými daty a fotografií a následné sesazení s vhodnými geometrickými transformacemi.

Cíle diplomové práce:

1. Nastudovat matematické metody vyhledávání korelací mezi mračnem bodů a fotografií – feature detection.
2. Seznámit se se zobrazovacími maticemi a jejich použitím při rekonstrukci.
3. Návrh algoritmu mapování textury na mračno bodů s použitím vybraných transfomací.
4. Testování na reálných datech získaných s pomocí 3D skeneru ATOS a fotografií strojírenských součástí.

Seznam doporučené literatury:

STAMOS, I.. Automated Registration of 3D-Range with 2D-Color Images: an Overview. In: 2010. 44th Annual Conference on Information Sciences and Systems (CISS) [online]. IEEE, 2010, 2010, s. 1-6 [cit. 2019-09-13]. DOI: 10.1109/CISS.2010.5464815. ISBN 978-1-4244-7416-5. Dostupné z: http://ieeexplore.ieee.org/document/5464815/

SZELISKI, R. Computer Vision: Algorithms and Applications. London: Springer, c2011. Texts in computer science. ISBN 978-1-84882-934-3.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Mgr. Petr Vašík, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Tato diplomová práce se zabývá mapováním textur získaných z běžných fotografií na trojrozměrná mračna bodů pořízená 3D skenerem. Práce navrhuje mapovací metodu skládající se z rekonstrukce řídkého mračna ze vstupních fotografií a následné registrace tohoto mračna se vstupním hustým mračnem užitím FPFH (Fast Point Feature Histogram) deskriptorů. Součástí práce je implementace navržené metody v programovacím jazyce C++, která je otestovaná na reálných datech.

Summary
This diploma thesis deals with a mapping of textures obtained from ordinary photographs onto the three-dimensional point clouds created by a 3D scanner device. The thesis proposes a mapping method that consists of reconstruction of a sparse point cloud from input photos and registration of the point cloud with an input dense point cloud using FPFH (Fast Point Feature Histogram) descriptors. The thesis also contains an implementation of the proposed method in C++ programming language which is tested on a real world data.

Klíčová slova
rekonstrukce scény, FPFH deskriptory, registrace mračen, mapování textur, projektivní geometrie, epipolární geometrie, ORB detektor

Keywords
scene reconstruction, FPFH descriptors, point cloud registration, texture mapping, projective geometry, epipolar geometry, ORB detector

KVĚTNÝ, MICHAL. *Metody mapování textur na mračna bodů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2022. 89 s. Vedoucí práce Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem diplomovou práci *Metody mapování textur na mračna bodů* vypracoval samostatně pod vedením Mgr. Jany Procházkové, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Bc. Michal Květný

Chtěl bych poděkovat své vedoucí Mgr. Janě Procházkové, Ph.D. za podporu a trpělivost při vedení tvorby mé diplomové práce. Rád bych také poděkoval své rodině, která při mně stála i v těch nejtěžších chvílích mého studia.

Bc. Michal Květný

Obsah

Úvod

Dnešní doba umožňuje dříve velmi nákladným technologiím rychle prostupovat do běžného života. K těmto technologiím patří například i 3D skenování. V této oblasti se vyskytuje celá řada přístupů, jak efektivně získat přesná a hustá mračna bodů reprezentující trojrozměrný objekt. Mezi běžně používané se řadí jak skenery kontaktní, které geometrii předmětu zaznamenávají mechanickým měřením, tak i bezkontaktní, které pro skenování používají nejčastěji světlo nebo zvuk odražený od povrchu tohoto předmětu.

U obou typů může být žádoucí doplnit geometrii předmětu informací o barvě povrchu. Tu je možné získat například pořízením fotografií. Některé optické skenery trojrozměrný objekt rekonstruují přímo ze skutečných fotografií, a proto je lze navrhnout tak, aby generovaná mračna tuto informaci rovnou obsahovala. Jiné ale mohou využívat například technologii měření odrazu laserového paprsku a jejich senzory tak žádnou informaci o skutečné barvě neposkytují.

Jako jedno z řešení se nabízí pevně spojit skenovací zařízení s klasickou kamerou a zajistit tak jejich známou vzájemnou polohu. Tento poměrně jednoduchý přístup má však několik nevýhod.

Laserové skenování klade specifické nároky na umístění skeneru, které se nemusí shodovat s požadavky na pořizování digitálních fotografií. Navíc je skenování pracný a časově náročný proces, zatímco fotografování je potřeba provést co nejrychleji za pokud možno konstantních světelných podmínek. Kvůli pevnému spojení obou zařízení není možné nastavení kamery přizpůsobit konkrétní scéně bez nutnosti rekalibrace. To vede k nižší kvalitě získaných snímků. V neposlední řadě tento přístup neřeší problematiku mapování textur, které je potřeba pořídit v jiném časovém okamžiku než během skenování. Může se jednat například o historické snímky, kterými chceme obarvit současnou scénu nebo třeba snímky z termokamery, kterými chceme sledovat vývoj změn teploty u nějaké strojírenské součásti.

Jiným řešením může být ruční sesazování obrázku s modelem, které je však velmi pomalé a náchylné na chyby, a tak své uplatnění nachází pouze ve specifických případech.

Proto je zřejmé, že automatické sesazení obrázku s mračnem bodů je důležitý problém, jehož řešení je aplikovatelné v řadě odvětvích.

K automatickému sesazení je možné přistoupit mnoha způsoby. První kategorie metod se pokouší samostatné snímky přímo mapovat na nebarevná trojrozměrná data [1, 2]. Tento přístup mnohdy využívá předchozí znalost typu zobrazované scény nebo jejího nasvícení. Druhá kategorie využívá hrubě obarvená trojrozměrná data a pouze scénu dobarvuje dalšími samostatnými snímky, přičemž používá texturu povrchu k nalezení vztahu s dalšími snímky [3, 4]. Poslední kategorie metod využívá sérii snímku pro rekonstrukci nové trojrozměrné scény, s jejíž pomocí následně obarvuje původní data [5, 6].

Cílem této práce je prezentovat metody používané pro automatické mapování textur. Tato práce navrhuje metodu pro obarvení mračna bodů sadou obrázku. Obrázky jsou nejdříve metodou *structure-from-motion* převedeny na mračno bodů s využitím znalostí epipolární geometrie. Obě mračna jsou následně sesazena registrací s pomocí FPFH deskriptorů. Pro řešení viditelnosti se využívá heuristická metoda. Navržená metoda je implementována v jazyce C++ s využitím knihoven OpenCV [7] a Open3D [8] a testována na několika datových vzorcích.

Práce je rozdělena do sedmi kapitol. Kapitola 1 se věnuje vlastnostem projektivní geometrie a prezentuje nutné teoretické základy pro tuto práci.

Kapitola 2 srovnává matematické a počítačové reprezentace dat, se kterými pracuje tato práce. V první části jsou popsány reprezentace digitálních obrazů. V části druhé je pak uveden základ reprezentace mračen bodů.

Kapitola 3 uvádí popis modelu kamery. Ten je odvozen s využitím poznatků o projektivní geometrii. Konec kapitoly se věnuje popisu kalibrace.

Kapitola 4 se zabývá metodami detekce významných bodů v obrazech i prostorových datech. Je v ní popsán princip deskriptorů a předvedeno fungování algoritmu ORB, který je použit pro implementaci metod navržených v této práci.

V kapitole 5 jsou popsány principy epipolární geometrie. Tato věda stojí na základech projektivní geometrie a popisuje vztah mezi dvěma kamerami. V této kapitole je rovněž odvozen princip rekonstrukce scény.

Kapitola 6 popisuje registraci dvou mračen bodů s využitím FPFH deskriptorů a objasňuje princip barvení mračna.

Poslední kapitola 7 se zabývá popisem implementace navrženého algoritmu a prezentací jeho výsledků při použití na několika datových vzorcích.

1. Projektivní geometrie

Geometrie, která je základem dnešní počítačové grafiky, byla používána už v dobách antického Řecka. Matematici jako Pythagoras, Theaitétos z Athén, Hippokrates z Chiu nebo Eudoxos z Knidu zřejmě znali, používali a rozvíjeli tehdejší geometrické znalosti. Byl to ale až Eukleidés, který poznatky této doby popsal ve svém díle Základy [9].

Zde Eukleidés mimo jiné položil základy axiomatické teorie rovinné a prostorové geometrie. Konstrukce v Eukleidově geometrii se omezují na ty, které lze provést pouze s použitím pravítka a kružítka. Ačkoliv se jeho systém pohledem moderní matematiky ukazuje jako neúplný, je dodnes tato kniha považována za nejznámější a nejvlivnější dílo starověké matematiky.

Modernější přístup k zavedení geometrie použil německý matematik David Hilbert, který na konci 19. století zavedl současnou rovinnou geometrii [10]. Jeho axiomatický systém se skládá ze tří axiomů incidence, čtyřech axiomů uspořádání, šesti axiomů shodnosti, dvou axiomů spojitosti a axiomu rovnoběžnosti. Tento systém lze dále rozšířit doplněním šesti dalších axiomů incidence tak, aby obsahoval i prostorovou geometrii.

V této kapitole bude zaveden pojem eukleidovského prostoru. Ten bude dále rozšířen na projektivní prostor. V něm budou zavedeny homogenní souřadnice a bude předvedeno jejich použití pro vybrané geometrické transformace.

Text v této kapitole předpokládá znalosti z oblasti lineární algebry, zejména pak znalost vektorových prostorů, alespoň v rozsahu [11]. Zpracování této kapitoly vychází z [11, 12, 13, 14, 15].

1.1. Eukleidovské prostory

Jak již bylo řečeno, moderní geometrie stojí na několika Hilbertových axiomech. Zde bude uveden pouze jeden, který bude později nutné změnit pro korektní zavedení projektivních prostorů. Zbytek axiomů lze nalézt například v [12].

Axiom 1.1.1 (Eukleidův axiom rovnoběžnosti)**.** *Nechť p je přímka a A je bod, který na ní neleží. Pak lze bodem A vést právě jednu přímku, která s přímkou p nemá žádný společný bod.*

Poznámka. Tento axiom byl v trochu pozměněné formě obsažen už v původních Eukleidových postulátech. Pro svou formulaci byl některými mysliteli považován spíše za větu, kterou by mělo být možné z ostatních axiomů dokázat. Až později se ukázalo, že je skutečně na ostatních axiomech nezávislý [12] a dnes je běžně nahrazován v tzv. neeukleidovských geometriích [16].

Jedním z modelů, který axiomy geometrie realizuje, je syntetická geometrie. V ní si body představujeme jako bezrozměrné tečky, přímky jako tenké rovné čáry a roviny jako rovné plochy nulové tloušťky. Tato představa byla v podstatě jedinou až do 17. století, kdy René Descartes zavedl pravoúhlou souřadnicovou soustavu (dnes známou jako kartézskou soustavu souřadnic podle latinského přepisu jeho jména) a pomohl tak vzniku analytického modelu geometrie [12].

V analytické geometrii jsou všechny geometrické útvary reprezentovány nějakými algebraickými ekvivalenty. Body jsou uspořádané dvojice (v rovině), resp. trojice (v prostoru)

reálných čísel, přímky (v rovině) jsou lineární rovnice o dvou neznámých, případně (v prostoru) soustavy dvou rovnic o třech neznámých apod.

Pro korektní zavedení takové geometrie potřebujeme tento model popsat nějakou vhodnou matematickou strukturou. Ačkoliv se nabízí použít k tomuto popisu vektorový prostor, výhodnější strukturou je prostor afinní, který umožňuje rozlišovat mezi body a směry.

Definice 1.1.2 (Afinní prostor)**.** Nechť A je neprázdná množina a V je vektorový prostor nad polem $(\mathbb{R}, +, \cdot)$. Zavedeme binární operaci $- : A \times A \to V$ s vlastnostmi

1. $\forall X, Y, Z \in A$ platí $(Y - X) + (Z - Y) = (Z - X)$,
2. $\exists O \in A$ takový, že $\forall \mathbf{x} \in V \exists! X \in A$, pro který platí $X - O = \mathbf{x}$.

Trojici $\mathcal{A} = (A, V, -)$ nazýváme afinní prostor. Prvky množiny A nazýváme body. Vektor $\mathbf{x}$ je polohový vektor bodu X. Příslušný vektorový prostor V se nazývá zaměření afinního prostoru. Je-li přirozené číslo n dimenzí V, pak říkáme, že afinní prostor $\mathcal{A}$ je n-rozměrný.

Zvlášť pak jednorozměrný (resp. dvojrozměrný) afinní prostor nazýváme afinní přímkou (resp. afinní rovinou).

Poznámka. V našem případě budou roli bodů budou hrát výhradě uspořádané dvojice, případně uspořádané trojice reálných čísel a k nim budeme vždy brát buď dvojrozměrný, případně trojrozměrný vektorový prostor V.

Afinní prostor splňuje téměř všechny axiomy geometrie. Jediné chybějící jsou axiomy shodnosti. Ty však lze do afinního prostoru jednoduše doplnit zavedením skalárního (vnitřního) součinu.

Definice 1.1.3 (Skalární součin)**.** Nechť V je vektorový prostor. Zobrazení $\langle , \rangle : V \times V \to \mathbb{R}_0^+$ se nazývá skalární součin, pokud splňuje podmínky

1. $\forall \mathbf{u}, \mathbf{v} \in V : \langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$,
2. $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V : \langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$,
3. $\forall \mathbf{u}, \mathbf{v} \in V, a \in \mathbb{R} : \langle a\mathbf{u}, \mathbf{v} \rangle = a\langle \mathbf{u}, \mathbf{v} \rangle$,
4. $\langle \mathbf{o}, \mathbf{o} \rangle = 0$ tehdy a jen tehdy, když $\mathbf{o}$ je nulový vektor ve V.

Vektorový prostor s takto zavedeným zobrazením nazýváme vektorový prostor se skalárním součinem nebo též unitární prostor. Je-li z kontextu zřejmé, že se jedná o skalární součin, pak jej též zjednodušeně zapisujeme $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{uv}$.

Přítomnost skalárního součinu umožňuje jednoduše definovat velikost (normu) vektoru a úhel mezi dvěma vektory.

Definice 1.1.4 (Norma a úhel vektorů)**.** Nechť V je vektorový prostor se skalárním součinem a buď $\mathbf{u}, \mathbf{v} \in V$. Pak číslo

$$\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle} \tag{1.1}$$

nazýváme norma (velikost) vektoru $\mathbf{u}$.

Dále nechť $\|\mathbf{u}\| \neq 0$ a $\|\mathbf{v}\| \neq 0$, pak tyto dva vektory určují úhel, jehož velikost φ splňuje následující rovnost

$$\cos \varphi = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}. \tag{1.2}$$

Poznamenejme, že vektorový prostor se skalárním součinem je speciálním případem metrického prostoru. Jedná se o vektorový prostor, na kterém existuje zobrazení, které umožňuje měřit vzdálenost mezi vektory. Tomuto zobrazení říkáme metrika.

Definice 1.1.5 (Metrika)**.** Nechť V je vektorový prostor a zobrazení $\rho : V \times V \to \mathbb{R}_0^+$. Pak toto zobrazení nazveme metrika, jestliže splňuje podmínky

1. $\forall \mathbf{u}, \mathbf{v} \in V : \rho(\mathbf{u}, \mathbf{v}) = \rho(\mathbf{v}, \mathbf{u})$,
2. $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V : \rho(\mathbf{u}, \mathbf{w}) \leq \rho(\mathbf{u}, \mathbf{v}) + \rho(\mathbf{v}, \mathbf{w})$,
3. $\forall \mathbf{u}, \mathbf{v} \in V : \rho(\mathbf{u}, \mathbf{v}) = 0 \iff \mathbf{u} = \mathbf{v}$.

Vektorový prostor se zobrazením ρ, pak nazveme metrický prostor.

Pojem metriky má v matematice velký význam a budeme jej hojně využívat v průběhu této práce. Jak již bylo zmíněno, unitární prostor je speciálním případem prostoru metrického. To podtrhuje následující věta.

Věta 1.1.6. *Nechť V je unitární prostor a nechť $\mathbf{u}, \mathbf{v} \in V$, pak zobrazení*

$$\rho(\mathbf{u}, \mathbf{v}) = \|\mathbf{v} - \mathbf{u}\| \tag{1.3}$$

splňuje axiomy metriky.

Definice 1.1.3 a 1.1.4 se přirozeně přenesou do afinního prostoru $\mathcal{A} = (A, V, -)$ tak, že se skalární součin zavede na jeho zaměření V. Takový prostor budeme nazývat afinní prostor se skalárním součinem. Shodnost pak definujeme tak, že dvojice úseček AB a CD se shoduje právě tehdy, když příslušné vektory $(B-A)$ a $(D-C)$ mají stejnou normu. Délku úsečky AB (normu příslušného vektoru $(B-A)$) budeme dále označovat $|AB| = ||B-A||$. Afinní prostor, na jehož zaměření je definován vnitřní součin tedy již splňuje všechny požadované axiomy, a je tak modelem eukleidovské geometrie [12].

Definice 1.1.7 (Eukleidovský prostor)**.** Afinní prostor se skalárním součinem nazýváme eukleidovský prostor a značíme $\mathbb{E}^n$, kde n je dimenze tohoto prostoru.

Poznámka. Množinu bodů A v prostoru $\mathbb{E}^n$ a prostor $\mathbb{E}^n$ někdy pro zjednodušení zápisů ztotožňujeme.

Nevýhodou tohoto popisu prostoru je složitost vyjádření některých běžně používaných zobrazení $\mathbb{E}^n \to \mathbb{E}^n$ jako je například rotace kolem obecného bodu nebo osová souměrnost podle obecné osy. Tyto zobrazení se řadí do třídy projektivních zobrazení, což jsou taková, ve kterých je obrazem přímky zase přímka. Obecně lze taková zobrazení popsat pomocí lineárních rovnic [12]. Více o tomto typu zobrazení bude uvedeno v části 1.2.1. Uveďme pouze, jak by vypadal obecný zápis projektivního zobrazení v $\mathbb{E}^2$.

Nechť $\mathcal{F} : \mathbb{E}^2 \to \mathbb{E}^2$ je projektivní zobrazení a $\mathcal{F}([x_1, x_2]) = [x_1', x_2']$, pak souřadnice obrazu jsou určeny vztahy

$$x_1' = a_{11}x_1 + a_{12}x_2 + t_1, \tag{1.4}$$
$$x_2' = a_{21}x_1 + a_{22}x_2 + t_2. \tag{1.5}$$

Oba tyto vztahy (1.4, 1.5) lze popsat souhrnně maticovým zápisem

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{t}, \tag{1.6}$$

kde vektory $\mathbf{x}$ a $\mathbf{x}'$ jsou po řadě polohové vektory bodů $[x_1, x_2]$ a $[x'_1, x'_2]$, prvky matice $\mathbf{A}$ jsou

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{1.7}$$

a vektor $\mathbf{t}$ je vektor translace, pro který platí

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \tag{1.8}$$

Z maticového zápisu je patrné, že translace (posunutí) je realizována přičtením vektoru. To značně komplikuje skládání více různých zobrazení. Řešení tohoto problému nabízí rozšířený eukleidovský prostor s homogenními souřadnicemi.

1.2. Projektivní prostory

Pro rozšíření Eukleidova prostoru je nutné původní seznam axiomů doplnit o nový, projektivní axiom.

Axiom 1.2.1 (Projektivní)**.** *Dvě přímky ležící v téže rovině mají společný bod.*

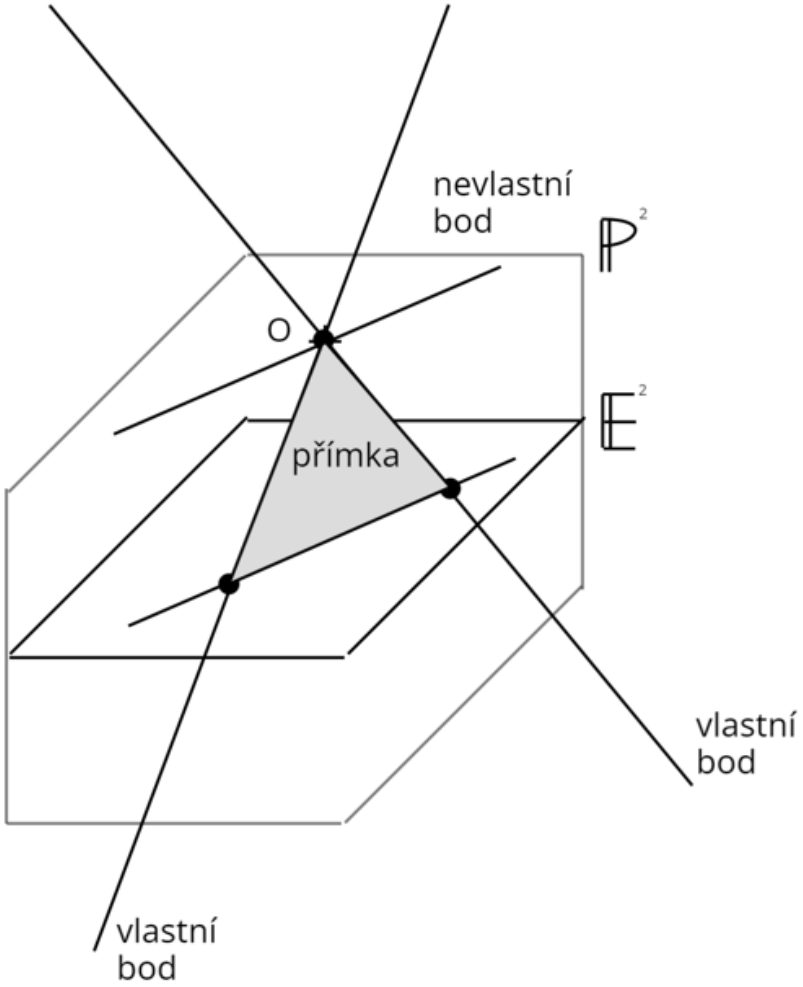
Tento axiom je ve sporu s axiomem 1.1.1, to však lze velmi snadno napravit tak, že budeme rozlišovat mezi body vlastními a nevlastními. Axiom 1.1.1 se pak upřesní následovně.

Axiom 1.2.2 (Eukleidův axiom rovnoběžnosti v projektivním prostoru)**.** *Nechť p je přímka a A je bod, který na ní neleží. Pak lze bodem A vést právě jednu přímku, která s přímkou p nemá společný žádný **vlastní** bod.*

Přímky uváděné v axiomu 1.2.2 jsou rovnoběžky a nemají společný žádný vlastní bod. Podle axiomu 1.2.1 však nějaký společný bod mít musí, ten nazveme nevlastním bodem. Prostor, který doplníme o axiom 1.2.1 a ve kterém axiom 1.1.1 nahradíme axiomem 1.2.2 budeme nazývat projektivní prostor a budeme jej značit $\mathbb{P}^n$. Dále budeme o prostoru $\mathbb{P}^2$ mluvit jako o projektivní rovině a o prostoru $\mathbb{P}^3$ jen jako o projektivním prostoru [13]. Vyšší dimenze, ač možné, v kontextu této práce nemá smysl uvažovat.

Syntetickým modelem projektivní roviny může být trojrozměrný eukleidovský prostor, ve kterém si projektivní body představujeme jako přímky, které procházejí jedním společným bodem O tohoto eukleidovského prostoru. Množina všech bodů projektivního prostoru je tak trsem přímek. Libovolnou eukleidovskou rovinu, která neprochází bodem O, si lze představit jako klasickou eukleidovskou rovinu $\mathbb{E}^2$. Průsečíky přímek různoběžných s touto rovinou jsou modely vlastních bodů v $\mathbb{E}^2$. Přímky rovnoběžné s touto rovinou s ní patrně nemají žádný společný bod, a tedy nemají v $\mathbb{E}^2$ klasický model bezrozměrné tečky. Tyto body však mohou v $\mathbb{E}^2$ mít význam směru [13].

Přímkou projektivního prostoru je rovina určená dvěma různými projektivními body, přímkami v eukleidovském prostoru. Dvě takto definované roviny mají vždy společnou eukleidovskou přímku, projektivní bod. Společným bodem dvou rovnoběžných projektivních



Obrázek 1.1: Syntetický model projektivní roviny.

přímek je nevlastní bod, jehož odpovídající eukleidovská přímka je rovnoběžná s rovinou znázorňující prostor $\mathbb{E}^2$. Všechny tyto nevlastní body leží na jediné projektivní přímce, eukleidovské rovině, která je se zvolenou eukleidovskou rovinou $\mathbb{E}^2$ rovnoběžná a která prochází bodem O. Této přímce říkáme nevlastní přímka. Až na tuto jedinou nevlastní přímku všechny projektivní přímky mají s rovinou $\mathbb{E}^2$ společnou průsečnici, jež je modelem odpovídající přímky v $\mathbb{E}^2$ [13]. Představa syntetického modelu projektivní roviny je lépe patrná z obrázku 1.1.

Pro zavedení analytické projektivní geometrie v rovině uvedený model doplníme kartézskou souřadnou soustavou $(O, \mathbf{i}, \mathbf{j}, \mathbf{k})$, kde počátkem je společný bod eukleidovských přímek, které jsou modelem projektivních bodů. Osy značíme po řadě x, y a ω.

Zřejmě tedy bodem X projektivního prostoru je množina vektorů

$$X = \{k \cdot (x_1, x_2, \omega_X)^\intercal \neq \mathbf{o} \mid k \in \mathbb{R}\}. \tag{1.9}$$

Vektory patřící do této množiny nazýváme reprezentanty bodu X. Jako množinu všech nevlastních bodů, nevlastní přímku, zvolíme rovinu $\omega = 0$. Eukleidovskou rovinou reprezentující „klasický“ prostor $\mathbb{E}^2$ bude rovina $\omega = 1$.

Vlastním bodem je bod, jehož odpovídají eukleidovská přímka protíná rovinu $\mathbb{E}^2$. To však může nastat pouze pokud $\omega_X \neq 0$. To ale znamená, že zvolením $k = \frac{1}{\omega_X}$ získáme reprezentant $(\frac{x_1}{\omega_X}, \frac{x_2}{\omega_X}, 1)^\intercal$, který budeme nazývat eukleidovský reprezentant.

Eukleidovská přímka odpovídající nevlastnímu bodu leží v rovině $\omega = 0$. To znamená, že jeho reprezentanty jsou tvaru $(kx_1, kx_2, 0)^\intercal$.

Konstrukce projektivního prostoru $\mathbb{P}^3$ probíhá podobně, ale vyžaduje již čtyřrozměrný eukleidovský prostor. Uveďme tedy jen, že analogicky k $\mathbb{P}^2$ mají vlastní body reprezentanty tvaru $(\frac{x_1}{\omega_X}, \frac{x_2}{\omega_X}, \frac{x_3}{\omega_X}, 1)^\intercal$. Nevlastní body lze reprezentovat vektory $(kx_1, kx_2, kx_3, 0)^\intercal$.

Kartézské souřadnice libovolného reprezentanta bodu X v projektivním prostoru nazýváme homogenní souřadnice a pro přehlednost budeme značit $\mathbf{X}$, abychom zdůraznili, že lze tento bod chápat jako matici typu $(n+1) \times 1$, kde n je dimenze daného prostoru. Je-li navíc poslední souřadnice rovna 1 nebo 0, pak hovoříme o normalizovaných homogenních souřadnicích.

V dalších částech se budeme blíže věnovat vybraných vlastnostem prostorů $\mathbb{P}_2$ a $\mathbb{P}_3$.

1.2.1. Projektivní rovina

Stejně jako jsme v minulé části popsali homogenní souřadnice bodu v projektivní rovině, můžeme i každé projektivní přímce projektivní roviny přiřadit vlastní homogenní souřadnice. V eukleidovské rovině $\mathbb{E}^2$ je každou přímku p možné vyjádřit obecnou rovnicí

$$p : ax + by + c = 0, \tag{1.10}$$

ve které $a, b, c \in \mathbb{R}$. Přímce popsané rovnicí (1.10) přiřadíme vektor $\mathbf{p} = (a, b, c)^\intercal$, který budeme nazývat homogenní souřadnice přímky p. Je patrné, že každá přímka projektivní roviny má nekonečně mnoho homogenních souřadnic ve tvaru $(ka, kb, kc)^\intercal$, kde $k \neq 0$ je reálné číslo [14].

Důsledkem tohoto přístupu k přímkám je fakt, že mezi přímkou a bodem platí v projektivní rovině vztah, který nazýváme princip duality [14]. Ten nám umožňuje ve všech tvrzeních nahrazovat slovo *bod* slovem *přímka* a obráceně, aniž by došlo ke změně zbytku tvrzení. Předveďme tento princip na příkladu tvrzení: „*Dvěma různými body prochází právě jedna přímka.*“ Aplikací principu duality je možné stejně říct: „*Dvě různé přímky prochází společně právě jedním bodem.*“

Díky homogenním souřadnicím můžeme snadno vyjádřit, že bod X s homogenními souřadnicemi $\mathbf{X}$ leží na přímce p s homogenními souřadnicemi $\mathbf{p}$ právě tehdy, když

$$\mathbf{X}^\intercal \mathbf{p} = 0. \tag{1.11}$$

Důkaz tohoto tvrzení se provede triviálně převedením maticového vztahu na vztah v rovnici (1.10). Ve smyslu duality také jednoduše obdržíme ekvivalentní rovnici

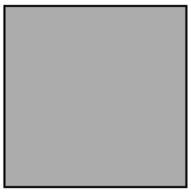
$$\mathbf{p}^\intercal \mathbf{X} = 0. \tag{1.12}$$

Máme-li dvě přímky p a q, pak užitím jejich homogenních souřadnic $\mathbf{p}$ a $\mathbf{q}$ lze jednoduše nalézt jejich průsečík P vztahem

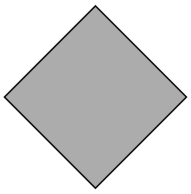
$$\mathbf{P} = \mathbf{p} \times \mathbf{q}, \tag{1.13}$$

ve kterém symbolem $\mathbf{P}$ rozumíme homogenní souřadnice průsečíku P. Dualitou naopak získáme vztah pro výpočet přímky p procházející dvěma body A a B

$$\mathbf{p} = \mathbf{A} \times \mathbf{B}. \tag{1.14}$$

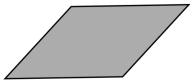


(a) Bez transformace.

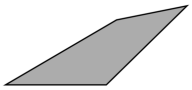


(b) Shodná zobrazení – transformovaný objekt má stejnou geometrii, liší se pouze jeho poloha v rovině.

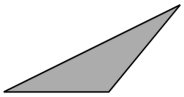
(c) Podobnostní zobrazení – transformovaný objekt může mít jinou velikost.



(d) Afinní zobrazení – transformovaný objekt má jinou geometrii, ale rovnoběžky zůstávají rovnoběžkami.



(e) Kolineární zobrazení – transformovaný objekt má jinou geometrii, ale přímky setrvávají přímkami.



(f) Projektivní zobrazení – některé přímky se mohou zobrazit na body.

Obrázek 1.2: Účinky různých typů projektivních zobrazení na dvojrozměrný objekt.

Podotkneme, že vektorový součin v $\mathbb{R}^3$ lze realizovat rovněž násobením matic. Mějme vektory $\mathbf{a} = (a_1, a_2, a_3)^\intercal$ a $\mathbf{b} = (b_1, b_2, b_3)^\intercal$ z $\mathbb{R}^3$ a definujme antisymetrickou matici příslušnou vektoru $\mathbf{a}$

$$[\mathbf{a}]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}, \tag{1.15}$$

pak

$$\mathbf{a} \times \mathbf{b} = (a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1)^\intercal = [\mathbf{a}]_\times \mathbf{b} = (\mathbf{a}^\intercal [\mathbf{b}]_\times)^\intercal. \tag{1.16}$$

Matice $[\mathbf{b}]_\times$ je definována obdobně jako matice $[\mathbf{a}]_\times$. Tento způsob zápisu bude využit v kapitole 5, která je věnována epipolární geometrii.

Použijme nyní vztahy (1.13) a (1.14) pro ověření poznatků z části 1.2. Uvažujme dvě rovnoběžné přímky $\mathbf{p} = (a, b, c_1)^\intercal$ a $\mathbf{q} = (a, b, c_2)^\intercal$. Dosazením do (1.13) získáme průsečík těchto přímek, kterým je bod $\mathbf{P} = (b, -a, 0)^\intercal$. Jak je zřejmé z třetí homogenní souřadnice tohoto bodu, jedná se o nevlastní bod, což je konzistentní s představou dvou přímek protínajících se v nekonečnu.

Vezmeme-li naopak dva nevlastní body $\mathbf{A} = (a, b, 0)^\intercal$ a $\mathbf{B} = (c, d, 0)^\intercal$, pak tyto dva body spolu určují přímku danou rovnicí (1.14) $\mathbf{p} = (0, 0, ad - bc)^\intercal$, jejíž normované souřadnice $\mathbf{p}_\infty = (0, 0, 1)^\intercal$ odpovídají nevlastní přímce. Ani tento výsledek není ve sporu s představou syntetického modelu z části 1.2.

Dále využijeme znalosti o projektivní rovině pro popis vybraných důležitých transformací tohoto prostoru. Všechny níže uvedené typy transformací je možné pozorovat na obrázku 1.2.

Projektivní zobrazení

Důležitým objektem zájmu v projektivní geometrii jsou vlastnosti zobrazení, které nazýváme projektivní zobrazení (promítání). Slovem promítání rozumíme zobrazení, v němž obrazem přímky je opět přímka, případně bod. Speciálním typem promítání, které má pro tuto práci ještě větší význam, je kolineární zobrazení. Jedná se o zobrazení, které je projektivní a navíc prosté, tedy pro něj platí, že každou přímku zobrazí jako přímku [13].

Definice 1.2.3 (Kolineární zobrazení)**.** Zobrazení $h : \mathbb{P}^2 \to \mathbb{P}^2$ nazýváme kolineární, jestliže je invertibilní a splňuje vlastnost, že body X_1, X_2 a X_3 leží na jedné přímce právě tehdy, když i body $h(X_1)$, $h(X_2)$ a $h(X_3)$ leží na jedné přímce.

Poznámka. Značení kolineárního zobrazení h v předchozí definici není náhodné. Jedná se o zkratku jiného běžně používaného názvu kolineárního zobrazení, kterým je homografie.

Dodejme, že množina všech kolineárních zobrazení je uzavřená na inverzi a skládání, a tedy tvoří podgrupu v grupě všech zobrazení. Rovněž tak platí, že každé kolineární zobrazení nad projektivní rovinou je možné popsat pomocí matice.

Věta 1.2.4. *Zobrazení $h : \mathbb{P}^2 \to \mathbb{P}^2$ je kolineární tehdy a jen tehdy, když existuje regulární matice $\mathbf{H}$ typu 3×3 taková, že bod $h(X)$ v homogenních souřadnicích odpovídá bodu $\mathbf{HX}$.*

Uveďme alespoň důkaz implikace zprava doleva, druhý směr důkazu je podstatně složitější.

Důkaz. Nechť body X_1, X_2 a X_3 leží na přímce $\mathbf{p}$, a tedy $\mathbf{p}^\intercal \mathbf{X_i} = 0$ pro všechny $i = 1, 2, 3$. Dále nechť $\mathbf{H}$ je regulární matice typu 3×3, ke které tedy nutně existuje inverzní matice $\mathbf{H^{-1}}$. Dosazením do rovnice přímky $\mathbf{p}^\intercal \mathbf{X_i} = \mathbf{p}^\intercal \mathbf{H^{-1}} \mathbf{HX_i} = \mathbf{H^{-\intercal}} \mathbf{pHX_i} = 0$ snadno ověříme, že body $\mathbf{HX_i}$ leží na přímce $\mathbf{H^{-\intercal}p}$, a tak je zobrazení dané maticí $\mathbf{H}$ kolineární [14]. □

Tato věta nám umožňuje provést alternativní definici kolineárního zobrazení.

Definice 1.2.5 (Kolineární zobrazení)**.** Rovinné kolineární zobrazení je lineární zobrazení homogenních trojrozměrných vektorů popsané regulární maticí a vztahem

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \omega'_X \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \omega_X \end{pmatrix}. \tag{1.17}$$

Všimněme si, že zobrazení h popsané maticí $\mathbf{H}$ v rovnici (1.17) nezávisí na vynásobení této matice nenulovým koeficientem, proto o této matici někdy hovoříme jako o homogenní matici (ve stejném smyslu jako hovoříme o homogenních souřadnicích bodů a přímek). Taková matice má pouze 8 stupňů volnosti (navzájem nezávislých prvků) a lze ji určit při znalosti čtyř bodů a jejich obrazů [14].

Jediným invariantem tohoto typu zobrazení tohoto typu je tzv. dvojpoměr čtyř různých bodů. Dvojpoměr různých bodů A, B, C a D definujeme jako číslo

$$\delta(A, B, C, D) = \frac{|AB||CD|}{|AC||BD|}. \tag{1.18}$$

Shodná zobrazení

Speciálním případem kolineárního zobrazení je zobrazení shodné (izometrické). Vlastností tohoto zobrazení je, že zachovává eukleidovskou metriku. Toto zobrazení je popsáno maticovou rovnicí

$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} \psi\cos\varphi & -\sin\varphi & t_x \\ \psi\sin\varphi & \cos\varphi & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}, \tag{1.19}$$

za podmínky $\psi = \pm 1$. V případě, že ψ se rovná jedné, pak toto zobrazení zachovává orientaci a jedná se pouze o kombinaci rotace a translace. V opačném případě se jedná o zrcadlové zobrazení a kombinuje zrcadlení, rotaci a translaci [14].

Konkrétně kombinaci rotace a translace také nazýváme eukleidovské zobrazení a toto zobrazení je modelem pohybu pevného tělesa. V praxi se jedná o nejužitečnější transformace. Rovinné eukleidovské zobrazení lze maticově zapsat také jako

$$\mathbf{X}' = \mathbf{H_E X} = \begin{pmatrix} \mathbf{R_{2\times 2}} & \mathbf{t_{2\times 1}} \\ \mathbf{0_{1\times 2}} & 1 \end{pmatrix} \mathbf{X}, \tag{1.20}$$

kde matice $\mathbf{R_{2\times 2}}$ je rovinná ortogonální rotační matice typu 2×2, $\mathbf{t_{2\times 1}}$ je dvojrozměrný vektor translace a $\mathbf{0_{1\times 2}}$ je matice nul typu 1×2. Rovinné eukleidovské zobrazení má tři stupně volnosti a lze tak určit ze znalosti transformace dvou bodů. Toto zobrazení zachovává délky úseček, velikosti úhlů a stejně tak i obsahy [14].

Dodejme, že izometrická zobrazení tvoří podgrupu, jen pokud zachovávají orientaci. Ostatní podgrupu netvoří.

Podobnostní zobrazení

Méně striktním příkladem kolineárního zobrazení je podobnost. Toto zobrazení zachovává velikost úhlů. V principu se jedná o kombinaci izometrického zobrazení s izotropickou (rovnoměrnou vzhledem ke směru) změnou měřítka. Pokud pomineme možnost zrcadlení, lze podobnost v rovině popsat vztahem

$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} s\cos\varphi & -s\varphi & t_x \\ s\sin\varphi & s\varphi & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}. \tag{1.21}$$

Kompaktněji lze toto zobrazení zapsat také jako

$$\mathbf{X}' = \mathbf{H_S X} = \begin{pmatrix} s\mathbf{R_{2\times 2}} & \mathbf{t_{2\times 1}} \\ \mathbf{0_{1\times 2}} & 1 \end{pmatrix} \mathbf{X}, \tag{1.22}$$

kde se označení matic shoduje s označením ve vztahu (1.20) a skalár s je koeficient izotropické změny měřítka. Rovinná podobnost má čtyři stupně volnosti a lze tak stále určit z korespondence mezi dvěma body. Invariantem tohoto zobrazení jsou velikosti úhlů a poměry délek [14].

Afinní zobrazení

Afinní zobrazení, nebo též afinita, je regulární lineární zobrazení následované translací. Maticově lze zapsat jako

$$\begin{pmatrix} x'_1 \\ x'_2 \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}, \tag{1.23}$$

případně kompaktně také jako

$$\mathbf{X}' = \mathbf{H_A X} = \begin{pmatrix} \mathbf{A_{2\times 2}} & \mathbf{t_{2\times 1}} \\ \mathbf{0_{1\times 2}} & 1 \end{pmatrix} \mathbf{X}, \tag{1.24}$$

kde značení je opět shodné jako ve vztahu (1.20) a matice $\mathbf{A_{2\times 2}}$ je regulární matice typu 2×2. Rovinná afinita má 6 stupňů volnosti a je tak možné ji určit třemi vzájemně si odpovídajícími body [14].

Pro lepší pochopení vlastnosti takového zobrazení je možné matici $\mathbf{A_{2\times 2}}$ užitím SVD [17] rozkladu rozložit na součin matic rovinné rotace a matice anizotropní (nerovnoměrné vzhledem ke směru) změny měřítka

$$\mathbf{A_{2\times 2}} = \mathbf{R_{2\times 2}}(\varphi)\mathbf{R_{2\times 2}}(-\theta)\mathbf{D_{2\times 2}}\mathbf{R_{2\times 2}}(\theta), \tag{1.25}$$

kde matice $\mathbf{R_{2\times 2}}(\varphi)$ a $\mathbf{R_{2\times 2}}(\theta)$ symbolizují po řadě rotace o úhel φ a θ a $\mathbf{D_{2\times 2}}$ je diagonální matice

$$\mathbf{D_{2\times 2}} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}. \tag{1.26}$$

Jedná se tedy o složení rotace o úhel θ, anizotropickou změnu měřítka koeficienty λ_1 a λ_2 ve směrech otočených os x a y, rotaci zpět o úhel $-\theta$ a další rotaci o úhel φ.

Tato transformace zachovává poměry obsahů, poměry délek rovnoběžných úseček a rovnoběžnosti přímek [14].

1.2.2. Projektivní prostor

Zatímco v projektivní rovině $\mathbb{P}^2$ jsme snadno zavedli homogenní souřadnice bodů a přímek, v projektivním prostoru $\mathbb{P}^3$ je jednoduché definovat homogenní souřadnice bodů a rovin. Definice homogenních souřadnic přímky v projektivním prostoru je složitější a v této práci se jí nebudeme zabývat, je však možné ji nalézt například v [14]. Způsob definice homogenních souřadnic bodů je totožný jako v části 1.2.

Rovina σ lze v eukleidovském prostoru $\mathbb{E}^3$ popsat obecnou rovnicí

$$\sigma : ax + by + cz + d = 0. \tag{1.27}$$

Rovině popsané rovnicí (1.27) přiřadíme vektor $\boldsymbol{\sigma} = (a, b, c, d)^\mathsf{T}$, který budeme nazývat homogenní souřadnice roviny σ a podobně jako u přímek v projektivní rovině, ani zde nemá vliv jeho vynásobení nenulovou konstantou. Rovina má tak nekonečně mnoho homogenních souřadnic tvaru $(ka, kb, kc, kd)^\mathsf{T}$, kde $k \neq 0$ je reálné číslo [14].

V projektivním prostoru existuje podobně jako v projektivní rovině dualita mezi bodem a rovinou. Obdobou rovnic (1.11) a (1.12) jsou zde rovnice

$$\mathbf{X}^\mathsf{T}\boldsymbol{\sigma} = 0, \tag{1.28}$$

$$\boldsymbol{\sigma}\mathbf{X}^\mathsf{T} = 0. \tag{1.29}$$

Tři body A, B a C, které nejsou kolineární, zřejmě společně určují rovinu. Všechny tři takové body musí splňovat rovnici (1.28), a tedy musí platit i maticová rovnice

$$\begin{pmatrix} \mathbf{A}^\intercal \\ \mathbf{B}^\intercal \\ \mathbf{C}^\intercal \end{pmatrix} \boldsymbol{\sigma} = \mathbf{o}. \tag{1.30}$$

Homogenní souřadnice roviny tak získáme jako jádro lineární zobrazení definovaného maticí

$$\begin{pmatrix} \mathbf{A}^\intercal \\ \mathbf{B}^\intercal \\ \mathbf{C}^\intercal \end{pmatrix}. \tag{1.31}$$

V případě, že body A, B a C leží na jedné přímce, pak je jádro tohoto zobrazení dvojrozměrné a popisuje svazek rovin, které mají společnou jednu přímku, na které leží body A, B a C.

Vzhledem k dualitě bodů a rovin můžeme rovněž určit společný bod třech různých rovin z maticové rovnice

$$\begin{pmatrix} \boldsymbol{\sigma}^\intercal \\ \boldsymbol{\pi}^\intercal \\ \boldsymbol{\rho}^\intercal \end{pmatrix} \mathbf{X} = \mathbf{o} \tag{1.32}$$

obdobným způsobem. Pokud roviny nejsou různé, ale mají společnou přímku, pak je jádro dimenze dva a určuje přímku, která je průsečnicí těchto rovin.

Projektivní transformace

Projektivní zobrazení v projektivním prostoru $\mathbb{P}^3$ mají obdobné vlastnosti jako jejich protějšky v projektivní rovině $\mathbb{P}^2$ a lze je obecně popsat maticovým zápisem

$$\mathbf{X}' = \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ \omega'_X \end{pmatrix} = \mathbf{H}\mathbf{X} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \omega_X \end{pmatrix}. \tag{1.33}$$

Je-li matice $\mathbf{H}$ regulární, hovoříme opět o zobrazeních kolineárních. Matice $\mathbf{H}$ má obecně 15 stupňů volnosti [14].

Shodná zobrazení

Nejjednodušším typem zobrazení v projektivním prostoru jsou podobně jako v projektivní rovině shodná zobrazení. Omezíme-li se pouze na eukleidovská zobrazení (tj. ta, která neobsahují zrcadlení), pak má matice $\mathbf{H}$ tvar

$$\mathbf{H_E} = \begin{pmatrix} \mathbf{R_{3\times 3}} & \mathbf{t_{3\times 1}} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix}. \tag{1.34}$$

V ní je matice $\mathbf{R_{3\times 3}}$ rotační matice řádu 3 a $\mathbf{t_{3\times 1}}$ trojrozměrný vektor translace. Matice $\mathbf{0_{1\times 3}}$ je maticí nul typu 1×3.

Tento typ transformace má celkem šest stupňů volnosti a je určen zobrazením třech bodů, které neleží na jedné přímce [14].

Podobnostní zobrazení

Podobnostní zobrazení jsou dalším ekvivalentem zobrazení v projektivní rovině. Jim příslušná matice je dána

$$\mathbf{H_S} = \begin{pmatrix} s\mathbf{R_{3\times 3}} & \mathbf{t_{3\times 1}} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix}, \tag{1.35}$$

ve které značení odpovídá značení v rovnici (1.34) a s je skalární koeficient změny měřítka. Podobnostní zobrazení má sedm stupňů volnosti a je tak stejně jako eukleidovské zobrazení určeno třemi body [14].

Afinní zobrazení

Posledním zobrazením, kterým se budeme zabývat je zobrazení afinní. To je popsáno maticí

$$\mathbf{H_A} = \begin{pmatrix} \mathbf{A_{3\times 3}} & \mathbf{t_{3\times 1}} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix}. \tag{1.36}$$

Matice $\mathbf{A_{3\times 3}}$ je regulární matice typu 3×3, přičemž zbytek označení se shoduje se značením v rovnici (1.34). Prostorová afinita má 12 stupňů volnosti a je možné ji jednoznačně určit z korespondence mezi čtyřmi body, ze kterých žádné tři neleží v jedné přímce [14].

2. Počítačová reprezentace obrazů a mračen bodů

Pro počítačové zpracování dat je vždy nutné najít jejich vhodnou počítačovou reprezentaci. Ta musí data uchovávat v požadované kvalitě a umožňovat rychlý a snadný způsob pro jejich zpracovávání.

Tato diplomová práce studuje zpracovávání obrazových dat pořízených digitálním fotoaparátem, mračen bodů získaných 3D skenerem a jejich vzájemnou interakci. V následujících podkapitolách tak bude uvedeno, jak jsou tato matematická data běžně reprezentována v počítačové paměti.

Text v této kapitole předpokládá základní znalost běžných matematických pojmů z oblastí matematické analýzy, lineární algebry a počítačové grafiky. Zpracování této kapitoly vychází ze zdrojů [18, 19, 20, 21, 22, 23, 24, 25].

2.1. Digitální obrazy

Digitálním obrazem rozumíme diskretizovaný dvojrozměrný signál, který nejčastěji popisuje intenzitu dopadeného světla na snímací zařízení. V praxi se obvykle jedná o čip digitálního fotoaparátu. Měřenou veličinu je možné popsat pomocí diskrétní funkce dvou proměnných i. Hodnoty této funkce $i(x, y)$ představují hodnotu měřené veličiny v daném bodě obrazu. Pro počítačovou reprezentaci této funkce používáme nejčastěji tzv. obrazovou matici. Prvky této matice se obvykle nazývají pixely (z anglického *picture elements* – obrazové elementy) a jsou to nejmenší dále nedělitelné jednotky této reprezentace daného obrazu.

Definice 2.1.1 (Digitální obraz v odstínech šedi)**.** Zobrazení $i : R \to W$, kde

$$R = \{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, n-1\}, \quad m, n \in \mathbb{N} \tag{2.1}$$

a

$$W = \{0, 1, \ldots, w-1\}, \quad w \in \mathbb{N}, \tag{2.2}$$

se nazývá (digitální) obraz v odstínech šedi. Čísla m, n jsou po řadě šířka obrazu a výška obrazu. Číslo w určuje počet stupňů šedi. Jeho hodnota je obvykle 2^n, kde $n \in \mathbb{N}$ je bitová hloubka obrazu [23].

Poznámka. Jak již bylo zmíněno, do počítačové paměti se funkce i nejčastěji zapisuje jako matice typu $m \times n$, jejíž prvky tvoří funkční hodnoty $i(x, y)$. Tato reprezentace je však pouze názorná a při počítání s obrazy je třeba k nim přistupovat jako k funkcím. Například násobení dvou obrazů je tak na rozdíl od maticového násobení definováno po prvcích (*element-wise*).

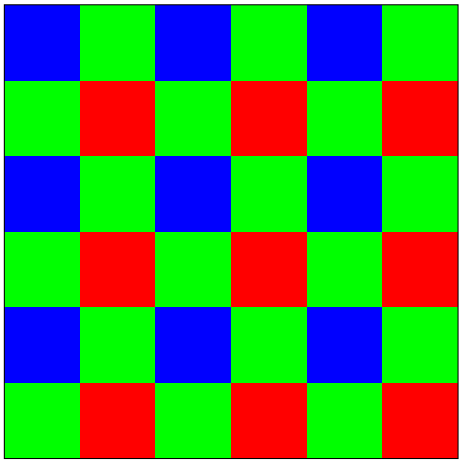
Bitová hloubka obrazu určuje, kolik bitů počítačové paměti je pro každý pixel vyhrazeno. Velmi často je bitová hloubka rovna 8. Takový obraz má pak 256 dosažitelných stupňů šedi. Tato hodnota přímo určuje dynamický rozsah obrazu, což je poměr mezi nejvyšší a nejnižší hodnotou měřené fyzikální veličiny, která se v něm může objevit. Ostatní hodnoty se při měření ztrácejí.

Ačkoliv je zde obraz definován jako funkce, jejíž hodnoty leží v oboru celých čísel, v praxi může být výhodné obraz transformovat vydělením číslem $(w-1)$. Získáme tak funkci, jejímž oborem hodnot je vždy podmnožina intervalu $\langle 0, 1\rangle$, což umožňuje obrazy porovnávat nehledě na různost jejich bitových hloubek.

Definice 2.1.2 (Barevný digitální obraz)**.** Trojici digitálních obrazů ve stupních šedi o stejných šířkách a výškách (r, g, b) nazýváme barevný (digitální) obraz. Jednotlivé obrazy r, g a b se nazývají (barevné) kanály, konkrétně po řadě červený, zelený a modrý kanál. Šířkou a výškou barevného obrazu rozumíme šířku a výšku jeho kanálů.

Poznámka. Ačkoliv se teoreticky počet stupňů šedi jednotlivých kanálů barevného digitálního obrazu nemusí rovnat, ve většině případů je tato hodnota pro všechny kanály stejná. Tato práce dále předpokládá pouze barevné obrazy, ve kterých se hodnota w ve všech kanálech shoduje.

Dnešní digitální fotoaparáty pořizují nejčastěji barevné digitální obrazy. K tomu používají čipy překryté Bayerovou maskou a fotodiody citlivé na určité vlnové délky spektra.



Obrázek 2.1: Příklad Bayerovy masky.

Pro některé algoritmy je však výhodnější zanedbat význam barevných kanálů a pracovat pouze s jedním obrazem ve stupních šedi. Převedení barevného obrazu na obraz ve stupních šedi se provádí pomocí váženého průměrování hodnot jednotlivých kanálů. Výsledný obraz ve stupních šedi má stejnou šířku a výšku jako původní barevný digitální obraz a získá se vztahem

$$i \doteq c_r r + c_g g + c_b b, \tag{2.3}$$

kde

$$c_r + c_g + c_b = 1, \quad c_r, c_b, c_g \in \mathbb{R}^+. \tag{2.4}$$

Volba konstant c_r, c_g a c_b není univerzální pro všechny obrazy. Ačkoliv lze použít $c_r = c_g = c_b = \frac{1}{3}$, tato volba nerespektuje rozdílnou citlivost senzoru na různé vlnové délky světla. Běžné čipy digitálních fotoaparátů jsou totiž obvykle mnohem citlivější na zelenou barvu než na barvy ostatní. To je zapříčiněno uspořádáním Bayerovy masky (viz obrá-

zek 2.1) a různou kvantovou účinností fotodiod. Mezi možné volby konstant tak patří například

$$c_r = 0,3, \quad c_g = 0,59, \quad c_b = 0,11; \tag{2.5}$$

$$c_r = 0,2126, \quad c_g = 0,7152, \quad c_b = 0,0722; \quad \text{nebo} \tag{2.6}$$

$$c_r = 0,299, \quad c_g = 0,587, \quad c_b = 0,114. \tag{2.7}$$

2.1.1. Dvojrozměrná diskrétní konvoluce

Důležitým nástrojem teorie zpracování signálů je konvoluce. Konvoluce je matematický operátor, který zpracovává dvě funkce (zkoumanou funkci a konvoluční jádro) a jehož výstupem je nějaká jiná funkce (odezva).

Digitální obrazy jsou dvojrozměrné diskretizované signály a pro práci s nimi je nutné použít dvojrozměrnou diskrétní konvoluci.

Definice 2.1.3 (Dvojrozměrná diskrétní konvoluce)**.** Nechť i_1 a i_2 jsou digitální obrazy. Definiční obor těchto obrazů se na celou množinu $\mathbb{Z}^2$ rozšíří následujícím způsobem

$$i_1^*(x,y) = \begin{cases} i_1(x,y) & \text{je definováno,} \\ 0 & \text{jinak,} \end{cases} \tag{2.8}$$

$$i_2^*(x,y) = \begin{cases} i_2(x,y) & \text{je definováno,} \\ 0 & \text{jinak.} \end{cases} \tag{2.9}$$

Dvojrozměrná diskrétní konvoluce obrazů i_1 a i_2 je pak dána vztahem

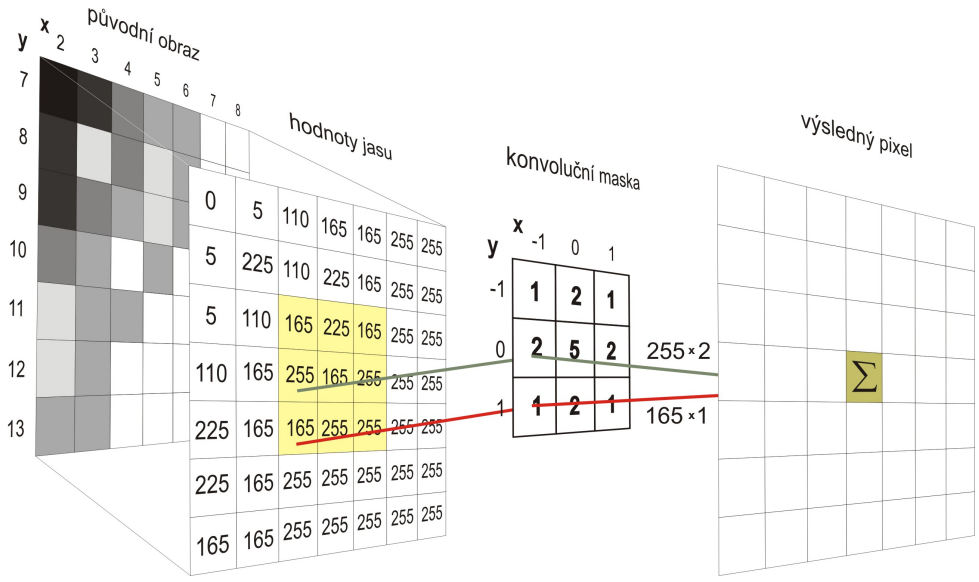
$$(i_1 * i_2)(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} i_1^*(k,l) \cdot i_2^*(x-k, y-l). \tag{2.10}$$

Poznámka. Výstupem dvojrozměrné diskrétní konvoluce dvou obrazů je obraz definovaný na celé množině $\mathbb{Z}^2$. V praxi se však jako výstup očekává obraz definovaný pouze na nějakém zúženém definičním oboru (obvykle stejném jako měl jeden z původních obrazů). Počítačový výpočet lze zjednodušit tím, že počítáme pouze hodnoty pixelů v bodech, které potřebujeme.

Dodejme, že existují i jiná rozšíření, než byla uvedena ve vztazích (2.8) a (2.9). Ta často využívají například zrcadlení krajních bodů obrazu.

V případě konvoluce dvou obrazů se konvolučnímu jádru obvykle říká konvoluční maska nebo filtr a je možné si jej představit jako tabulku, která se postupně přikládá k jednotlivým pixelům druhého obrazu. Hodnoty překrytých pixelů se vynásobí a sečtou, čímž vygenerují hodnotu pixelu pro nový obraz.

Pomocí konvoluce je možné snadněji vyjádřit některé složité operace, které provádíme s obrazy. Konvoluci lze použít například pro detekci hran, zaostřování a rozmazávání obrazu, ale ve svém principu je i základem konvolučních neuronových sítí, které dnes dominují na poli zpracování obrazů [27, 28, 29].



Obrázek 2.2: Princip diskrétní konvoluce [26].

Konvoluce zde bude použita k aproximaci diskrétních parciálních derivací obrazu. Jako konvoluční masky zvolíme například Sobelovy operátory

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \tag{2.11}$$

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}. \tag{2.12}$$

Pak pro obraz i platí

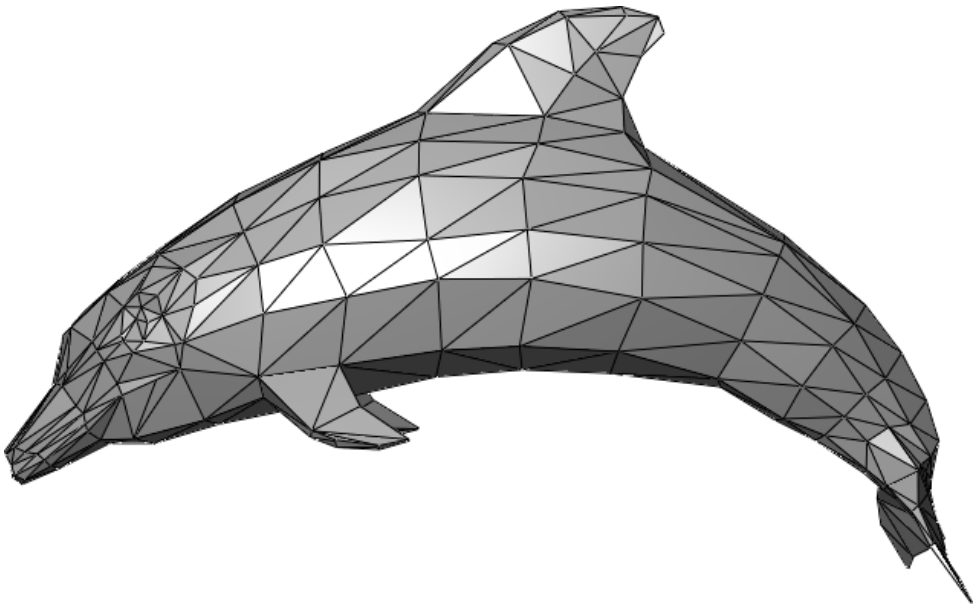
$$\frac{\partial i(x,y)}{\partial x} \approx (i * G_x)(x,y), \tag{2.13}$$

$$\frac{\partial i(x,y)}{\partial y} \approx (i * G_y)(x,y). \tag{2.14}$$

Poznamenejme, že Sobelovy operátory nejsou jedinou možnou volbou pro aproximaci parciálních derivací. Mezi jiné možnosti patří například Prewittové operátor nebo Robinsonův operátor. Také dodejme, že ačkoliv Sobelovy operátory uvádíme v rovnicích (2.11, 2.12) jako matice, ve skutečnosti se jedná o obrazy zapsané pomocí obrazové matice, proto je v tomto textu neznačíme tučně [19, 20, 21].

2.2. Mračna bodů

Dalšími zkoumanými objekty této práce jsou mračna bodů. Mračnem bodů se rozumí neuspořádaná množina bodů umístěných v prostoru. Tyto body jsou určeny třemi kartézskými souřadnicemi $[x, y, z]$. Mračno bodů svým rozprostřením obvykle reprezentuje reálný objekt. Typicky je mračno bodů výstupem 3D skenovacího zařízení (skeneru) nebo fotogrammetrického softwaru. Tyto zařízení a aplikace umožňují získat mnoho bodů na povrchu zkoumaných objektů.



Obrázek 2.3: Příklad trojúhelníkové sítě reprezentující delfína [33].

Ačkoliv lze mračna v této podobě zobrazovat i zkoumat, v některých případech je výhodné znát celý povrch objektů zájmu. Proces, který mračno bodů převádí na trojrozměrnou plochu nazýváme rekonstrukce povrchu. Výsledný povrch můžeme reprezentovat více způsoby. Tato práce se bude v některých částech zabývat mračny, které byly rekonstrukcí doplněny na polygonovou síť. Tu lze získat například pomocí Delaunayho triangulace [30], ale i díky *alpha-shapes* rekonstrukce [31] nebo *ball-pivotingu* [32]. Polygonové sítě často reprezentujeme pomocí neorientovaných prostorových grafů, což nám umožňuje přenést na ně řadu užitečných vlastnostní z teorie grafu.

Polygonová síť je zjednodušená reprezentace povrchu, ve které hrany mezi vrcholy (body mračna) tvoří strany polygonů. Tyto polygony tvoří zjednodušený povrch reprezentované plochy. Jedná se tedy vlastně o po částech lineární aproximaci skutečného povrchu tělesa, které síť reprezentuje. Jsou-li navíc všechny polygony v polygonové sítě trojúhelníky, hovoříme o trojúhelníkové síti. Příklad této sítě je na obrázku 2.3.

V praxi se takřka vždy setkáváme s trojúhelníkovými sítěmi. Důvodem je, že hardware dnešních počítačů je pro tento typ dat optimalizován. Každou polygonovou síť lze navíc na trojúhelníkovou síť převést pomocí teselace jejích stěn. Teselace (z anglického *tesellation*, mozaikování) je vyplnění roviny pomocí jednoho nebo více geometrických útvarů bez mezer a překryvů.

Dále uvedeme definici neorientovaného grafu a vysvětlíme jeho souvislost s polygonovou sítí.

Definice 2.2.1 (Jednoduchý neorientovaný graf)**.** Nechť V je množina bodů v prostoru (vrcholů) a

$$E = \{\{u, v\} \mid u, v \in V, u \neq v\} \tag{2.15}$$

je konečná množina hran, pak dvojice $G = (V, E)$ se nazývá jednoduchý neorientovaný graf nebo pouze graf [24].

Jak již bylo zmíněno, neorientovaný graf lze použít pro matematickou reprezentaci polygonové sítě. Množina bodů V je skutečnou množinou prostorových bodů původního

mračna. Tyto body v kontextu polygonové sítě obvykle nazýváme vrcholy. Množina hran E reprezentuje všechny spojnice dvou vrcholů, které v dané polygonové síti existují. Tento pohled nám umožňuje studovat polygonové sítě z pohledu teorie grafu a využívat pro jejich zpracovávání algoritmy, které se v této oblasti matematiky používají. Zároveň tak můžeme na polygonové sítě přenést pojmy hovořící o grafech (sousední vrcholy, cesta, vzdálenost vrcholů...).

Na těchto pojmech je možné vystavět další složitější struktury, které mohou popisovat i jiné vlastnosti digitálních modelů jako je například textura povrchu, materiál atd. Ačkoliv tyto složitější struktury mohou mít široké využití, pro tento text zcela dostačuje zpracovávání polygonových sítí, na které se budeme dívat z pohledu teorie grafu. Jako jedinou výjimku budeme každému vrcholu přiřazovat trojici celých čísel (r, g, b) takových, že $r, g, b \in \{0, 1, \ldots, w-1\}$, kde číslo w má stejný význam jako v definici 2.1.1. Této trojici budeme říkat barva daného vrcholu.

Poznámka. Poznamenejme, že v kontextu grafů se o barvě vrcholů obvykle hovoří ve smyslu barvení grafu [34]. Tam jsou barvy nejčastěji reprezentovány přirozenými čísly a je na ně dáván požadavek rozdílnosti pro sousední vrcholy. Význam barvy v této práci je přímo spojen s tím, jak je běžné chápána.

Zopakujme nyní ještě několik základních pojmů týkajících se grafů, které budeme používat v této práci. Definice těchto pojmů pocházejí z [24].

Definice 2.2.2 (Procházka v grafu)**.** Nechť $G = (V, E)$ je graf, pak procházkou mezi vrcholy $u_0, u_n \in V$ v grafu G rozumíme střídající se sekvenci vrcholů a hran

$$(u_0, e_1, u_1, e_2, \ldots, u_{n-1}, e_n, u_n), \tag{2.16}$$

ve které $n > 0$ je délka této procházky a $e_i = (u_{i-1}, u_i) \in E$ pro všechna $i \in \{1, \ldots, n\}$.

Definice 2.2.3 (Cesta v grafu)**.** Procházku v grafu mezi vrcholy $u_0, u_n \in V$ budeme nazývat cesta v grafu G, pokud pro různá i a j z $\{0, \ldots, n-1\}$ platí, že u_i a u_j jsou různé vrcholy.

Množinu všech cest mezi vrcholy u a v budeme značit $\mathcal{P}(u, v)$.

Poznámka. Cesta je procházka, která má všechny hrany i vrcholy různé.

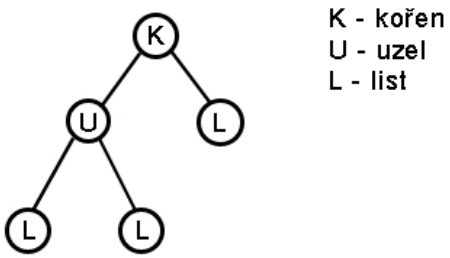
Definice 2.2.4 (Souvislý graf)**.** Graf nazýváme souvislý, jestliže v něm existuje cesta mezi každými dvěma vrcholy.

Definice 2.2.5 (Smyčka v grafu)**.** Smyčka v grafu G je cesta, která začíná a končí ve stejném vrcholu.

Definice 2.2.6 (Strom)**.** Souvislý graf se nazývá strom, jestliže neobsahuje žádné smyčky.

Poznámka. Bez definice poznamenejme, že ve stromech nalézáme tři typy vrcholů – kořen, uzly a listy. List je vrchol, který sousedí s pouze jediným dalším vrcholem. Tento vrchol je jeho rodič. Uzel je vrchol, jehož potomky jsou další uzly případně listy. Kořen je jediným vrcholem stromu, který nemá žádné rodiče, ale pouze potomky. Vrcholy stromu ilustruje pro lepší pochopení obrázek 2.4.

Definice 2.2.7 (Ohodnocený graf)**.** Je-li $G = (V, E)$ graf, pak zobrazení $h : E \to \mathbb{R}$ se nazývá ohodnocení grafu G. Graf G spolu s ohodnocením h se nazývá ohodnocený graf. Reálnému číslu $h(e)$ přiřazenému k hraně e říkáme hodnota (nebo též délka, cena) hrany.



Obrázek 2.4: Příklad stromu [35].

Poznámka. Dodejme, že pro polygonové sítě nemá záporná hodnota hrany příliš velký smysl. Obvykle hodnotu volíme buď rovnu skutečné euklidovské vzdálenosti vrcholů nebo rovnu 1.

Definice 2.2.8 (Délka cesty)**.** Délka cesty $p = (e_1, e_2, \ldots, e_n)$ v ohodnoceném grafu je definována přirozeně jako

$$l(p) = \sum_{i=1}^{n} h(e_i).$$

Definice 2.2.9 (Vzdálenost vrcholů v grafu)**.** V ohodnoceném grafu G je vzdálenost vrcholů u a v definována jako

$$d(u,v) = \begin{cases} \min\limits_{p \in \mathcal{P}(u,v)} \{l(p)\} & \text{pro} \quad \mathcal{P}(u,v) \neq \emptyset, \\ \infty & \text{jinak.} \end{cases}$$

Poznámka. Vzhledem k tomu, že polygonové sítě budeme vždy chápat jako souvislé grafy, tak existuje cesta mezi všemi vrcholy. Nikdy se tedy nemůže stát, že by vzdálenost dvou vrcholů v polygonové síti byla ∞.

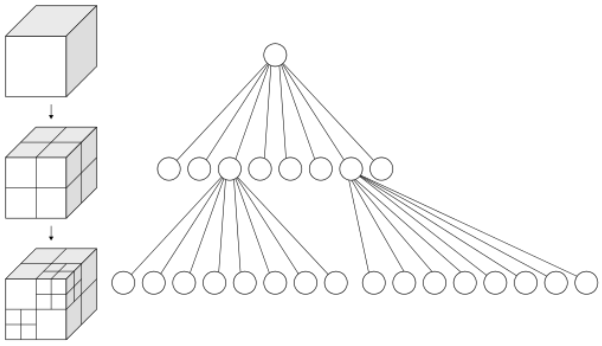
Zdůrazněme ještě, že vzdálenost vrcholů v grafu může být jiná než skutečná vzdálenost bodů v prostoru. Je to z toho důvodu, že vzdálenost vrcholů v grafu se měří po hranách tohoto grafu. Mezi těmito pojmy je nutné důrazně rozlišovat.

Pojem vzdálenosti nám umožňuje snadno zavést důležitý pojem okolí vrcholu.

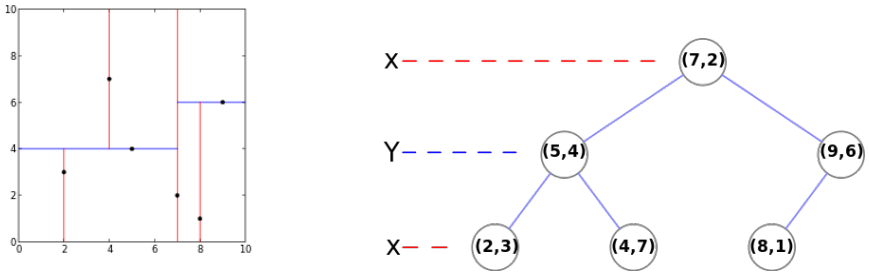
Definice 2.2.10 (Okolí vrcholu)**.** Máme-li vrchol u ohodnoceného grafu $G = (V, E)$ a číslo $k \in \mathbb{R}$, pak okolím vrcholu u o šířce k rozumíme množinu

$$O_k(u) = \{v \in V | d(u,v) \leq k\}.$$

V tomto textu bude dále potřeba rozlišovat u vzdáleností a okolí mezi klasickým ohodnocením euklidovskými vzdálenostmi a ohodnocením $h(e) = 1$ pro všechny hrany příslušného grafu. V druhém případě bude před pojmy používán přívlastek hranový (hranová vzdálenost, hranové okolí). V případě hranového okolí je logické šířku k udávat přirozeným číslem.



Obrázek 2.5: Oktálový strom [39].



(a) Vstupní data [43]. (b) Vzniklý strom [44].

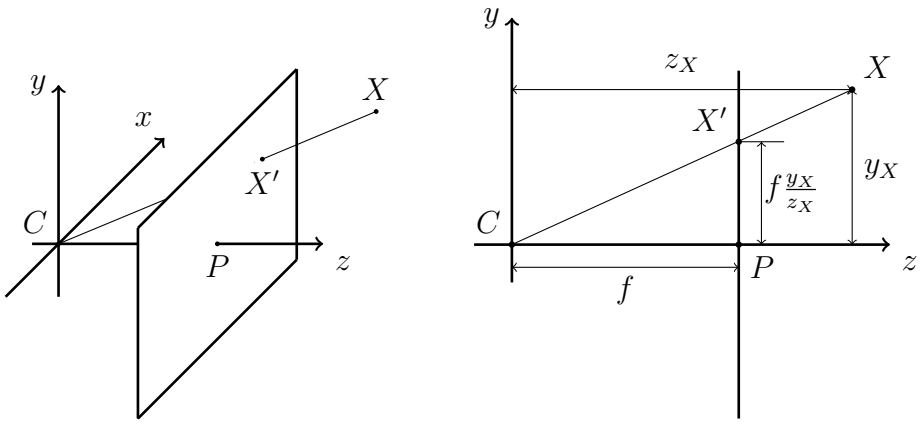
Obrázek 2.6: Příklad k-d stromu.

2.2.1. Počítačové struktury pro práci s mračny bodů

Přímá implementace matematické reprezentace mračen a polygonových sítí nemusí být pro použití v počítači příliš výhodná. Grafy bývají v počítači nejčastěji implementovány pomocí vrcholů s ukazateli nebo pomocí matice sousednosti. Ani jeden z těchto způsobů však nedokáže přímo postihnout trojrozměrnou geometrii našich dat. Pro optimalizaci se tak využívají oktálové nebo k-d stromy. Tato část čerpá z [36, 37, 38]

Oktálový strom Oktálový strom (*octree*) je strom, ve kterém každý uzel má přesně 8 potomků. Tyto stromy se využívají pro rekurzivní rozdělení trojrozměrného prostoru na jednotlivé oktanty až do nějaké předem stanovené úrovně. Body mračna pak přiřadíme vhodnému listu stromu podle toho, kde v prostoru se nacházejí. Tento způsob reprezentace dává obrovskou výhodu při hledání bodů v nějaké předem známé části prostoru, neboť značně omezuje množství bodů, které musíme projít. Princip oktálových stromů je nejlépe patrný z obrázku 2.5.

k-d strom Podobně jako *octree* i k-d strom je stromová struktura. Zatímco však oktálový strom rozděluje prostor rekurzivně na oktanty, k-d strom je binární (tj. jeho uzly mají vždy dva potomky) a rozděluje prostor na dva poloprostory postupně v jednotlivých dimenzích. Rozdělení prostoru k-d stromem je nepravidelné a své uplatnění nachází zejména ve vyšších dimenzích. Některé metody jej však uplatňují i pro práci s trojrozměrnými daty. Efektivní konstrukce a aplikace těchto stromů je aktivně řešeným problémem [40, 41, 42]. k-d strom je zobrazen na obrázku 2.6.



(a) Schéma dírkové komory. (b) Podobnost trojúhelníků.

Obrázek 3.1: Dírková komora.

3. Kamera

V kapitole 2.1 už byl naznačen vznik digitálních obrazů a byla popsána jejich reprezentace pomocí obrazové matice. Tato kapitola se věnuje matematickému modelování procesu pořízení obrazu ve skutečné kameře, k čemuž využívá zejména znalosti z projektivní geometrie uvedené v kapitole 1. Na konci kapitoly bude dále uveden princip kalibrace kamery a vysvětleno jeho použití pro získání parametrů této kamery. Text této kapitoly vychází z [14, 45].

Pořizování obrazu pomocí kamery je proces, při kterém zobrazujeme body trojrozměrného prostoru do dvojrozměrné roviny. O prostoru před kamerou hovoříme jako o scéně a rovinu, do které jednotlivé body scény zobrazujeme často nazýváme obrazovou rovinou.

Toto zobrazení můžeme v projektivní geometrii chápat jako lineární zobrazení, které lze popsat maticí. Tato matice se nejčastěji nazývá matice kamery.

3.1. Dírková komora

Nejjednodušší kamerou je dírková komora (někdy též latinsky *Camera Obscura*). Jedná se o jednoduché optické zařízení skládající se z temné schránky (může se jednat i o temnou místnost) s malým otvorem v jedné ze stěn. Světlo odražené od povrchu předmětů ve vnější scéně prochází otvorem v temné schránce a při dopadu na protější stěnu vytváří převrácený obraz vnější scény.

Tento jev znali už Číňané v 5. století př. n. l., ale psal o něm i Aristotelés a v 16. století n. l. jej běžně využívali renesanční umělci jako malířskou pomůcku. V té době však tyto přístroje promítané obrazy neuměly ustálit.

První černobílou fotografii pořídil až v roce 1826 francouzský vynálezce Joseph Nicéphore Niépce, který pro uchování obrazu použil vyleštěnou cínovou desku pokrytou roztokem asfaltu. Použití této technologie bylo velmi zdlouhavé a ukázalo se být v historii fotografie slepou uličkou. Doba expozice této jediné fotografie byla zhruba osm hodin. Od té doby Niépce, Louis Daguerre, William Fox Talbot a další vynálezci zdokonalovali snímání fotografií technologiemi založenými hlavně na sloučeninách stříbra, které na světle tmavnou [45].

V matematickém modelu dírkové komory budeme obrazovou rovinu umisťovat před optický střed (střed promítání, otvor ve stěně komory), takže výsledný obraz nebude převrácený. Nechť tedy středem promítání je počátek souřadného systému eukleidovského prostoru $\mathbb{E}^3$. V literatuře se tento bod značí C. Obrazovou rovinou (někdy též ohnisková rovina) buď rovina $z = f \neq 0$, která je rovnoběžná s rovinou určenou souřadnými osami x a y. Číslo f je ohnisková vzdálenost kamery (*focal length*). Souřadná osa z je zároveň optickou osou této kamery a s obrazovou rovinou této kamery se protíná v principiálním bodě P [14]. Schéma tohoto optického systému ilustruje obrázek 3.1a.

Libovolný bod scény $X = [x_X, y_X, z_X]$ zobrazíme na obrazový bod X' ležící v rovině obrazu tak, že bod X spojíme s bodem C přímkou, bod X' je průsečíkem této přímky s obrazovou rovinou. Z podobnosti trojúhelníků, jak je znázorněno na obrázku 3.1b, snadno odvodíme, že homogenní souřadnice bodu X' jsou

$$\mathbf{X}' = (x', y', f)^\intercal = (f\frac{x_X}{z_X}, f\frac{y_X}{z_X}, f)^\intercal, \tag{3.1}$$

což odpovídá bodu $[f\frac{x_X}{z_X}, f\frac{y_X}{z_X}]$ v novém souřadném systému zavedeném v obrazové rovině [14].

Použitím homogenních souřadnic je navíc možné toto zobrazení jednoduše popsat maticovým zápisem

$$\mathbf{X}' = \begin{pmatrix} fx_X \\ fy_X \\ z_X \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_X \\ y_X \\ z_X \\ 1 \end{pmatrix}. \tag{3.2}$$

Matici typu 3×4 z předchozího vztahu nazýváme matice kamery a označíme $\mathbf{P}$. Tedy platí vztah

$$\mathbf{X}' = \mathbf{P}\mathbf{X}. \tag{3.3}$$

Navíc můžeme tuto matici zapsat ve tvaru

$$\mathbf{P} = \mathbf{K}\begin{pmatrix} \mathbf{I_{3\times 3}} & \mathbf{0_{3\times 1}} \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.4}$$

kde matici $\mathbf{K}$ říkáme matice vnitřních parametrů (obsahuje parametry dané vnitřní stavbou kamery) nebo také kalibrační matice kamery, $\mathbf{I_{3\times 3}}$ je jednotková matice typu 3×3 a $\mathbf{0_{3\times 1}}$ je matice nul typu 3×1. Význam matice $\begin{pmatrix} \mathbf{I_{3\times 3}} & \mathbf{0_{3\times 1}} \end{pmatrix}$ bude vysvětlen později [14].

3.2. Digitální kamera

Rozšiřme nyní původní jednoduchý model dírkové komory na model, který se bude více blížit skutečné digitální kameře. Dřívější digitální fotoaparáty využívaly pro snímání fotorezistory, které mění svůj odpor v závislosti na množství dopadeného světla. Dnes jsou nejrozšířenější čipy typů CMOS (*Complementary Metal Oxide Semiconductor*) a CCD (*Charge Coupled Device*). Ty využívají technologii založenou na fotodiodách.

Čip digitální kamery je rozdělený na menší světlocitlivé části, kterým říkáme pixely. Ty by měly mít čtvercový tvar, ale v praxi mohou mít tvar obdélníku, který navíc může být

i mírně zkosený. Původní model také předpokládal, že počátkem souřadnicového systému v rovině obrazu je principiální bod. V obrazech ale počátek umisťujeme do jednoho z rohů, a tedy i to je nutné zohlednit v kalibrační matici kamery.

Pokud všechny tyto informace zahrneme do kalibrační matice, získáme matici

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.5}$$

kde platí

$$\alpha_x = \frac{f}{h_x}, \tag{3.6}$$

$$\alpha_y = \frac{f}{h_y}, \tag{3.7}$$

$$s = \alpha_y \operatorname{tg} \alpha \tag{3.8}$$

a h_x, h_y jsou skutečné délky stran pixelu na snímači a α je úhel zkosení pixelu vůči ose y. Pro zjednodušení budeme vždy předpokládat, že $s = 0$. Je patrné, že všechny tyto parametry souvisí s vnitřní stavbou kamery, a proto jsou součástí matice vnitřních parametrů [14]. Tuto matici je možné pro dané nastavení konkrétní kamery jednorázově určit procesem, který se nazývá kalibrace. Ten bude vysvětlen na konci této kapitoly.

3.3. Poloha kamery v prostoru

Kromě parametrů specifických pro danou kameru má kamera i parametry, které určují její polohu v trojrozměrném prostoru. Těmto parametrům říkáme vnější parametry a příslušná matice těchto parametrů se nazývá matice vnějších parametrů.

V části 3.1 jsme předpokládali, že optický střed kamery je umístěn v počátku souřadného systému a její optická osa je totožná se souřadnou osou z. To bylo vyjádřeno maticí $\begin{pmatrix} \mathbf{I_{3\times 3}} & \mathbf{0_{3\times 1}} \end{pmatrix}$ ve vztahu (3.4). Dále budeme uvažovat, že se optický střed nachází v obecném bodě prostoru $C = [x_C, y_C, z_C]$ s polohovým vektorem $\mathbf{c}$ a natočení optické osy vůči ose z je dáno rotační maticí $\mathbf{R_{3\times 3}}$. Pak lze matici kamery vyjádřit jako

$$\mathbf{P} = \mathbf{K} \begin{pmatrix} \mathbf{R_{3\times 3}} & -\mathbf{R_{3\times 3}}\mathbf{c} \end{pmatrix}, \tag{3.9}$$

kde matici $\mathbf{R_{3\times 3}}$ říkáme rotace kamery a vektoru $\mathbf{c}$ translace kamery [14].

Pro některé výpočty může být výhodnější pracovat s relativní polohou dvou různých kamer. Je-li tedy poloha první kamery zadána rotační maticí $\mathbf{R_1}$ a vektorem translace $\mathbf{c_1}$ v nějakém pevném souřadném systému a je-li poloha druhé kamery v souřadném systému první kamery určena relativně maticí rotace $\mathbf{R'_2}$ a vektorem translace $\mathbf{c'_2}$, pak polohu druhé kamery v pevném souřadném systému dokážeme vyjádřit jako

$$\mathbf{R_2} = \mathbf{R'_2}\mathbf{R_1}, \tag{3.10}$$

$$\mathbf{c_2}^\mathsf{T} = \mathbf{c_1}^\mathsf{T} + \mathbf{c'_2}^\mathsf{T}\mathbf{R_1}. \tag{3.11}$$

Dosazením kalibrační matice digitální kamery ve tvaru (3.5) do rovnice (3.9) obdržíme obecný model perspektivní kamery. Ten lze popsat maticí

$$\mathbf{P} = \begin{pmatrix} \mathbf{K}\mathbf{R_{3\times 3}} & -\mathbf{K}\mathbf{R_{3\times 3}}\mathbf{c} \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}. \tag{3.12}$$

Ukažme, jak lze z této matice zpět získat matice $\mathbf{K}$, $\mathbf{R_{3\times 3}}$ a vektor $\mathbf{c}$.

Označme $\mathbf{p_4} = (p_{14}, p_{24}, p_{34})^\intercal$ čtvrtý sloupec matice $\mathbf{P}$ a matici $\mathbf{M} = \mathbf{KR_{3\times 3}}$. Vidíme, že platí

$$-\mathbf{KR_{3\times 3}c} = -\mathbf{Mc} = \mathbf{p_4}, \tag{3.13}$$

a tedy

$$\mathbf{c} = -\mathbf{M^{-1}p_4}. \tag{3.14}$$

Protože kalibrační matice $\mathbf{K}$ je horní trojúhelníková a rotační matice $\mathbf{R_{3\times 3}}$ je ortogonální, dostaneme je RQ rozkladem matice $\mathbf{M}$. Dle [14] lze RQ rozklad provést například způsobem uvedeným dále.

Nechť

$$\mathbf{Q_x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, \tag{3.15}$$

$$\mathbf{Q_y} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}, \tag{3.16}$$

$$\mathbf{Q_z} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.17}$$

jsou matice tzv. Givensových rotací (podle matematika Jamese Wallace Givense, který je poprvé využil v padesátých letech), což jsou rotace v rovině dané dvěma souřadnými osami. Tvar těchto matic má za následek, že při násobení libovolné matice typu 3×3 zprava maticí Givensovy rotace je výsledkem matice, v níž jeden sloupec zůstane nezměněn a dva zbývající sloupce jsou nahrazeny nějakou lineární kombinací původních dvou sloupců. Úhly α, β a γ přitom můžeme volit tak, že se libovolný prvek ve změněných sloupcích vynuluje.

Ukažme si to například pro vynulování prvku a_{21} v nějaké libovolné matici $\mathbf{A}$ typu 3×3. Po vynásobení této matice zprava maticí $\mathbf{Q_z}$ získáme na druhém řádku v prvním sloupci hodnotu $a_{21}\cos\gamma + a_{22}\sin\gamma$. Abychom zde dostali nulu, pak musíme vyřešit rovnici

$$a_{21}\cos\gamma + a_{22}\sin\gamma = 0. \tag{3.18}$$

Tato rovnice má řešení

$$\cos\gamma = \frac{-a_{22}}{\sqrt{a_{21}^2 + a_{22}^2}}, \tag{3.19}$$

$$\sin\gamma = \frac{a_{21}}{\sqrt{a_{21}^2 + a_{22}^2}}, \tag{3.20}$$

které zjevně splňuje i podmínku přípustnosti vyplývající z vlastností goniometrických funkcí $\sin^2\gamma + \cos^2\gamma = 1$.

Tohoto principu můžeme využít pro postupné vynulování prvků v zadané matici tak, aby vznikla matice, která je horní trojúhelníková.

Vynásobme nejdříve matici $\mathbf{M}$ zprava maticí $\mathbf{Q_x}$, kde zvolíme

$$\cos\alpha = \frac{-m_{33}}{\sqrt{m_{32}^2 + m_{33}^2}}, \tag{3.21}$$

$$\sin\alpha = \frac{m_{32}}{\sqrt{m_{32}^2 + m_{33}^2}}. \tag{3.22}$$

Výsledná matice $\mathbf{M_x} = \mathbf{MQ_x}$ má prvek $m_{x32} = 0$. Dále tuto matici vynásobme zprava maticí $\mathbf{Q_y}$ s prvky

$$\cos\alpha = \frac{m_{x33}}{\sqrt{m_{x31}^2 + m_{x33}^2}}, \tag{3.23}$$

$$\sin\alpha = \frac{m_{x31}}{\sqrt{m_{x31}^2 + m_{x33}^2}}. \tag{3.24}$$

Tato operace nezmění druhý sloupec původní matice. Matice $\mathbf{M_{xy}} = \mathbf{MQ_xQ_y}$ má tedy oba prvky $m_{xy32} = m_{xy31} = 0$. Nakonec vynulujme prvek m_{xy21} vynásobením maticí $\mathbf{Q_z}$ volbou

$$\cos\alpha = \frac{-m_{xy22}}{\sqrt{m_{xy21}^2 + m_{xy22}^2}}, \tag{3.25}$$

$$\sin\alpha = \frac{m_{xy21}}{\sqrt{m_{xy21}^2 + m_{xy22}^2}}. \tag{3.26}$$

Protože první dva sloupce jsou nahrazeny lineární kombinací sebe sama, prvky m_{xyz31} a m_{xyz32} v matici $\mathbf{M_{xyz}} = \mathbf{MQ_xQ_yQ_z}$ zůstávají nulové a k nim přibude nulový prvek m_{xyz21}. Takže matice $\mathbf{M_{xyz}}$ je horní trojúhelníková.

Označme nyní $\mathbf{R} = \mathbf{M_{xyz}}$. Protože matice $\mathbf{Q_x}$, $\mathbf{Q_y}$ a $\mathbf{Q_z}$ jsou ortogonální, tak platí, že $\mathbf{M} = \mathbf{RQ_z^\intercal Q_y^\intercal Q_x^\intercal} = \mathbf{RQ}$, kde $\mathbf{Q} = \mathbf{Q_z^\intercal Q_y^\intercal Q_x^\intercal}$. Tedy $\mathbf{R}$ je kalibrační matice $\mathbf{K}$ této perspektivní kamery a matice $\mathbf{Q}$ je ve skutečnosti maticí rotace $\mathbf{R_{3\times3}}$ této kamery v prostoru.

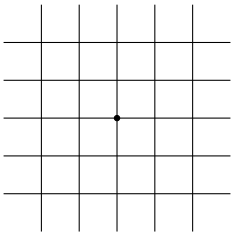
Poznamenejme, že existují i jiné posloupnosti Givensových rotací, které vedou ke stejnému výsledku a že obdobným postupem lze provést i podobné rozklady, které nazýváme QR, QL nebo LQ (kde výsledná matice $\mathbf{Q}$ je vždy ortogonální, matice $\mathbf{R}$ je horní trojúhelníková a matice $\mathbf{L}$ je dolní trojúhelníková). Tyto rozklady nacházejí široká uplatnění například v oblastech numerické matematiky.

3.4. Zkreslení

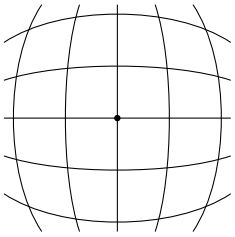
U některých kamer vzniká oproti skutečnosti významná změna obrazu vlivem nedokonalostí použité čočky. Tato změna nemá vliv na kvalitu získaného obrazu, ale ovlivňuje geometrii pořízené scény a zabraňuje tak použití běžného modelu perspektivní kamery ve fotogrammetrických aplikacích. Mezi nejpodstatnější se řadí radiální a tangenciální zkreslení (též distorze), jejichž popisu se věnuje tato kapitola. Později bude vysvětleno, jak je možné zmírnit negativní efekt způsobený zkreslením pomocí procesu kalibrace.

3.4.1. Radiální zkreslení

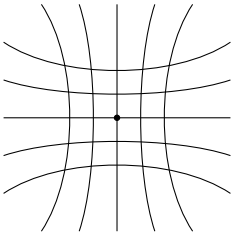
Radiální distorze je radiálně symetrická chyba v obrazu, která je tím výraznější, čím dál se na obrazu pohybujeme od jeho středu (resp. od optické osy). Tento druh zkreslení je



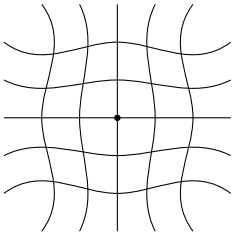
(a) Bez zkreslení.



(b) Zkreslení typu soudek.



(c) Zkreslení typu poduška.



(d) Kombinované zkreslení.

Obrázek 3.2: Druhy radiálního zkreslení – čáry znázorňují rovnoběžné a kolmé přímky scény, vyznačený bod je střed obrazu.

způsoben odlišným zakřivením čočky (a tedy odlišnou ohniskovou vzdáleností) na jejím okraji. Obvykle jej najdeme u transfokátorů, ale může se objevovat i u objektivů s pevnou ohniskovou vzdáleností.

Existují dvě hlavní kategorie radiálního zkreslení – zkreslení typu soudek (*barrel distortion*, obrázek 3.2b) a zkreslení typu poduška (*pincushion distortion*, obrázek 3.2c). V určitých případech může docházet i k jejich kombinaci (*mustache distortion*, někdy též *complex distortion*, obrázek 3.2d). Všechny druhy radiálního zkreslení se projevují tak, že přímky vypadají zakřivené, jak je patrné z obrázku 3.2.

Matematicky pozorujeme, že soudková a podušková distorze je kvadraticky závislá na vzdálenosti od optické osy a u složené distorze je dominantní kvartická závislost. Další sudé vyšší stupně jsou možné, ale méně obvyklé. Dále budeme uvažovat zkreslení nejvýše šestého stupně.

Označíme-li $[\tilde{x}, \tilde{y}]$ skutečné souřadnice nějakého bodu v obrazu, pak by se tento bod správně měl nacházet na ideálních souřadnicích $[x, y]$. Dále označme $[x_0, y_0]$ střed radiální distorze. Vzdálenost ideálních souřadnic bodu od tohoto středu lze vyjádřit jako $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. Vztah mezi skutečnými a ideálními souřadnicemi tak obecně modelujeme jako

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = D(r) \begin{pmatrix} x \\ y \end{pmatrix}, \tag{3.27}$$

kde funkci D nazýváme distorzní faktor. Tuto funkci budeme, jak již bylo zmíněno, aproximovat polynomem šestého stupně následujícím způsobem

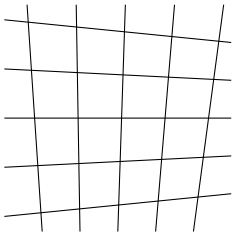
$$D(r) \approx 1 + k_1 r^2 + k_2 r^4 + k_3 r^6. \tag{3.28}$$

Koeficienty k_1, k_2 a k_3 jsou také považovány za neznámé vnitřní parametry kamery, jejichž určení je mimo jiné předmětem kalibrace kamery [46].

Známe-li distorzní faktor D, pak můžeme získaný obraz opravit přepočtem

$$\hat{x} = x_0 + D(\tilde{r})(\tilde{x} - x_0), \tag{3.29}$$

$$\hat{y} = y_0 + D(\tilde{r})(\tilde{y} - y_0), \tag{3.30}$$



Obrázek 3.3: Tangenciální zkreslení – čáry znázorňují rovnoběžné a kolmé přímky scény.

ve kterém $[\hat{x}, \hat{y}]$ jsou opravené souřadnice a $\tilde{r}$ je vzdálenost skutečných souřadnic bodu od středu radiální distorze daná vztahem $\tilde{r} = \sqrt{(\tilde{x} - x_0)^2 + (\tilde{y} - y_0)^2}$.

Dodejme, že v praxi se střed radiálního zkreslení může lišit od principiálního bodu obrazu a že ani symetrie tohoto zkreslení nemusí být dokonalá. To však bývá zanedbáváno.

3.4.2. Tangenciální zkreslení

Dalším druhem zkreslení způsobeným nedokonalostí konstrukce snímacího zařízení je zkreslení tangenciální. To je zapříčiněno tím, že čočka objektivu není umístěna dokonale rovnoběžně s obrazovou rovinou (snímacím čipem kamery). Efekt tohoto zkreslení je možno pozorovat na obrázku 3.3. Tangenciální distorze je obvykle méně významná a bývá mnohými autory zanedbávána, proto zde bude popsána jen stručně.

U tangenciálního zkreslení předpokládáme, že skutečné souřadnice $[\tilde{x}, \tilde{y}]$ vzniknou z ideálních souřadnic $[x, y]$ vztahy

$$\tilde{x} = x + [2p_1xy + p_2(r^2 + 2x^2)], \tag{3.31}$$

$$\tilde{y} = y + [p_1(r^2 + 2y^2) + 2p_2xy], \tag{3.32}$$

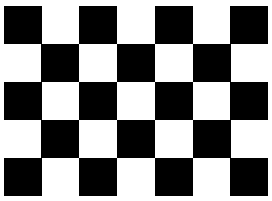
v nichž r opět označuje vzdálenost od středu obrazu $[x_0, y_0]$, kterou lze vyjádřit jako $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. Parametry p_1 a p_2 řadíme k vnitřním parametrům kamery, které stejně jako u parametrů radiální distorze můžeme určit kalibrací kamery [46].

3.5. Kalibrace kamery

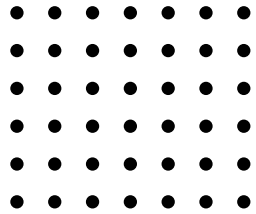
Vnitřní parametry kamery je možné určit pomocí kalibrace kamery. Jedná se o proces, při kterém neznámé parametry odhadujeme z několika fotografií známé scény, kterou dokážeme dobře popsat. Nejčastěji se pro tyto účely používá tzv. kalibrační vzor, kterým může být například šachovnice (*checkerboard*), mřížka s terčíky (*circle grid*) nebo ChArUco. Některé nejčastěji používané kalibrační vzory jsou zobrazeny na obrázku 3.4.

Dále bude popsán postup kalibrace kamery. Je klíčové, aby při tomto procesu a po něm zůstalo nastavení kamery zafixováno. Zejména je důležité, aby při kalibraci, ani po ní už nedocházelo ke změnám v ostření nebo přiblížení objektivu. Tyto změny mění vnitřní parametry kamery, a tím znehodnocují provedenou kalibraci. Rovněž předpokládáme, že poloha, ani rotace kamery se během kalibrace nemění. Tato část čerpá z [46].

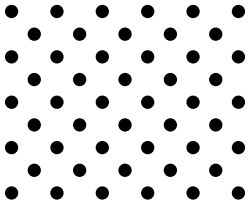
Vybraný kalibrační vzor se v přesně dané velikosti nanese na rovnou desku a následně se pořídí několik snímků scény, ve které se tato deska s viditelným kalibračním vzorem nachází. Pořizované snímky musejí kalibrační vzor zobrazovat z různých úhlů a vzdáleností (ale stále v oblasti, kde je obraz dostatečně ostrý). Zvláště pak pro dobrý odhad distorzních



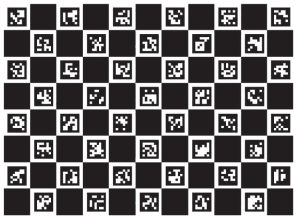
(a) Šachovnice.



(b) Mřížka s terčíky.

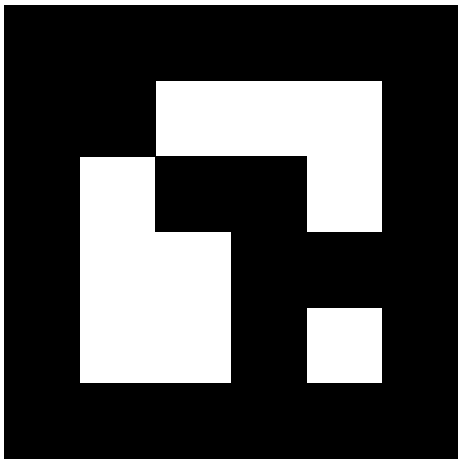


(c) Asymetrická mřížka s terčíky.



(d) ChArUco [47].

Obrázek 3.4: Příklady kalibračních vzorů.



Obrázek 3.5: Příklad značky ArUco.

koeficientů je nutné, aby se na snímcích kalibrační vzor nacházel v co nejvíce různých oblastech. Pro kvalitní odhad se doporučuje použít alespoň deset kalibračních snímků.

V pořízených obrazech se poté nalezne kalibrační vzor nějakým algoritmem, který je nejčastěji založený na metodách detekce významných bodů (viz kapitolu 4) nebo na metodách hledání přímek. Protože geometrii snímaného kalibračního vzoru dobře známe a můžeme předpokládat, že se celý vzor nachází v jedné rovině, dokážeme z těchto snímků odhadnout všechny vnitřní parametry kamery. Zároveň je možné odhadnout i relativní polohu kalibračního vzoru vůči poloze kamery, čehož lze využít v některých aplikacích, jak je kupříkladu předvedeno v ukázce kódu 3.2 na konci kapitoly 3.5.1.

Některé kalibrační metody používají kalibrační vzor nanesený na více na sebe kolmých desek. Výsledky tohoto postupu bývají někdy přesnější, ale konstrukce takového kalibračního zařízení je náročnější a náchylnější na nepřesnosti.

Výhodou použití kalibrační šachovnice je vysoká přesnost detekce. Některé algoritmy umějí naleznout mřížové body šachovnice i přesněji než na úroveň pixelů (*sub-pixel*). Naopak nevýhodou použití šachovnicového vzoru (obrázek 3.4a) pro kalibraci je v tom, že na každém snímku musí být celý tento vzor viditelný. To lze zásadně zlepšit použitím

kalibračního vzoru ChArUco (obrázek 3.4d), který má v bílých polích šachovnice umístěny značky ArUco (obrázek 3.5). Tyto značky lze v obraze snadno naleznout a protože jsou v šachovnici jedinečné, algoritmy jejich polohu dokáží využít pro doplnění mřížových bodů v zakrytých částech šachovnice. Tento přístup tak umožňuje, aby šachovnice byla ve scéně i částečně zakrytá a aby byla zachována dostatečná přesnost jejího použití.

Dále bude popsáno použití knihovny OpenCV v jazyce Python pro kalibraci kamery. Text bude pojednávat o kalibraci pomocí šachovnicového vzoru, ale stejný postup lze s drobnými modifikacemi (zejména v terminologii) aplikovat na jakýkoliv jiný kalibrační vzor.

3.5.1. Kalibrace kamery pomocí knihovny OpenCV

Jakmile máme pořízeny potřebné kalibrační snímky, můžeme použít knihovnu OpenCV pro kalibraci kamery. Potřebnými vstupními daty tohoto procesu jsou reálné trojrozměrné souřadnice mřížových bodů šachovnice a k nim odpovídající dvojrozměrné obrazové souřadnice v daných snímcích.

Problematika stanovení skutečných souřadnic šachovnice lze vyřešit jednoduše. Ačkoli snímky byly pořízeny statickou kamerou a kalibrační vzor byl v prostoru různě rotován a přemisťován, můžeme naopak předpokládat, že poloha kalibračního vzoru byla fixní a kamera se pohybovala odpovídajícím způsobem. Tedy dále můžeme předpokládat, že šachovnice byla umístěna do roviny $z = 0$, a tak souřadnice jejích mřížových bodů jsou $[x, y, 0]$. Volbou x a y nyní určujeme měřítko získaných kalibračních parametrů. Například zvolíme-li pro mřížové body souřadnice $[0, 0, 0], [1, 0, 0], [2, 0, 0], \ldots$, získané vnitřní parametry kamery budou reflektovat prostor, v němž délka jednotkového vektoru je rovna délce strany jednoho políčka šachovnice. Tuto volbu použijeme v případě, že neznáme přesnou délku stran šachových políček. V opačném případě můžeme zvolit x a y tak, aby prostor kamery byl definován v požadovaných jednotkách (je-li například délka strany šachovnice 20mm, pak volíme $[0, 0, 0], [20, 0, 0], [40, 0, 0], \ldots$, výsledný prostor je poté škálován v milimetrech).

Pro nalezení dvojrozměrných obrazových bodů v kalibračních snímcích lze využít funkci knihovny OpenCV:

```
ret, corners = cv2.findChessboardCorners(img, (m, n), output, flags)
```

Vstupními argumenty této funkce jsou

`img` – obrazová matice vstupního obrazu ve stupních šedi;

`(m, n)` – dvojice čísel m a n popisující velikost vzoru, šachovnice o rozměrech 5×7 na obrázku 3.4a má velikost vzoru $(4, 6)$;

`output` – výstupní pole, do kterého chceme umístit nalezené mřížové body (obvykle volíme `None`, protože funkce toto pole i vrací;

`flags` – případné upřesňující vlajky (tento argument je ve většině případů možno vynechat).

Výstupem této funkce jsou

`ret` – pravdivostní hodnota popisující, zda byla v obraze šachovnice nalezena;

`corners` – seznam nalezených mřížových bodů.

Nalezené mřížové body lze dále zpřesnit použitím funkce `cv2.cornerSubPix()`, jejíž použití je popsáno v [48].

Posledním krokem kalibračního procesu je odhadnutí vnitřních parametrů kamery. To zajišťuje funkce:

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(realPoints, imgPoints,
    imgSize, mtx, dist)
```

Do této funkce vstupují mimo jiné i argumenty

`realPoints` – seznam seznamů skutečných souřadnic šachovnicových bodů (často se bude jednat o opakování stejných souřadnic, ale funkce dovoluje tyto pozice měnit, pokud bychom chtěli ke kalibraci použít jiný přístup);

`imgPoints` – seznam seznamů zjištěných mřížových bodů v jednotlivých obrazech (jednotlivé položky seznamu musejí korespondovat s položkami seznamu skutečných bodů);

`imgSize` – velikost nějakého obrazu pořízeného kamerou (je použita pro inicializaci kalibračního algoritmu);

`mtx` – vstupní (a případně i výstupní) kalibrační matice kamery, kterou můžeme specifikovat, pokud ji známe (obvykle volíme `None`);

`dist` – vstupní (a případně i výstupní) seznam distorzních koeficientů, které lze do funkce dodat, pokud je náhodou již známe (obvykle volíme `None`).

Tato funkce dovoluje řadu dalších nastavení, která lze specifikovat doplněním dalších vstupních parametrů. Tyto parametry pro tento text nejsou důležité a nebudou více rozebírány. Informace o jejich použití lze nalézt v [48]. Funkce na výstupu vrací

`ret` – pravdivostní hodnota popisují, zda kalibrační funkce proběhla úspěšně;

`mtx` – odhadnutá kalibrační matice kamery;

`dist` – nalezený vektor distorzních koeficientů ve tvaru $(k_1, k_2, p_1, p_2, k_3)$;

`rvecs` – seznam odhadnutých Rodriguesových rotačních vektorů [49] popisujících rotaci kamery vůči pevně umístěné šachovnici v prostoru na jednotlivých snímcích;

`tvecs` – seznam odhadnutých translačních vektorů kamery vzhledem k poloze pevně umístěné šachovnice pro jednotlivé snímky.

Popsané funkce hrají klíčovou roli v kalibraci pomocí knihovny OpenCV. Dále bude následovat ukázka vzorového kódu demonstrující celý kalibrační proces. V ukázce jsou kromě knihovny OpenCV použity i knihovny NumPy [50] a glob. Ukázka předpokládá, že se veškeré kalibrační snímky nacházejí ve složce `calib` a jsou ve formátu `jpg`. Pro kalibraci jsou v ukázce použity šachovnice s velikostí vzoru $(4, 6)$ stejně jako na obrázku 3.4a.

```
import numpy as np
import cv2
import glob

# pripraveni skutecnych souradnic sachovnicovych bodu
realPoints = np.zeros((4*6, 3), np.float32)
realPoints[:, :2] = np.mgrid[0:4, 0:6].T.reshape(-1, 2)

realPointsList = []
imgPointsList = []

calibImages = glob.glob("calib/*.jpg")

# ukoncovaci kriterium pro sub-pixelove zpresneni nalezenych bodu
crit = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

for imgFile in calibImages:
    img = cv2.imread(imgFile)
    imgGrayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # nalezeni sachovnicovych bodu
    ret, corners = cv2.findChessboardCorners(imgGrayscale, (4, 6), None)

    # pokud byla sachovnice nalezena, prida skutecne i obrazove body do
    prislusneho seznamu
    if ret:
        realPointsList.append(realPoints)

        cornersSubPixel = cv2.cornerSubPix(imgGrayscale, corners, (11, 11),
    (-1, -1), crit)
        imgPointsList.append(cornersSubPixel)

# provedeni kalibrace kamery
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(realPointsList,
    imgPointsList, imgGrayscale.shape[::-1], None, None)
```

Kód 3.1: Proces získání kalibrační matice a distorzních koeficientů

Na závěr kapitoly uveďme, jak je možné ověřit přesnost zjištěných vnitřních parametrů kamery. Jelikož znalost vnitřních parametrů spolu s odhadnutou vzájemnou polohou kamery a šachovnice postačuje pro sestavení úplného modelu perspektivní kamery, můžeme tyto informace použít pro projekci námi definovaných skutečných trojrozměrných šachovnicových bodů do jednotlivých obrazů. Polohu promítnutých bodů, pak můžeme porovnat s polohou mřížových bodů získanou ze snímků. Tento postup prezentuje ukázka 3.2.

```
meanError = 0
numberOfImages = len(realPointsList)

for i in range(numberOfImages):
    # promitne skutecne sachovnicove body pomoci zjistenych parametru kamery
    projectedPoints, _ = cv2.projectPoints(realPointsList[i], rvecs[i], tvecs[i], mtx, dist)

    # zmeri vzdalenost nalezenych bodu a promitnutych bodu v norme L2
    error = cv2.norm(imgPointsList[i], projectedPoints, cv2.NORM_L2) / len(projectedPoints)
    meanError += error

# ulozi do promenne prumernou velikost chyby
meanError = meanError / numberOfImages
```

Kód 3.2: Zjištění chyby odhadu vnitřních parametrů kamery

4. Významné body

Každým dnem přibývá dat, která je potřeba počítačově zpracovávat. Stále dostupnější technika, levnější datová úložiště a nástup modernějších technologií – to vše pomohlo dávno překonat problém získávání dat. Dnes největším problémem je veškerá data rychle a přesně zpracovávat.

Tento problém si žádá nalezení nových účinnějších algoritmů pro práci s daty. Jedním z oborů, ve kterém dochází k největší mobilizaci, je počítačové vidění. Tato oblast na pomezí matematiky a výpočetní techniky se snaží snímáním okolního prostředí získat co nejvíce informací.

Ještě stále dominantní dvojrozměrné vnímání okolního světa pomocí fotoaparátů a kamer je pozvolna nahrazováno nebo doplňováno populárním trojrozměrným skenováním. Větší dostupnost těchto skenovacích zařízení nám nebrání předpokládat, že už zanedlouho budou stroje běžně vnímat i ve třech dimenzích. Vývoj metod počítačového vidění má tak klíčový význam na poli moderní techniky.

Obrazy i v poměrně nízkém rozlišení obsahují miliony pixelů. Jedním ze základních přístupů pro zrychlení algoritmů počítačového vidění je omezení počtu bodů, se kterými je nutné v algoritmech pracovat. Je tedy nutné zvolit několik dostatečně reprezentativních pixelů a vyhnout se tak práci s body, které nám nepřinášejí žádnou dodatečnou informaci. Takové body se v literatuře označují jako body zájmu nebo významné body (*features*). Abychom významné body nemuseli vnímat pouze samostatně, ale také v kontextu jejich okolí, musíme je doplnit o deskriptory – vektory, které okolí těchto bodů zjednodušeně, ale také dostatečně popisují.

V počítačové grafice se používá celá řada algoritmů pro detekci významných bodů. Mezi nejznámější patří zejména algoritmy SIFT (*Scale-Invariant Feature Transform*) [51] a SURF (*Speeded-Up Robust Features*) [52], které jsou však patentované a jejich užití je tak podmíněno zakoupením licence. K volně dostupným se řadí například algoritmus FAST (*Features from Accelerated Segment Test*) [53], který je velmi rychlý a hodí se i pro využití v aplikacích běžících v reálném čase. Jeho nevýhodou je, že nalezené významné body nejsou invariantní vůči otočení a škálování. To se snaží změnit jeho nadstavba ORB (*Oriented FAST and Rotated BRIEF*) [54], která původní algoritmus doplňuje o řadu vylepšení, díky kterým si z velké části zachovává svou rychlost, ale ztrácí některé své negativní vlastnosti. K tomu využívá upravený deskriptor BRIEF (*Binary Robust Independent Elementary Features*) [55]. Existují rovněž přístupy, které práci s obrazovými daty zefektivňují metodami strojového učení. Není třeba pochybovat, že existují obdobné algoritmy, které hledají významné body v mračnech bodů.

Tato kapitola se zabývá fungováním algoritmu ORB a jeho využitím pro zjištění množiny tzv. odpovídajících si bodů mezi dvěma obrazy. Nejdříve bude odvozena Harrisova-Stephensova metoda výběru významných bodů v obrazu [56], která je jedním ze základních prvků pro posouzení kvality nalezených významných bodů v algoritmu ORB. Tento algoritmus následně rozšíříme pro hledání významných bodů v trojrozměrných datech. Dále bude vysvětlen princip deskriptorů, zejména pak deskriptoru BRIEF a jeho upravené varianty rBRIEF (*Rotated BRIEF*), která je invariantní vůči rotaci. Konec této kapitoly je věnován algoritmu ORB.

Text v této kapitole vychází z původních článků a ze zdrojů [19, 25, 58, 59, 60, 61, 62, 63, 64, 65].

4.1. Odvození Harrisova detektoru

Harrisův detektor rohů je algoritmus původně vyvinutý Chrisem Harrisem a Mikem Stephensenem v roce 1988. Jejich práce rozvíjí detektor Hanse P. Moravce z roku 1977 [57], jehož hlavním nedostatkem je podle samotného autora fakt, že není izotropní. Oba tyto detektory definují jako významný bod v obrazu tzv. roh. Jedná se o bod, ve kterém se protínají dvě hrany v obrazu, přičemž hranou zde rozumíme ostrou změnu v intenzitě obrazu. Roh můžeme také chápat jako bod, který se ve všech směrech dostatečně liší od svého okolí. Odvoďme nyní metodu pro rychlé nalezení takových bodů. Toto odvození vychází z [58].

Zvolme libovolně nějaký čtvercový výřez obrazu $\mathcal{W}$ se středem v bodě $S_{\mathcal{W}}$. Míru změny obrazu pro vektor posunu $(\Delta x, \Delta y)^\intercal$ lze popsat autokorelační funkcí

$$E(\Delta x, \Delta y) = \sum_{x,y \in \mathcal{W}} W(x,y)[i(x+\Delta x, y+\Delta y) - i(x,y)]^2, \tag{4.1}$$

kde i je digitální obraz a W je váhová funkce. Ta udává příslušnost bodu $[x, y]$ do $\mathcal{W}$.

Volba váhové funkce umožňuje měnit vlastnosti Harrisova detektoru. Nejjednodušeji lze pro tuto funkci zavést

$$W(x,y) = \begin{cases} 1 & (x,y) \in \mathcal{W}, \\ 0 & (x,y) \notin \mathcal{W}, \end{cases} \tag{4.2}$$

avšak zajímavější volbou je diskretizovaná Gaussova dvojrozměrná funkce, která poskytuje lepší výsledky při práci se zašumělým obrazem a je dnes pro tuto metodu doporučována.

Ze vztahu (4.1) vyplývá, že hodnota autokorelační funkce je malá, pokud jsou hodnoty obrazu v posunutých bodech podobné jako v bodech výchozích. Pro nalezení bodů, které se nejvíce odlišují od svého okolí, je tak žádoucí hledat maxima této funkce.

Člen $i(x+\Delta x, y+\Delta y)$ ve vztahu (4.1) lze rozvinout Taylorovým rozvojem prvního stupně

$$i(x+\Delta x, y+\Delta y) \approx i(x,y) + \frac{\partial i(x,y)}{\partial x}\Delta x + \frac{\partial i(x,y)}{\partial y}\Delta y. \tag{4.3}$$

Předpokládáme-li, že posuny Δx a Δy jsou dostatečně malé, lze rozvoj (4.3) považovat za dostatečně dobrou aproximaci rozvíjené funkce i.

Protože obraz je diskrétní funkce, je nutné chápat parciální derivace v rozvoji (4.3) jako diskrétní aproximace parciálních derivací. Pro zjednodušení zápisu si tyto parciální derivace navíc označíme

$$\frac{\partial i(x,y)}{\partial x} = i_x(x,y) = i_x, \tag{4.4}$$

$$\frac{\partial i(x,y)}{\partial y} = i_y(x,y) = i_y. \tag{4.5}$$

Po dosazení aproximace (4.3) do rovnice (4.1) a algebraických úpravách obdržíme aproximaci autokorelační funkce

$$\begin{aligned} E(\Delta x, \Delta y) \approx \sum_{x,y \in \mathcal{W}} W(x,y)[&(i_x(x,y)\Delta x)^2 + \\ &+ 2\Delta x \Delta y i_x(x,y) i_y(x,y) + (i_y(x,y)\Delta y)^2], \end{aligned} \tag{4.6}$$

kterou je možné přepsat do maticového tvaru

$$E(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \begin{pmatrix} \sum\limits_{x,y\in\mathcal{W}} W i_x^2 & \sum\limits_{x,y\in\mathcal{W}} W i_x i_y \\ \sum\limits_{x,y\in\mathcal{W}} W i_x i_y & \sum\limits_{x,y\in\mathcal{W}} W i_y^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}. \tag{4.7}$$

Matici s parciálním derivacemi ze předchozího vztahu budeme dále značit

$$\mathbf{H} = \begin{pmatrix} \sum\limits_{x,y\in\mathcal{W}} W i_x^2 & \sum\limits_{x,y\in\mathcal{W}} W i_x i_y \\ \sum\limits_{x,y\in\mathcal{W}} W i_x i_y & \sum\limits_{x,y\in\mathcal{W}} W i_y^2 \end{pmatrix}. \tag{4.8}$$

V díle Harrise a Stephense [56] je předvedena souvislost vlastních čísel matice $\mathbf{H}$ a skutečnosti, zda je střed $S_{\mathcal{W}}$ rohem v obrazu i. Nalezení vlastních čísel matice je však výpočetně zbytečně složitý problém, a tak s výhodou můžeme využít jejich známých vlastností

$$\det \mathbf{H} = \lambda_1 \lambda_2, \tag{4.9}$$

$$\operatorname{trace} \mathbf{H} = \lambda_1 + \lambda_2. \tag{4.10}$$

Čísla λ_1 a λ_2 jsou vlastní čísla čtvercové matice $\mathbf{H}$ řádu 2. Pro daný střed $S_{\mathcal{W}}$ pak definujeme odezvu detektoru jako

$$R(S_{\mathcal{W}}) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(\mathbf{H}) - k \cdot \operatorname{trace}(\mathbf{H})^2. \tag{4.11}$$

Parametr k se volí z doporučeného intervalu $\langle 0,04; 0,06\rangle$. Obdobně odezvu zjistíme i pro další body obrazu. Body zájmu jsou potom diskrétní lokální maxima funkce odezvy detektoru R. Tedy body zájmu $[x_H, y_H]$ splňují podmínku

$$R(x_H, y_H) > R(x, y), \quad \forall [x, y] \in O_d(x, y), \tag{4.12}$$

ve které $O_d(x, y)$ má význam nějakého diskrétního okolí bodu $[x, y]$ (tedy se jedná o množinu všech bodů obrazu, které mají od bodu $[x, y]$ vzdálenost menší než d; v praxi se obvykle jedná o vzdálenost $d = 3$ v manhattanské metrice).

Předchozí text uvádí Harrisův detektor pouze coby nástroj pro nalezení významných bodů. V dalších částech předvedeme, jak je možné použít odezvu Harrisova detektoru pro hodnocení kvality významného bodu i v jiných detektorech.

4.1.1. Filtrace významných bodů

Pro většinu aplikací je výhodné, když jsou významné body dobře rozmístěny po celém obrazu. Toho je možné docílit následným filtrováním nalezených bodů. Označme $P = \{P_1, \ldots, P_n\}$ množinu nalezených významných bodů. Tuto množinu uspořádáme podle jejich Harrisovy odezvy sestupně. Tedy dále platí, že

$$R(P_1) \geq R(P_2) \geq \cdots \geq R(P_n). \tag{4.13}$$

Vezměme ještě $Q = \emptyset$ množinu, kterou v průběhu filtrace naplníme významnými body s dobrým rozmístěním. Procházejme postupně množinu P a bod P_j přidejme do množiny Q, pokud splňuje podmínku

$$\min_{k\in\langle 1;\operatorname{card} Q\rangle} \|P_j - Q_k\|_2 > \rho. \tag{4.14}$$

Tedy bod P_j přidáme do množiny Q, pokud je to bod s nejvyšší hodnotou R, který je zároveň od všech bodů již přítomných v množině Q vzdálen více než ρ. Za číslo ρ obvykle volíme zlomek úhlopříčky celého obrazu. Hodnotou tohoto parametru lze ovlivňovat hustotu a nepřímo i počet významných bodů.

4.1.2. Algoritmus Harrisova detektoru pro obrazy

Nyní uvedeme algoritmus Harrisova detektoru tak, jak by se implementoval v nějakém programovacím jazyce. Pro zjednodušení zápisu některých výpočtů bude použita dvojrozměrná diskrétní konvoluce.

1. Načti vstupní data – obraz i, váhovou funkci W a parametr k.
2. Vypočítej aproximace parciálních derivací obrazu i

$$i_x = (i * G_x), \tag{4.15}$$
$$i_y = (i * G_y). \tag{4.16}$$

   G_x a G_y jsou Sobelovy operátory.
3. Vypočítej pomocné součiny

$$i_x^2 = i_x \cdot i_x, \tag{4.17}$$
$$i_y^2 = i_y \cdot i_y, \tag{4.18}$$
$$i_x i_y = i_x \cdot i_y. \tag{4.19}$$

4. Pro každý bod obrazu $[x, y]$ sestav matici $\mathbf{H}$

$$\mathbf{H}(x,y) = \begin{pmatrix} (i_x^2 * W)(x,y) & (i_x i_y * W)(x,y) \\ (i_x i_y * W)(x,y) & (i_y^2 * W)(x,y) \end{pmatrix}. \tag{4.20}$$

5. Pro každý bod obrazu $[x, y]$ vypočítej odezvu detektoru R

$$R(x,y) = \det\left(\mathbf{H}(x,y)\right) - k \cdot \operatorname{trace}\left(\mathbf{H}(x,y)\right)^2. \tag{4.21}$$

6. Vyber lokální extrémy funkce R.
7. *Volitelně:* Proveď filtraci významných bodů tak, aby měly dobré rozložení ve zkoumaném obrazu i (kapitola 4.1.1).
8. Vrať nalezené významné body jako výstup programu.

4.2. Rozšíření detektoru pro trojrozměrná data

Doteď jsme o detekci významných bodů hovořili pouze v kontextu dvojrozměrných obrazových dat. Jak již však bylo naznačeno v úvodu této kapitoly, zabývat se detekcí bodů zájmů má smysl i pro data trojrozměrná, jakými jsou například mračna bodů. V této části rozšíříme Harrisův detektor v duchu článku [65].

Největším problémem použití Harrisovy metody pro hledání významných bodů v mračnech je výpočet parciálních derivací. Zatímco v obrazech jsme parciální derivace poměrně snadno aproximovali pomocí Sobelových operátorů, u mračen toto není dost dobře možné, neboť body v mračnu nejsou rozmístěny v mřížce.

Pro řešení tohoto problému nejdříve část mračna v okolí posuzovaného bodu lokálně nahradíme aproximačním paraboloidem a ten pak parciálně zderivujeme. Pro aproximaci budeme s výhodou využívat reprezentaci mračna pomocí polygonové sítě, jak bylo popsáno v části 2.2.

Harrisův-Stephensův algoritmus aplikovaný na mračna bodů se tak skládá ze tří částí, které provedeme pro každý vrchol v dané polygonové sítě.

Nahrazení aproximačním paraboloidem V prvním kroku algoritmu nejdříve vybereme vhodně široké hranové okolí $O_k(v)$ vrcholu v. Připomeňme, že hranovým okolím $O_k(v)$ rozumíme množinu všech vrcholů u z mračna takových, že se z nich do vrcholu v dostaneme po hranách sítě v nejvýše k krocích. Šířku okolí k je možné v algoritmu volit adaptivním způsobem v závislosti na poloze okolních bodů a optimalizovat tak jeho fungování. V této práci je k voleno konstantně.

Dále nalezneme těžiště T bodů okolí a body posuneme tak, aby bod T odpovídal počátku souřadnicového systému. Tento krok zlepšuje stabilitu algoritmu.

Pomocí algoritmu PCA (*Principal Component Analysis*) [66], ve kterém zachováváme všechny tři hlavní komponenty, nalezneme nejlepší aproximační rovinu. Použití tohoto algoritmu pro hledání aproximační roviny může být poněkud netradiční, ale tento přístup má jistou výhodu. Kdybychom pro hledání aproximační roviny chtěli použít například metodu nejmenších čtverců [18], museli bychom předpokládat, že jsou body rozprostřeny tak, že jejich $z = f(x, y)$. To však nemusí být splněno. Body okolí navíc s využitím PCA transformujeme do nových souřadnic, čímž máme zajištěno, že jejich rozptyl v ose z je minimální.

Všechny body následně posuneme tak, aby vrchol v odpovídal počátku souřadnicového systému.

Nyní již metodou nejmenších čtverců [18] nalezneme aproximační paraboloid v následujícím výhodném tvaru

$$f(x, y) = \frac{p_1}{2}x^2 + p_2xy + \frac{p_3}{2}y^2 + p_4x + p_5y + p_6. \tag{4.22}$$

Derivace nalezeného paraboloidu Druhý krok algoritmu se věnuje spočítání parciálních derivací aproximačního paraboloidu. Ačkoli by bylo možné použít běžné vztahy

$$f_x(x, y) = \frac{\partial f(x, y)}{\partial x} = p_1x + p_2y + p_4, \tag{4.23}$$

$$f_y(x, y) = \frac{\partial f(x, y)}{\partial y} = p_2x + p_3y + p_5, \tag{4.24}$$

tyto derivace mohou být do značné míry ovlivněny šumem ve zdrojovém mračnu.

Místo toho můžeme, stejně jako v originálním detektoru, použít jako filtr Gaussovu funkci. Narazíme však na problém. Protože v původním výrazu jsou derivace diskrétní funkce (aproximace parciálních derivací obrazu) a nyní se jedná o spojité funkce. Tento

problém lze vyřešit integrací parciálních derivací s dvojrozměrnou Gaussovou funkcí podle vztahů

$$A = \frac{1}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} f_x(x,y)^2 dxdy, \tag{4.25}$$

$$B = \frac{1}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} f_y(x,y)^2 dxdy, \tag{4.26}$$

$$C = \frac{1}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} f_x(x,y) f_y(x,y) dxdy. \tag{4.27}$$

Tyto výrazy lze navíc užitím kalkulu výrazně zjednodušit do tvarů

$$A = p_4^2 + 2p_1^2 + 2p_2^2, \tag{4.28}$$

$$B = p_5^2 + 2p_2^2 + 2p_3^2, \tag{4.29}$$

$$C = p_4 p_5 + 2p_1 p_2 + 2p_2 p_3. \tag{4.30}$$

Výpočet odezvy detektoru Posledním krokem je vypočtení odezvy Harrisova detektoru.

Stejně jako v původní úloze i nyní sestavíme z vypočtených parciálních derivací matici

$$\mathbf{H} = \begin{pmatrix} A & C \\ C & B \end{pmatrix}. \tag{4.31}$$

Z této matice vypočítáme odezvu detektoru R podle stejného vztahu jako v dvojrozměrném případě

$$R = \det(\mathbf{H}) - k \cdot \operatorname{trace}(\mathbf{H})^2, \tag{4.32}$$

kde k je opět zvolená konstanta z doporučeného intervalu $\langle 0,04; 0,06 \rangle$.

Obdobně jako v dvojrozměrné úloze i nyní vybereme množinu bodů, která má lokálně největší hodnotu odezvy R a tu následně volitelně filtrujeme způsobem popsaným v 4.1.1. Konstantu ρ pro filtrování obvykle volíme jako zlomek úhlopříčky celé polygonové sítě (skutečné prostorové vzdálenosti dvou nejvzdálenějších bodů).

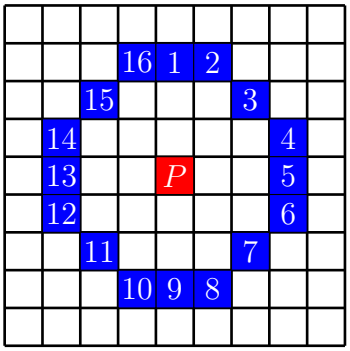
4.3. Detektor FAST

Algoritmus FAST [53], uveřejněný roku 2006 Edwardem Rostenem a Tomem Drummondem, používá pro posouzení významnosti obrazového bodu velmi jednoduchý porovnávací test. Uvažujme nějaký bod P v obrazu i. Tento bod budeme považovat za významný, jestliže existuje množina n souvislých pixelů (oblouk) P_j v kružnici k s poloměrem r kolem bodu P takových, že platí

$$(\forall j \in \{1,\ldots,n\} : i(P) + t < i(P_j)) \vee (\forall j \in \{1,\ldots,n\} : i(P) - t > i(P_j)). \tag{4.33}$$

Hodnota t je parametr nazývaný též práh (*threshold*), který umožňuje měnit vlastnosti testu.

Předně zdůrazněme, že kružnicí k rozumíme rastrovou kružnici vzniklou Bresenhamovým kružnicovým algoritmem [67], což je konečná množina, a tedy podmínku (4.33) lze



Obrázek 4.1: Schéma okolí zkoumaného bodu v algoritmu FAST – červený bod označuje zkoumaný bod P, hodnotu jasu v tomto bodě porovnáváme s hodnotami jasu v modrých bodech, čísla těchto bodů mají význam v algoritmu.

ověřit pouhým vyzkoušením všech možností. Algoritmus však při volbě vhodných parametrů r a n umožňuje i vysokorychlostní test, který efektivně vyřadí většinu nevýznamných bodů. Zvolme tedy například

$$r = 6, \tag{4.34}$$

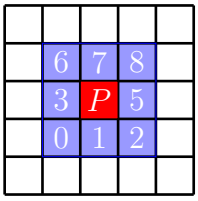
$$n = 12, \tag{4.35}$$

což jsou hodnoty používané i v původní práci Rostena a Drummonda. Pro tyto hodnoty je Bresenhamova kružnice 16prvková množina. Schéma této situace je znázorněno na obrázku 4.1. Pokud hodnotu $i(P)+t$, resp. $i(P)-t$ porovnáme pouze se čtyřmi pixely s čísly $1, 5, 9$ a 13, pak vidíme, že podmínku (4.33) lze splnit jen tehdy, když alespoň tři z těchto čtyř pixelů mají všechny vyšší, resp. nižší intenzitu než $i(P)+t$, resp. $i(P)-t$. Ověření úplné podmínky (4.33) provedeme pouze za předpokladu, že je splněn vysokorychlostní test.

Detektor FAST je velmi efektivní algoritmus, ale má také několik nevýhod [53, 54]:

1. vysokorychlostní test se obtížně zobecňuje pro $n < 12$;
2. volba a pořadí vyhodnocování pixelů ve vysokorychlostním testu ovlivňuje efektivitu tohoto testu a nutí nás tak implicitně odhadovat rozložení významných bodů v obrazu;
3. informace zjištěná vysokorychlostním testem se zahazuje, a nepřenáší se tedy do dalších testů;
4. více významných bodů obvykle leží vedle sebe;
5. nepřináší žádnou míru kvality nalezených bodů;
6. nalezené významné body nesplňují důležité požadavky na invarianci vůči rotaci a škálování.

Autoři algoritmu pro řešení nedostatků 1 - 3 navrhují obohacení algoritmu metodou strojového učení, při čemž ze sady vstupních obrazů (získaných nejlépe konkrétně pro požadovanou aplikaci) vygenerují rozhodovací strom, který je následně převeden na zdrojový kód v jazyce C a zkompilován. Výsledný program je navíc možné značně optimalizovat díky znalosti výkonnostních dat získaných zpracováním vstupních obrazů.



Obrázek 4.2: Schéma patch deskriptoru pro $n = 3$ – červený bod označuje významný bod P; modře jsou označeny body, které tvoří vektor deskriptoru; čísla v bodech označují jejich pořadí ve vektoru, bod P je zároveň bodem p_4.

Problémy 4 a 5 spolu do jisté míry souvisí. Pokud bychom dokázali ohodnotit kvalitu nalezených bodů, mohli bychom z každé skupiny spoluležících významných bodů vybrat ty, které mají nejvyšší kvalitu, a tím zásadně omezit problém významných bodů ležících vedle sebe. Rosten a Drummond ve své práci navrhují několik vztahů pro dodatečné posouzení kvality nalezených bodů, z nichž hlavní doporučovaný je založen na sumě absolutních rozdílů jasů mezi pixely v souvislém oblouku a středovým pixelem.

Později v této kapitole uvedeme rozšíření algoritmu FAST, které tyto problémy řeší jiným způsobem. Předtím však ještě vysvětlíme princip deskriptorů, předvedeme některé jejich typy a zmíníme základní strategie řešení problémů s invariancí.

4.4. Popis bodu pomocí deskriptoru

Dalším důležitým tématem, které se týká významných bodů je deskripce těchto bodů. V mnoha aplikacích hledáme významné body proto, abychom dokázali na více snímcích rozpoznat stejný objekt. Jedná se například o aplikace, ve kterých sledujeme polohu jednoho objektu na videu nebo vyhledáváme nějaký objekt jednoho obrazu v jiném obraze. V takových chvílích potřebujeme nejen významné body nalézt, ale také je dokázat nějakým způsobem popsat tak, aby následné porovnávání bylo výpočetně efektivní a aby splňovalo další požadavky, které na něj budeme klást. Mezi tyto požadavky obvykle patří, že popis bodu musí být nezávislý na translaci a otočení a někdy navíc i na škálování nebo změně jasu a kontrastu obrazu. Kdyby tyto požadavky nebyly splněny, nebylo by možné stejné objekty rozpoznat, pokud by mezi dvěma snímky došlo k nějaké transformaci. Jinými slovy požadujeme, aby se deskriptor jednoho bodu příliš neměnil, pokud na obraz aplikujeme nějakou transformaci. Zároveň ale chceme, aby popis jednoho bodu byl dostatečně odlišný od ostatních bodů v obrazu a nedocházelo tak k falešným shodám.

4.4.1. Patch deskriptor

Nejjednodušší myšlenkou pro popis významného bodu je přímé použití obrazové matice v nějakém okolí významného bodu – obrazové okénko (*image patch*). Mějme obraz i a v něm významný bod $P = [x_P, y_P]$. K tomuto bodu přiřaďme n^2-rozměrný vektor $\mathbf{p} = (p_0, \ldots, p_{n^2-1})^\intercal$ sestavený z části obrazové matice i, kde n je liché číslo, které splňuje $2 < n \in \mathbb{N}$. Tento vektor sestavíme z okolí bodu P podle rovnice

$$p_j = i\left(x_P + (j \bmod n) - \left\lfloor \frac{n}{2} \right\rfloor, y_P + \left\lfloor \frac{j}{n} \right\rfloor - \left\lfloor \frac{n}{2} \right\rfloor\right). \tag{4.36}$$

Princip tohoto deskriptoru je zobrazen na obrázku 4.2.

Dva takto popsané body zájmu můžeme porovnávat například podle čtverce eukleidovské metriky. Výhoda použití čtverce ve výpočtu je v tom, že výpočet odmocniny je na většině procesorů značně neefektivní, a proto je lepší se mu vyhnout. Tedy odlišnost dvou deskriptorů $\mathbf{p_1}$ a $\mathbf{p_2}$ je dána vztahem

$$d(\mathbf{p_1}, \mathbf{p_2}) = \|\mathbf{p_1} - \mathbf{p_2}\|_2^2 . \tag{4.37}$$

Takový deskriptor je invariantní vůči translaci, ale už není invariantní vůči rotaci, škálování, či změnám v jasu a kontrastu. Uvažujme, že obraz i transformujeme podle vztahu

$$i'(x, y) = \alpha i(x, y) + \beta. \tag{4.38}$$

Aby náš deskriptor byl invariantní vůči tomuto typu transformace, můžeme v něm provést několik změn. Nejdříve proveďme

$$\mathbf{p}' = \mathbf{p} - \bar{\mathbf{p}}. \tag{4.39}$$

Symbol $\bar{\mathbf{p}}$ značí průměr čísel $p_0, \ldots, p_{n^2-1}$. Tímto získáváme deskriptor, který je invariantní vůči přičtení konstanty β. Dále zaveďme

$$\mathbf{p}'' = \frac{\mathbf{p}'}{\|\mathbf{p}'\|}. \tag{4.40}$$

Deskriptor $\mathbf{p}''$ je invariantní vůči násobení konstantou α. Navíc pro dva deskriptory $\mathbf{p_1''}$ a $\mathbf{p_2''}$ platí, že

$$\langle \mathbf{p_1''}, \mathbf{p_2''} \rangle = \cos \varphi, \tag{4.41}$$

a tedy hodnota jejich skalárního součinu je největší (rovna jedné), pokud jsou vektory totožné a nejmenší (rovna minus jedné), pokud jsou navzájem opačné. Tedy skalární součin je možno chápat jako míru podobnosti těchto dvou deskriptorů. Těmito úpravami jsme získali deskriptor, který je invariantní na lineární změny v jasu obrazu [68].

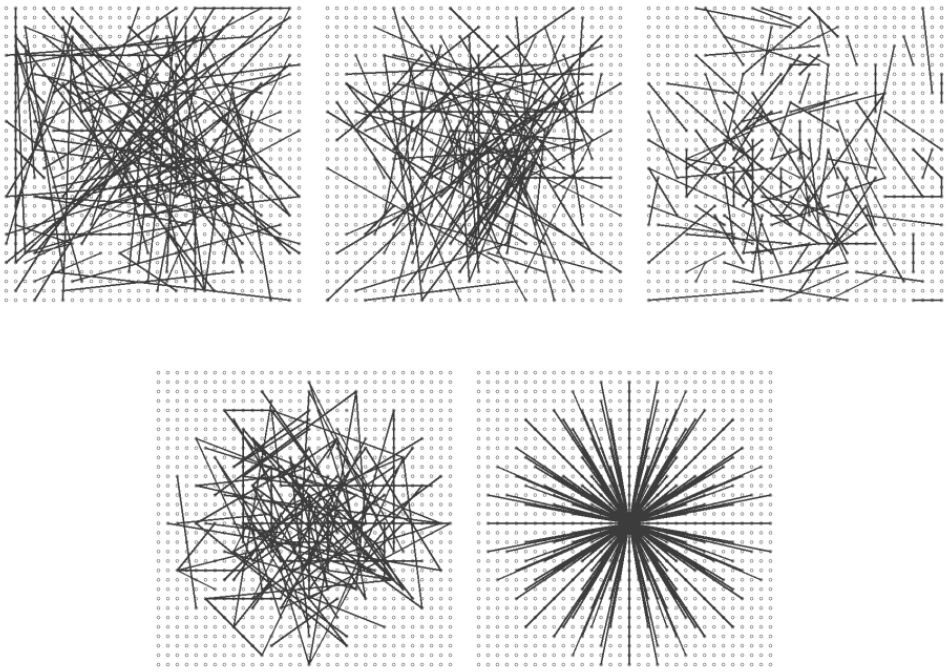
Abychom získali deskriptor, který je invariantní vůči rotaci, můžeme jej například navíc doplnit o tzv. dominantní směr. Jedná se o dvojrozměrný jednotkový vektor, který je pro daný významný bod stanoven tak, aby jeho směr byl pokud možno závislý pouze na rotaci obrazu. Pokud takový vektor získáme, můžeme porovnávané deskriptory otočit tak, aby jejich dominantní směry byly totožné a my tak porovnávali deskriptory, které mají stejnou orientaci.

Nejjednodušší způsob, jak získat dominantní směr pro patch deskriptor je spočítání gradientu ve významném bodě.

Poznamenejme, že spolu s rotací tohoto deskriptoru se vážou jistá omezení. Vzhledem k tomu, že se pohybujeme v rastru a náš deskriptor má tvar čtverce, je možné jej přesně otočit pouze o násobky 90°. Je tedy nutné se buď omezit na pouze čtyři dominantní směry, a ztratit tím přesnost, nebo rotovat přesně a nové pixely dopočítávat bilineární interpolací.

Invarianci vůči škálování je možné řešit pomocí škálovací pyramidy. Jedná se o techniku, kdy se deskriptor ukládá v postupně zmenšovaných rozlišeních, a díky tomu se získá větší rozsah škál pro porovnávání.

Na závěr této části dodejme, že patch deskriptor je základem deskriptoru MOPS (*Multiscale Oriented PatcheS descriptor*) [68], který využívá všechny výše uvedené techniky.



Obrázek 4.3: Strategie pro výběr dvojic v binárních testech deskriptoru BRIEF [55].

4.4.2. BRIEF deskriptor

Výpočetně velmi efektivní, avšak stále dostatečně přesný deskriptor může být BRIEF (*Binary Robust Independent Elementary Features*) [55]. Tento deskriptor, jak již napovídá jeho název, je tvořen řetězcem binárních hodnot. Díky tomu je paměťově kompaktní a porovnávání dvou takových deskriptorů může být provedeno Hammingovou metrikou, kterou lze pro dnešní procesory implementovat znatelně rychlejšími instrukcemi, než jiné běžně používané metriky.

Princip deskriptoru BRIEF je založen na provedení několika binárních testů, jejichž výsledek stanoví výsledný deskriptor. Mějme obraz i a v něm významný bod $P = [x_P, y_P]$. Pro tento bod vezměme obdobně jako v případě patch deskriptoru z části 4.4.1 obrazové okénko, které označíme $\mathbf{p}$. Definujme binární test jako zobrazení τ následujícím způsobem

$$\tau(\mathbf{p}, X, Y) = \begin{cases} 1 & i(X) < i(Y), \\ 0 & \text{jinak}, \end{cases} \tag{4.42}$$

kde body X a Y jsou body patřící do obrazového okénka $\mathbf{p}$ a hodnoty $i(X)$ a $i(Y)$ jsou skutečné hodnoty intenzity obrazu v daných bodech. Volba množiny dvojic bodů X a Y jednoznačně definuje množinu binárních testů, jejich hodnoty postupně umístíme do vektoru, který pojmenujeme BRIEF deskriptor [55].

Autoři ve svém článku navrhují několik strategií pro volbu dvojic v binárních testech. Tyto strategie je možné vidět na obrázku 4.3. Klíčové je, že ač jsou dvojice bodů pro binární testy vybírány různě, jsou pro každý významný bod vybrány stejně.

Doporučovaný počet dvojic bodů je 128, 256 nebo 512. Autoři rovněž doporučují obraz před deskripcí vyhladit konvolucí s diskretizovanou Gaussovou dvojrozměrnou funkcí.

Pro vyhodnocení podobnosti dvou BRIEF deskriptorů s velkou výhodou použijeme Hammingovu metriku.

Definice 4.4.1 (Hammingova metrika)**.** Nechť **u** a **v** jsou vektory binárních číslic 0 a 1. Pak Hammingovu metriku ρ_H těchto dvou vektorů definujeme jako počet souřadnic, v nichž se vektory liší.

Velkou předností této metriky je, že je možné ji na počítači implementovat pouze instrukcí `XOR` následovanou instrukcí `BIT COUNT`.

4.5. Detektor ORB

V této části vezmeme jednoduchý detektor FAST z kapitoly 4.3, který mírně pozměníme a doplníme jej o modifikovanou variantu deskriptoru BRIEF z kapitoly 4.4.2, abychom tak vytvořili robustní detektor ORB [54], který se může poměřovat i s dnes nejpoužívanějšími detektory SIFT a SURF.

Jak již bylo zmíněno v části 4.3, detektor FAST doprovází řada nevýhod. Autoři detektoru ORB navrhují následující vylepšení, která většinu nevýhod eliminují.

Celý proces začíná detekcí algoritmem FAST s poloměrem Bresenhamovy kružnice $r = 9$. Původní algoritmus neposkytuje žádnou hodnoty pro posouzení kvality nalezených bodů. Zde se pro doplnění použije hodnota odezvy Harrisova detektoru z části 4.1. Pro nalezení n významných bodů se obvykle nejdříve nastaví hodnota prahu t v detektoru FAST dostatečně nízko, aby algoritmus našel alespoň n bodů. Poté se těmto bodům přiřadí jejich hodnota Harrisovy odezvy a ze sestupně seřazeného seznamu těchto bodů se vybere prvních n [54].

FAST neposkytuje body invariantní vůči změně měřítka. To autoři řeší použitím metody škálovací pyramidy, která byla zmíněna v části o patch deskriptoru 4.4.1.

Pro řešení orientace detekovaných bodů, a tedy i invariance vůči rotaci, je autory článku navrženo řešení užitím těžiště intenzity. To předpokládá, že intenzita ve významném bodě není vyvážená do jeho středu. Momenty intenzity vybraného obrazového okénka okolo významného bodu P tak definujeme vztahem

$$m_{pq} = \sum_{x,y} x^p y^q i(x, y) \tag{4.43}$$

a s pomocí těchto momentů je možné stanovit těžiště intenzity jako bod

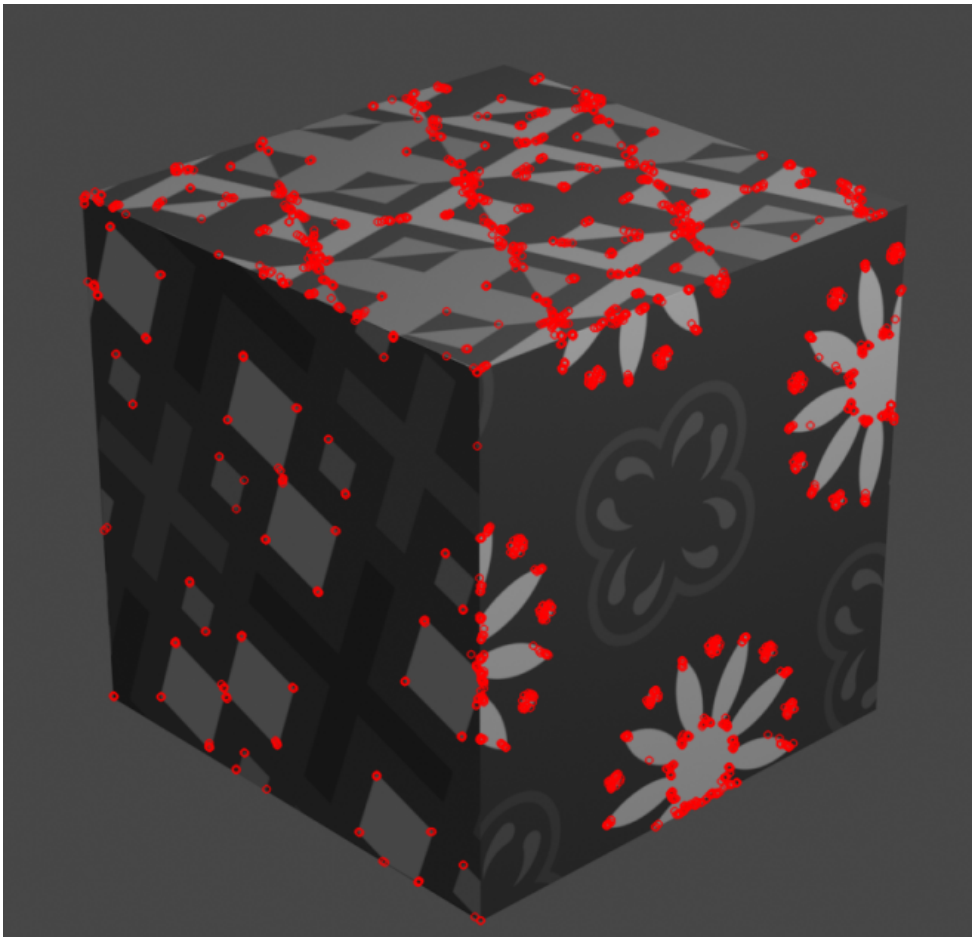
$$T = \left[\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right]. \tag{4.44}$$

Z tohoto těžiště můžeme zkonstruovat vektor $(T - P)$, jehož úhel lze použít pro orientaci daného významného bodu. Zdůrazněme, že tento úhel lze spočítat ze vztahu

$$\varphi = \arctan_2(m_{01}, m_{10}), \tag{4.45}$$

kde funkce $\arctan_2(x, y)$ je definována jako úhel v eukleidovské rovině mezi kladnou osou x a polohovým vektorem $\mathbf{x} = (x, y)^\intercal$.

Aby bylo možné takto vylepšený detektor FAST použít i pro porovnávání významných bodů, je nutné k němu přiřadit vhodný deskriptor. Autoři algoritmu pro tento účel



Obrázek 4.4: Významné body nalezené algoritmem ORB v obrázku krychle.

zvolili deskriptor BRIEF. Pro orientaci tohoto deskriptoru se použije rotace původního obrazového okénka rotační maticí $\mathbf{R}$ s úhlem φ. Pro rotaci se doporučuje diskretizace úhlu na násobky $\frac{2\pi}{30}$.

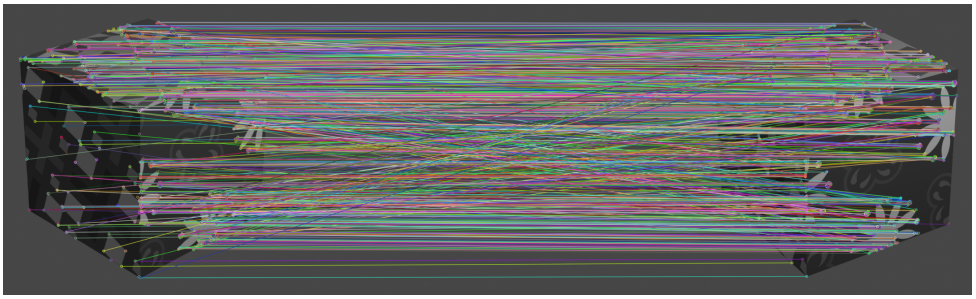
Dodejme, že autoři navrhují ještě další drobné úpravy tohoto deskriptoru, které celkově zvyšují jeho účinnost, ale jsou již nad rámec této práce.

Na obrázku 4.4 je možné pozorovat významné body detekované algoritmem ORB.

4.6. Korespondence bodů

Posledním teoretickým tématem této kapitoly bude předvedení aplikace významných bodů obrazu pro nalezení množiny korespondujících si (odpovídajících si) bodů v obrazu. Důležitost této úlohy ilustruje legenda, že Takeo Kanade, významný japonský vědec zabývající se počítačovým viděním, jednou na otázku, jaké jsou tři nejpodstatnější problémy počítačového vidění, odpověděl třikrát slovem korespondence [69].

Mějme dva obrazy i_1 a i_2 zobrazující stejnou scénu. Obraz i_1 může být například hlavní obraz a i_2 nějaký jeho výřez, nebo se může jednat o dva obrazy pořízené mírně odlišně nastavenou kamerou. Našim cílem je nyní nalézt co největší množinu dvojic bodů z obrazů i_1 a i_2, pro které platí, že jsou obrazem stejného prostorového bodu scény.



Obrázek 4.5: Páry významných bodů získané metodou párování hrubou silou.

K řešení tohoto problému přistoupíme použitím detekce významných bodů a jejich deskripce užitím deskriptorů. Nejdříve v obou obrazech provedeme detekci významných bodů a jejich deskripci například užitím algoritmu ORB. Získáme tak dvě množiny významných bodů P_1 pro obraz i_1 a P_2 pro obraz i_2.

Následně je potřeba ze všech dvojic takto nalezených bodů $P_1 \times P_2$ vybrat ty, které si skutečně odpovídají. Dnes standardně používaným přístupem je „párování hrubou silou“ (*Brute-Force Matching*). Tato metoda nejdříve seřadí významné body množiny P_1 dle jejich kvality a následně je testuje vůči všem bodům množiny P_2. Jako odpovídající si bod je zvolen ten, ve kterém se deskriptory obou testovaných bodů nejméně liší. Tato metoda prochází všechny dvojice párovaných bodů a je tak velmi přesná. Její nevýhodou je však kvadratická výpočetní složitost. Pro kritické aplikace běžící v reálném čase lze tak použít metody, které pro vhodný výběr používají náhodu.

Výsledek párování hrubou silou je zobrazen na obrázku 4.5.

Dodejme, že řešení tohoto problému bude klíčovým vstupem v kapitole 5, zejména pak v části 5.1.2.

4.7. Detekce a párování bodů v knihovně OpenCV

V této části popíšeme, jak lze významné body detekovat a párovat s využitím knihovny OpenCV. Knihovna OpenCV implementuje řadu běžných detekčních algoritmů a mezi nimi již zmiňovaný algoritmus ORB. Pro vytvoření detektoru ORB je v knihovně k dispozici statická metoda:

```
Ptr<ORB> orb = cv::ORB::create(int nfeatures)
```

Tato metoda nabízí celou řadu parametrů vytvořeného detektoru, z nichž nejdůležitějším je

`nfeatures` – celočíselná hodnota počtu požadovaných významných bodů.

Návratovou hodnotou této metody je vytvořený parametrizovaný detektor. Ten, stejně jako ostatní detektor významných bodů v knihovně OpenCV, implementuje metodu:

```
void detectAndCompute(InputArray image, InputArray mask, std::vector<
    KeyPoint> &keypoints, OutputArray descriptors)
```

Tato metoda bere na vstupu následující parametry.

`image` – vstupní obrazová matice;

`mask` – maska umožňující vybrat pouze část obrazu pro detekci;

`keypoints` – referenci na vektor, do kterého má detektor vložit nalezené významné body;

`descriptors` – výstupní matice deskriptorů příslušných k nalezeným bodům.

Pro následné párování významných bodů je v knihovně dostupná třída `cv::BFMatcher` implementující metodu pro párování hrubou silou. Vytvoření této párovací třídy se děje opět pomocí statické metody:

```
Ptr<BFMatcher> matcher = cv::BFMatcher::create(int normType, bool
    crossCheck)
```

Tato metoda na vstupu přijímá tyto argumenty:

`normType` – celočíselná hodnota ztotožňující se s určitou normou použitou pro porovnávání deskriptorů, v případě, že používáme detektor ORB, je nutné zvolit `cv::NORM_HAMMING`;

`crossCheck` – pravdivostní hodnota, která určuje, zda mají být za odpovídající si body prohlášeny pouze ty, které si odpovídají vzájemně (oboustranně).

Výstupem je vytvořený porovnávač. Ten pro porovnání bodů zájmu používá metodu:

```
void match(InputArray queryDescriptors, InputArray trainDescriptors, std::
    vector<DMatch> &matches)
```

Vstupem této metody jsou parametry:

`queryDescriptors` – matice deskriptorů jednoho obrazu;

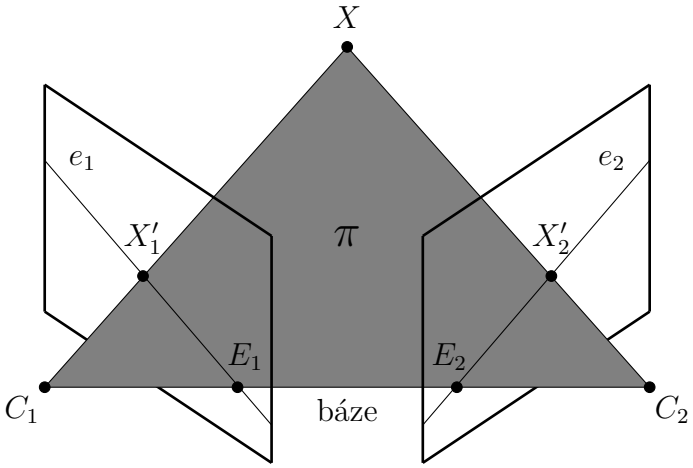
`trainDescriptors` – matice deskriptorů druhého obrazu;

`matches` – reference na vektor, do kterého má porovnávač vložit nalezené dvojice bodů.

Celý proces je pak zobrazen v ukázce 4.1.

```cpp
#include <vector>
#include <opencv2/opencv.hpp>

void detect_and_match(const cv::Mat &image1, const cv::Mat &image2,
    std::vector<cv::DMatch> &matches)
{
    // vytvorime detektor pro nalezeni 1000 vyznamnych bodu
    auto orb = cv::ORB::Create(1000);

    // detekujeme vyznamne body a stanovime jejich deskriptory pro oba
    obrazy
    std::vector<cv::KeyPoint> keypoints1;
    std::vector<cv::KeyPoint> keypoints2;
    cv::Mat descriptors1;
    cv::Mat descriptors2;
    orb->detectAndCompute(img1, cv::noArray(), keypoints1, descriptors1);
    orb->detectAndCompute(img2, cv::noArray(), keypoints2, descriptors2);

    // vytvorime vhodny porovnavac vyznamnych bodu
    auto bfMatcher = cv::BFMatcher(cv::NORM_HAMMING, true);

    // nalezneme odpovidajici si body
    bfMatcher.match(descriptors1, descriptors2, matches);
}
```

Kód 4.1: Detekce a párování významných bodů



Obrázek 5.1: Epipolární geometrie

5. Epipolární geometrie

Kapitola 3 popisuje situaci, ve které nějakou scénu pozorujeme jedinou kamerou. Takovou situaci jsme dokázali popsat modelem perspektivní kamery, který byl reprezentován maticí. Nyní na základě tohoto modelu vybudujeme teorii nutnou pro popis situace, ve kterém na jednu scénu pohlížíme ze dvou směrů. Těmto směrům říkáme pohledy. Oba pohledy mohou být pořízeny zároveň dvěma kamerami, nebo postupně jednou kamerou. Tyto přístupy jsou geometricky ekvivalentní, a proto mezi nimi není potřeba rozlišovat.

Každý z těchto pohledů můžeme charakterizovat vlastní maticí perspektivní kamery $\mathbf{P_1}$ a $\mathbf{P_2}$. Nechť X je trojrozměrný bod scény a $\mathbf{X}$ jsou jeho homogenní souřadnice. V prvním pohledu se bod X zobrazí na bod X_1' podle rovnice $\mathbf{X_1'} = \mathbf{P_1X}$ a v druhém ekvivalentně na bod X_2' podle vztahu $\mathbf{X_2'} = \mathbf{P_2X}$. O bodech X_1' a X_2' říkáme, že si odpovídají, protože jsou obrazem stejného bodu scény X. Všimneme si, že tato situace je totožná s popisem v části 4.6.

V této kapitole odpovíme na otázky, jak poloha bodu v obraze získaného z jednoho pohledu omezuje polohu odpovídajícího bodu v obraze druhého pohledu; jak určit matice obou kamer, je-li dána množina vzájemně si odpovídajících bodů pro oba pohledy a jak stanovit skutečnou trojrozměrnou pozici bodu scény, který je vzorem odpovídajících si obrazových bodů v obou pohledech.

Kapitola vychází z [14, 18, 48, 70].

5.1. Fundamentální matice

Zvolme libovolný bod scény $X \in \mathbb{P}^3$. Tento bod sledujeme, jak již bylo zmíněno, z pohledu dvou různých perspektivních kamer, jejichž středy promítání jsou body C_1 a C_2. Přímka spojující tyto dva body se nazývá báze nebo také bázová přímka. Dále diskutujme jen body X, které na bázové přímce neleží.

Průmět bodu X pro první kameru značíme X_1' a určíme jej tak, že bod X spojíme se středem C_1 přímkou. Bod X_1' je pak průsečíkem této přímky s obrazovou rovinou první kamery. Spojnice bodů X a C_1 se v obraze druhé kamery zobrazí jako přímka, kterou označíme e_2. Obdobně bod X_2' získáme jako průsečík obrazové roviny druhé kamery

s přímkou spojující body X a C_2, která se v prvním obraze zobrazí jako přímka e_1. Tuto situaci názorně ilustruje obrázek 5.1.

Přímky e_1 a e_2 nazýváme epipolární přímky. Zde odpovídáme na první otázku z úvodu této kapitoly. Poloha bodu v jednom z obrazů určuje v druhém obraze epipolární přímku, na které musí obraz tohoto bodu nutně ležet.

Bod X spolu s bází určují rovinu π, které říkáme epipolární rovina. V této rovině rovněž leží epipolární přímky e_1 a e_2, které jsou navíc průsečnicemi roviny π a jednotlivých obrazových rovin. Průsečíky epipolárních přímek e_1 a e_2 s bází nazýváme epipóly a značíme po řadě E_1 a E_2. Zdůrazněme, že všechny epipolární přímky jednoho pohledu musejí ležet v nějaké rovině, ve které leží i bázová přímka a protože žádná z epipolárních přímek není s bázovou přímkou rovnoběžná, musejí mít vždy stejný společný bod, a tím bodem je právě epipól tohoto pohledu.

Speciálním případem, kdy transformací mezi oběma kamerami je pouze translace rovnoběžná s obrazovou rovinou a rotace kolem osy kolmé k obrazové rovině se dále nebudeme zabývat. Pouze podotkneme, že v tomto zvláštním případě jsou epipolární přímky skutečně rovnoběžné s bází a epipóly tak leží v nevlastním bodě.

Výše popsaný vztah mezi dvěma obrazy nazýváme epipolární geometrie. Její algebraickou reprezentací je fundamentální matice, kterou dále v textu geometricky odvodíme ze zobrazení mezi obrazovým bodem a jeho epipolární přímkou a určíme její vlastnosti.

Jak už bylo řečeno, ke každému bodu obrazu X_1 existuje epipolární přímka e_2, na které leží bod X_2. Tedy můžeme definovat zobrazení $X_1 \to e_2$ z obrazového bodu na jemu odpovídající epipolární přímku. Ukažme nyní, že toto zobrazení lze reprezentovat maticí.

Uvažujme libovolnou rovinu σ v prostoru, která neprochází žádným z bodů C_1 a C_2. Nechť na této rovině leží bod scény X, který je vzorem obrazového bodu X_1'. Tento bod je rovněž vzorem obrazového bodu X_2'. Zároveň zjevně platí, že pro libovolný jiný bod $Y \neq X$ existují odlišné body $Y_1' \neq X_1'$ a $Y_2' \neq X_2'$, které jsou obrazem tohoto bodu v daných pohledech (zobrazovací paprsek spojující body C_1 a Y (respektive C_2 a Y) má totiž vždy právě jeden průsečík s rovinou σ). Volbou roviny σ jsme tak schopni zavést zobrazení $X_1' \to X_2'$, které je vzájemně jednoznačné. Toto zobrazení nazýváme přenos bodu přes rovinu a značíme H_σ. Jelikož zobrazení mezi X_1' a X je projektivní, stejně jako mezi body X_2' a X, je zobrazení H_σ dvojrozměrná homografie. Přenos bodu přes rovinu můžeme zapsat maticovým zápisem

$$\mathbf{X_2'} = \mathbf{H}_\sigma \mathbf{X_1'}, \tag{5.1}$$

v němž matice $\mathbf{H}_\sigma$ je typu 3×3.

Dále sestrojme epipolární přímku e_2, která odpovídá bodu X_1'. Tato přímka prochází body X_2' a E_2 a lze ji proto zapsat vektorovým součinem

$$\mathbf{e_2} = \mathbf{E_2} \times \mathbf{X_2'} = [\mathbf{E_2}]_\times \mathbf{X_2'}. \tag{5.2}$$

Protože X_2' lze zapsat ve tvaru uvedeném ve vztahu (5.1), získáváme dosazením do (5.2) epipolární přímku vyjádřenou jako

$$\mathbf{e_2} = [\mathbf{E_2}]_\times \mathbf{H}_\sigma \mathbf{X_1'} = \mathbf{F} \mathbf{X_1'}. \tag{5.3}$$

Matici $\mathbf{F}$ typu 3×3 ze vztahu (5.3) nazýváme fundamentální matice. Ačkoli bylo odvození fundamentální matice provedeno s využitím roviny σ, zdůrazněme, že existence matice $\mathbf{F}$

není na této rovině závislá. Rovina σ se používá pouze pro definování zobrazení z jednoho obrazu na druhý. Kromě geometrického odvození lze pro získání fundamentální matice použít i algebraickou cestu. Tento postup je více popsán například v [14]. Geometrické odvození popsané výše čerpá rovněž z [14].

Dále budou uvedeny některé významné vlastnosti fundamentální matice.

5.1.1. Vybrané vlastnosti fundamentální matice

Nejvýznamnější vlastností fundamentální matice je platnost následující rovnosti

$$\mathbf{X_2'}^{\mathsf{T}}\mathbf{F}\mathbf{X_1'} = 0, \tag{5.4}$$

ve které X_1' a X_2' je dvojice odpovídajících si bodů. Jestliže totiž X_2' je bod, který na druhém obraze odpovídá bodu X_1', pak tento bod zřejmě musí ležet na epipolární přímce $e_2 = \mathbf{F}\mathbf{X_1'}$. Jak již bylo odvozeno v kapitole 1.2, pokud bod X_2' leží na přímce e_2, musí pro něj platit rovnost $\mathbf{X_2'}^{\mathsf{T}} e_2 = \mathbf{X_2'}^{\mathsf{T}}\mathbf{F}\mathbf{X_1'} = 0$ [14].

Tento výsledek umožňuje vypočítat fundamentální matici bez použití matic jednotlivých kamer pouze se znalostí množiny nějakých odpovídajících si bodů. Jak výpočet provést bude předvedeno v části 5.1.2.

Pro epipóly navíc platí rovnosti

$$\mathbf{F}\mathbf{E_1} = \mathbf{o}, \tag{5.5}$$

$$\mathbf{F}^{\mathsf{T}}\mathbf{E_2} = \mathbf{o}, \tag{5.6}$$

ve kterých $\mathbf{o}$ je nulový vektor [14].

Zobrazení definované fundamentální maticí $\mathbf{F}$ zobrazuje dvojrozměrnou rovinu na jednoparametrickou soustavu přímek (všechny epipolární přímky totiž procházejí epipólem a mají tak pouze jeden stupeň volnosti), a proto musí mít hodnost menší než tři. Protože hodnost matice $[\mathbf{E_2}]_\times$ je dvě a hodnost matice $\mathbf{H}_\sigma$ je tři, platí, že

$$\operatorname{rank}\mathbf{F} = 2. \tag{5.7}$$

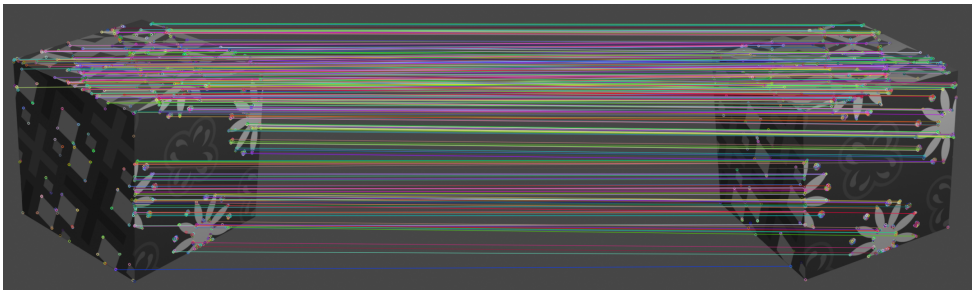
Zároveň platí, že matice $\mathbf{F}$ má pouze sedm stupňů volnosti. Zobrazovací matice typu 3×3 má pouze osm nezávislých poměrů, protože má devět prvků a škálování je nevýznamné; zároveň však, jelikož $\operatorname{rank}\mathbf{F} = 2 < 3$, musí splňovat podmínku

$$\det\mathbf{F} = 0, \tag{5.8}$$

což odebírá jeden stupeň volnosti [14]. Poznamenejme, že podmínku (5.8) budeme dále v textu nazývat podmínkou singularity fundamentální matice.

Nechť $\mathbf{F}$ je fundamentální matice popisující epipolární geometrii mezi pohledy s maticemi kamer $\mathbf{P_1}$ a $\mathbf{P_2}$, pak platí, že fundamentální maticí pro dvojici pohledů $\mathbf{P_2}$ a $\mathbf{P_1}$ (tj. stejné matice kamer v opačném pořadí) je matice $\mathbf{F}^{\mathsf{T}}$ [14].

Známe-li fundamentální matici mezi dvěma obrazy. Je možné ji využít pro filtrování párů významných bodů. Páry významných bodů totiž odpovídají stejnému bodu prostoru, a tak musejí vyhovovat rovnici 5.4. Obrázek 5.2 zobrazuje výsledek tohoto filtrování.



Obrázek 5.2: Páry významných bodů z obrázku 4.5 filtrované s pomocí fundamentální matice.

5.1.2. Výpočet fundamentální matice

Při znalosti matic kamer jednotlivých pohledů je výpočet fundamentální matice mezi těmito pohledy jednoduchý, neboť pro obecné kamery $\mathbf{P_1}$ a $\mathbf{P_2}$ platí vztah

$$\mathbf{F} = [\mathbf{E_2}]_\times \mathbf{P_2}\mathbf{P_1^+}. \tag{5.9}$$

V tomto vztahu symbol $\mathbf{P_1^+}$ značí Mooreovu-Penroseovu pseudoinverzní matici [71] k matici $\mathbf{P_1}$. Bod E_2 získáme jako $\mathbf{E_2} = \mathbf{P_2}\mathbf{C_1}$ za podmínky $\mathbf{P_1}\mathbf{C_1} = \mathbf{o}$, kde C_1 je střed promítání první kamery a $\mathbf{o}$ je nulový vektor.

Komplikovanější situace nastává, když chceme výpočet provést pouze s využitím nějaké známé množiny odpovídajících si bodů. Mějme dva odpovídající si obrazové body $X_1' = [x_1, y_1]$ a $X_2' = [x_2, y_2]$, u kterých navíc požadujeme, že jsou vlastní. Dále uvažujme fundamentální matici $\mathbf{F} = (f_{ij})$. Dosazením do vztahu (5.4) obdržíme rovnici

$$x_1x_2f_{11} + y_1x_2f_{12} + x_2f_{13} + x_1y_2f_{21} + y_1y_2f_{22} + y_2f_{23} + x_1f_{31} + y_1f_{32} + f_{33} = 0. \tag{5.10}$$

Známe-li n dvojic odpovídajících si vlastních obrazových bodů $X_1'^i = [x_1^i, y_1^i]$ a $X_2'^i = [x_2^i, y_2^i]$, kde $i = 1, \ldots, n$, pak tímto způsobem získáme soustavu n lineárních rovnic, kterou můžeme zapsat v maticovém tvaru

$$\mathbf{Af} = \begin{pmatrix} x_1^1x_2^1 & y_1^1x_2^1 & x_2^1 & x_1^1y_2^1 & y_1^1y_2^1 & y_2^1 & x_1^1 & y_1^1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^nx_2^n & y_1^nx_2^n & x_2^n & x_1^ny_2^n & y_1^ny_2^n & y_2^n & x_1^n & y_1^n & 1 \end{pmatrix} \mathbf{f} = \mathbf{o}. \tag{5.11}$$

Vektor $\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\intercal$ je vektor neznámých prvků matice $\mathbf{F}$.

Soustava lineárních rovnic (5.11) je homogenní, takže má vždy triviální (nulové) řešení. Z Frobeniovy věty víme, že pro existenci dalších (nenulových) řešení je zapotřebí, aby hodnost matice $\mathbf{A}$ byla nejvýše rovna osmi. Volbu konkrétního řešení pak provádíme s obvyklým požadavkem

$$\|\mathbf{f}\| = 1. \tag{5.12}$$

Nastat tak mohou tři případy:

1. $\operatorname{rank}\mathbf{A} < 8$, přičemž získáme parametricky popsanou množinu řešení;
2. $\operatorname{rank}\mathbf{A} = 8$, a tak spolu s podmínkou (5.12) existuje jediné řešení soustavy;
3. $\operatorname{rank}\mathbf{A} > 8$, kdy můžeme nalézt alespoň přibližné řešení ve smyslu metody nejmenších čtverců.

Ve 2. případě lze získat řešení přímo běžnými metodami pro řešení soustav lineárních rovnic. Případ 3 nastává, když vstupní data (odpovídající si dvojice bodů) nejsou dána přesně (v literatuře jsou taková data též označována jako zašumělá). V takové situaci je možné přibližné řešení ve smyslu metody nejmenších čtverců získat například užitím singulárního rozkladu (SVD) matice **A**.

Při řešení však musíme brát v úvahu, že singularita je významnou vlastností fundamentální matice, která například zaručuje, že se epipolární přímky budou protínat v jediném bodě. Nalezené řešení obecně nemusí definovat matici s hodností dvě, a proto musíme podniknout kroky, které tuto vlastnost vynutí. Obvykle tak nalezenou matici nahradíme nejbližší singulární maticí, přičemž vzdáleností zde rozumíme Frobeniovu normu rozdílu obou matic.

Definice 5.1.1. Nechť **A** je matice komplexních čísel typu $m \times n$, pak Frobeniovou normou této matice rozumíme číslo

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}, \tag{5.13}$$

přičemž číslo a_{ij} je prvek matice **A** na řádku i a ve sloupci j.

Postup, ve kterém řešení nejdříve nalezneme pomocí SVD a následně opravíme nahrazením nejbližší singulární maticí, je základem tzv. osmibodové metody. Tento postup dále rozvedeme v části popisující normalizovanou osmibodovou metodu. Ta jej navíc doplňuje o úpravu vstupních dat, která významně zlepšuje podmíněnost této úlohy.

V praxi se pro výpočet fundamentální matice používají nejčastěji dvě dobře známé metody – sedmibodová metoda a normalizovaná osmibodová metoda. Popis těchto metod je převzat z [14].

Sedmibodová metoda

Pokud matice **A** v soustavě rovnic (5.11) má hodnost sedm, lze z této soustavy stále získat přesné řešení s využitím podmínky singularity (5.8). Tento případ obyčejně nastává, když na vstupu použijeme pouze sedm dvojic odpovídajících si bodů, ale může k němu dojít i v případě degenerace vstupních dat.

Obecné řešení soustavy (5.11) za těchto podmínek a za podmínky (5.12) tvoří dvojrozměrný prostor, který lze popsat množinou vektorů

$$\mathbf{f} = \alpha\mathbf{f_1} + (1-\alpha)\mathbf{f_2}, \tag{5.14}$$

kde $\mathbf{f_1}$ a $\mathbf{f_2}$ jsou lineárně nezávislé jednotkové vektory a $\alpha \in (0;1)$ je parametr. Vektory $\mathbf{f_1}$ a $\mathbf{f_2}$ získáme například singulárním rozkladem matice **A**. Příslušnou matici **F** lze zapsat jako

$$\mathbf{F} = \alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2}, \tag{5.15}$$

kde matice $\mathbf{F_1}$ a $\mathbf{F_2}$ jsou matice sestavené ze složek vektorů $\mathbf{f_1}$ a $\mathbf{f_2}$. Dosazením takto popsané matice **F** do podmínky (5.8) obdržíme kubickou rovnici

$$\det\left(\alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2}\right) = 0 \tag{5.16}$$

s neznámou α. Tato rovnice má jedno nebo tři reálná řešení (komplexní řešení zanedbáváme) a tedy určuje jednu nebo tři možné fundamentální matice, mezi nimiž je potřeba vybrat tu nejvýhodnější později během aplikace. S výhodou je možné tento algoritmus použít v iteračních metodách, jakými jsou například metody založené na RANSACu (*RAndom SAmpling Consensus*) [72], a které umožňují efektivně porovnat jednotlivé matice.

Normalizovaná osmibodová metoda

Základem normalizované osmibodové metody, jak již bylo popsáno dříve v této kapitole, je přímé řešení soustavy (5.11) v případě, kdy hodnost matice $\mathbf{A}$ je alespoň osm. Vstupem tohoto algoritmu je proto nejméně osm dvojic vzájemně si odpovídajících bodů. Klíčovým krokem tohoto postupu je normalizace vstupních bodů, kterou provádíme pro každý obraz zvlášť. Cílem této transformace je translace bodů tak, aby jejich společné těžiště leželo v počátku souřadnicového systému a aby jejich průměrná vzdálenost od počátku byla rovna $\sqrt{2}$. Transformaci provedeme pomocí projektivního zobrazení daného maticemi $\mathbf{T_1}$ pro body v prvním obrazu a $\mathbf{T_2}$ pro body v druhém obrazu, přičemž užijeme vztahy

$$\widehat{\mathbf{X}}_\mathbf{1}^{\prime\mathbf{i}} = \mathbf{T_1}\mathbf{X}_\mathbf{1}^{\prime\mathbf{i}}, \tag{5.17}$$

$$\widehat{\mathbf{X}}_\mathbf{2}^{\prime\mathbf{i}} = \mathbf{T_2}\mathbf{X}_\mathbf{2}^{\prime\mathbf{i}}. \tag{5.18}$$

Tato úprava podstatně zlepšuje podmíněnost celé úlohy, a tím i stabilitu jejího řešení. Přidaná složitost transformace je vzhledem k celkovému užitku nevýznamná.

Po normalizaci vstupních dat následuje sestavení soustavy rovnic (5.11) tak, jak je uvedeno na začátku kapitoly 5.1.2 a její řešení za podmínky (5.12). To obvykle provádíme algoritmem SVD, kdy jako řešení bereme singulární vektor příslušný nejmenšímu singulárnímu číslu matice $\mathbf{A}$.

Získaný vektor $\widehat{\mathbf{f}}$ a z jeho složek sestavená matice $\widehat{\mathbf{F}}$ obecně nesplňují podmínku singularity, takže ji vynutíme nahrazením Frobeniovsky nejbližší maticí $\widehat{\mathbf{F}}'$, která tuto podmínku splňuje. K tomuto lze opět využít singulární rozklad. Nechť

$$\widehat{\mathbf{F}} = \widehat{\mathbf{U}}\widehat{\mathbf{D}}\widehat{\mathbf{V}}^\intercal \tag{5.19}$$

je singulární rozklad matice $\widehat{\mathbf{F}}$, ve kterém matice $\widehat{\mathbf{U}}$ a $\widehat{\mathbf{V}}$ jsou ortonormální a matice $\widehat{\mathbf{D}}$ je diagonální matice, která má na hlavní diagonále po řadě prvky $r \geq s \geq t$. Pak matice

$$\widehat{\mathbf{F}}' = \widehat{\mathbf{U}}\widehat{\mathbf{D}}'\widehat{\mathbf{V}}^\intercal \tag{5.20}$$

je singulární a minimalizuje Frobeniovu normu $||\widehat{\mathbf{F}} - \widehat{\mathbf{F}}'||$, jestliže $\widehat{\mathbf{D}}'$ je diagonální matice, která má na hlavní diagonále po řadě prvky r, s a 0.

Tento přístup pro prosazení singularity, ač jednoduchý a nenáročný na výpočet, nevede na numericky optimální řešení, jelikož ne všechny prvky fundamentální matice mají stejný význam a jsou stejně pevně svázány bodovou korespondencí mezi obrazy. Jiné přístupy se snaží minimalizovat Mahalanobisovu vzdálenost nebo používají iterativní metody pro přímé nalezení singulární matice, která řeší soustavu (5.11). Více o těchto odlišných postupech je možno nalézt v [73, 74].

Posledním krokem je denormalizace matice $\widehat{\mathbf{F}}'$, kterou učiníme zpětnou transformací podle vztahu

$$\mathbf{F} = \mathbf{T_2}^\intercal\widehat{\mathbf{F}}'\mathbf{T_1}. \tag{5.21}$$

Poznamenejme, že při použití normalizované osmibodové metody se doporučuje vynucení podmínky singularity před denormalizací.

Normalizovaná osmibodová metoda je považována za jednu z nejjednodušších metod pro výpočet fundamentální matice, která se snadno počítačově implementuje zejména v případě, že máme k dispozici potřebné algebraické metody.

5.2. Esenciální matice

Speciálním typem fundamentální matice je esenciální matice, kterou budeme značit $\mathbf{E}$. Tato matice popisuje vztah mezi dvěma pohledy převedenými do tzv. normalizovaných souřadnic. Uvažujme matici kamery $\mathbf{P} = \mathbf{K}\begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix}$ a obrazový bod $\mathbf{X}' = \mathbf{P}\mathbf{X}$. Známe-li kalibrační matici kamery $\mathbf{K}$, pak můžeme obrazový bod X' transformovat pomocí vztahu

$$\mathbf{X}'' = \mathbf{K}^{-1}\mathbf{X}'. \tag{5.22}$$

Bod X'' je vyjádřením bodu X' v normalizovaných souřadnicích. V podstatě se jedná o obraz bodu X při pohledu pomocí kamery, jejíž kalibrační maticí je jednotková matice. Matice kamery

$$\mathbf{P}' = \mathbf{K}^{-1}\mathbf{P} = \mathbf{K}^{-1}\mathbf{K}\begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix} = \mathbf{I}\begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix} = \begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix} \tag{5.23}$$

se nazývá normalizovaná matice kamery [14].

Ukažme, jak lze esenciální matici určit z fundamentální matice, známe-li kalibrační matice kamer obou pohledů. Jelikož esenciální matice je speciálním typem fundamentální matice, platí pro ni obdoba vztahu (5.4), kde oba body vyjádříme pomocí normalizovaných souřadnic

$$\mathbf{X}_2''^{\mathsf{T}}\mathbf{E}\mathbf{X}_1'' = 0. \tag{5.24}$$

Nahrazením normalizovaných bodů za původní dle vztahu (5.22) získáváme

$$(\mathbf{K}_2^{-1}\mathbf{X}_2')^{\mathsf{T}}\mathbf{E}(\mathbf{K}_1^{-1}\mathbf{X}_1') = \mathbf{X}_2'^{\mathsf{T}}(\mathbf{K}_2^{-\mathsf{T}}\mathbf{E}\mathbf{K}_1^{-1})\mathbf{X}_1' = 0, \tag{5.25}$$

tedy platí

$$\mathbf{F} = \mathbf{K}_2^{-\mathsf{T}}\mathbf{E}\mathbf{K}_1^{-1} \tag{5.26}$$

a

$$\mathbf{E} = \mathbf{K}_2^{\mathsf{T}}\mathbf{F}\mathbf{K}_1. \tag{5.27}$$

Jsou-li kalibrační matice obou kamer $\mathbf{K_1}$ a $\mathbf{K_2}$ známé, lze esenciální matici $\mathbf{E}$ snadno určit ze vztahu (5.27) [14].

Podotkneme, že zvolíme-li souřadný systém pro normalizované kamery tak, aby první měla matici $\mathbf{P}_1' = \begin{pmatrix}\mathbf{I} & \mathbf{o}\end{pmatrix}$ a druhá matici $\mathbf{P}_2' = \begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix}$, pak lze esenciální matici zapsat ve tvaru

$$\mathbf{E} = [\mathbf{t}]_\times\mathbf{R}. \tag{5.28}$$

Tato skutečnost je užitečným výsledkem, který v další části využijeme pro rekonstrukci scény [14].

5.3. Rekonstrukce scény

Rekonstrukce scény je proces, při kterém z několika snímků scény pořízených z různých pohledů sestavujeme zpětně trojrozměrný model scény. Vstupem je tedy množina snímků a výstupem je mračno bodů. V úvodu se omezíme pouze na případ, kdy scénu rekonstruujeme pouze ze dvou vstupních snímku. Ten následně rozšíříme na případ více vstupních snímků.

Máme-li dva snímky, je možné použít metody uvedené v části 5.1.2 pro výpočet fundamentální matice $\mathbf{F}$, ze které při znalosti kalibračních matic získáme esenciální matici $\mathbf{E}$ podle vztahu (5.27).

Dalším krokem rekonstrukce je určení matic obou kamer $\mathbf{P_1}$ a $\mathbf{P_2}$. Bez újmy na obecnosti budeme vycházet z předpokladu $\mathbf{P'_1} = \begin{pmatrix}\mathbf{I} & \mathbf{o}\end{pmatrix}$ a $\mathbf{P'_2} = \begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix}$. Zdůrazněme, že tento předpoklad lze vždy zajistit vhodnou volbou souřadného systému s normalizovanými souřadnicemi. Pro nalezení matic obou kamer, tak stačí stanovit vektor translace $\mathbf{t}$ a matici rotace $\mathbf{R}$.

Zajímavou vlastností esenciální matice, která přímo vyplývá z jejího rozkladu podle (5.28), je, že jedno její singulární číslo je rovno nule a zbylá dvě se rovnají, proto lze tuto matici rozložit SVD rozkladem na

$$\mathbf{E} = \mathbf{U}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{pmatrix}\mathbf{V}^\mathsf{T}. \tag{5.29}$$

S využitím tohoto rozkladu a rozkladu (5.28), stanovíme matici $[\mathbf{t}]_\times$ podle vztahu

$$[\mathbf{t}]_\times = \mathbf{U}\begin{pmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\mathbf{U}^\mathsf{T}. \tag{5.30}$$

Vektor $\mathbf{t}$ se z této antisymetrické matice získá přirozeně, jak bylo definováno ve vztahu (1.15). Tento vektor však známe až na znaménko. Rotační matici $\mathbf{R}$ nelze stanovit jednoznačně, ale vyhovuje jedné z následujících rovnic

$$\mathbf{R_1} = \mathbf{U}\begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\mathbf{V}^\mathsf{T}, \tag{5.31}$$

$$\mathbf{R_2} = \mathbf{U}\begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}^\mathsf{T}\mathbf{V}^\mathsf{T}. \tag{5.32}$$

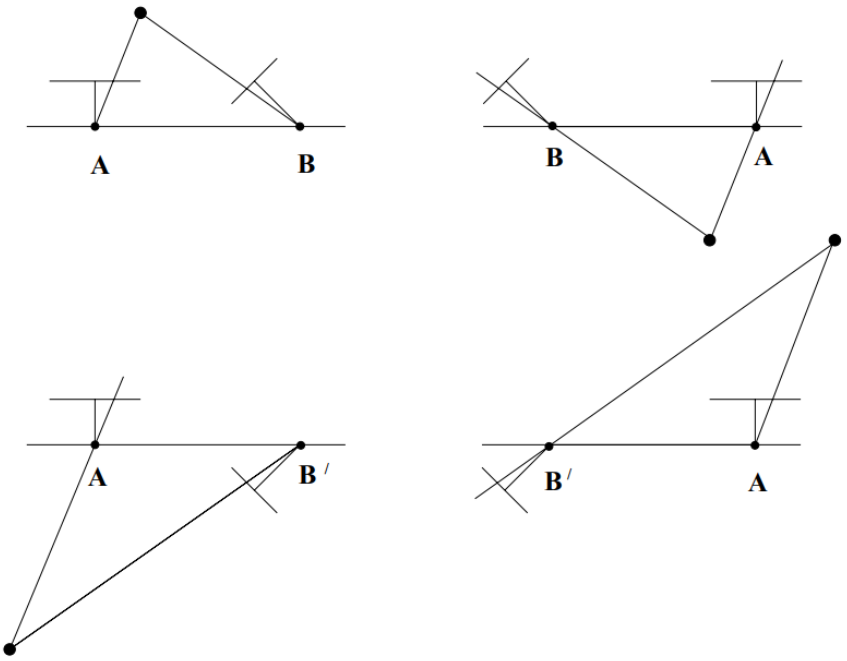
Vzhledem k nejednoznačnostem při určení vektoru $\mathbf{t}$ a matice $\mathbf{R}$ máme čtyři možnosti pro druhou normalizovanou matici kamery $\mathbf{P_2}$. Těmito možnostmi jsou

$$\mathbf{P'_{21}} = \begin{pmatrix}\mathbf{R_1} & \mathbf{t}\end{pmatrix}, \tag{5.33}$$

$$\mathbf{P'_{22}} = \begin{pmatrix}\mathbf{R_1} & -\mathbf{t}\end{pmatrix}, \tag{5.34}$$

$$\mathbf{P'_{23}} = \begin{pmatrix}\mathbf{R_2} & \mathbf{t}\end{pmatrix}, \tag{5.35}$$

$$\mathbf{P'_{24}} = \begin{pmatrix}\mathbf{R_2} & -\mathbf{t}\end{pmatrix}. \tag{5.36}$$



Obrázek 5.3: Čtyři řešení pro druhou matici kamery. Pouze řešení vlevo nahoře je správné, neboť umisťuje body scény před obě kamery [14].

Volba vhodné matice kamery se provádí podmínkou chirality [14]. Ta říká, že body scény se musejí nacházet před kamerou. Jedná se o triviální vlastnost, která působí zřejmě, ale v praxi se z důvodu chyb v datech často část bodů objeví za kamerou. Aby bylo možné tuto podmínku ověřit, je nutné zrekonstruovat pro každou možnost několik bodů. Pro další rekonstrukci vybíráme matici s největším počtem bodů před kamerou. Body, které podmínku chirality pro vybranou matici nesplňují, je vhodné vyfiltrovat. Tuto podmínku ilustruje obrázek 5.3.

Matice kamer se poté vypočítají podle vztahů

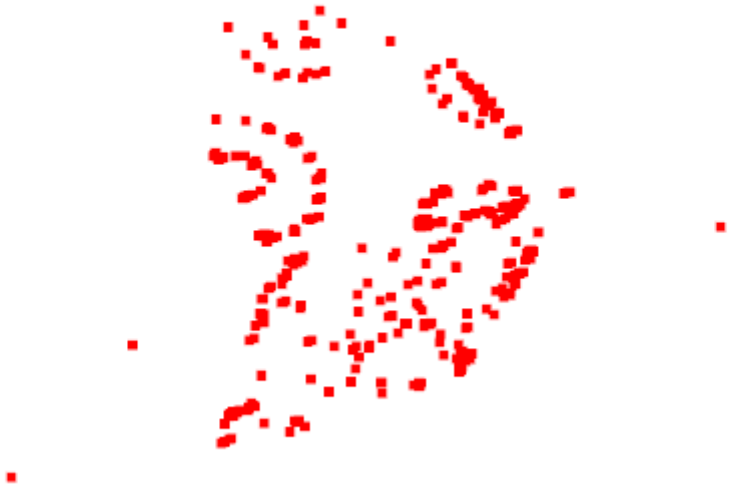
$$\mathbf{P_1} = \mathbf{K}\mathbf{P}'_\mathbf{1}, \tag{5.37}$$
$$\mathbf{P_2} = \mathbf{K}\mathbf{P}'_\mathbf{2}. \tag{5.38}$$

Jakmile známe obě matice kamer $\mathbf{P_1}$ a $\mathbf{P_2}$, můžeme provést rekonstrukci scény lineární triangulací. Z kapitoly 3 víme, že platí vztahy

$$\mathbf{X_1}' = \mathbf{P_1}\mathbf{X}, \tag{5.39}$$
$$\mathbf{X_2}' = \mathbf{P_2}\mathbf{X}. \tag{5.40}$$

Tyto vztahy lze také zapsat ve tvaru vektorového součinu

$$\mathbf{X_1}' \times (\mathbf{P_1}\mathbf{X}) = \mathbf{o}, \tag{5.41}$$
$$\mathbf{X_2}' \times (\mathbf{P_2}\mathbf{X}) = \mathbf{o} \tag{5.42}$$



Obrázek 5.4: Část krychle rekonstruované užitím esenciální matice.

s pomocí kterého získáme pro každou z kamer lineární soustavu rovnic

$$x_i(\mathbf{p_i^{3^\mathsf{T}}}\mathbf{X}) - (\mathbf{p_i^{1^\mathsf{T}}}\mathbf{X}) = 0, \tag{5.43}$$

$$y_i(\mathbf{p_i^{3^\mathsf{T}}}\mathbf{X}) - (\mathbf{p_i^{2^\mathsf{T}}}\mathbf{X}) = 0, \tag{5.44}$$

$$x_i(\mathbf{p_i^{2^\mathsf{T}}}\mathbf{X}) - y_i(\mathbf{p_i^{1^\mathsf{T}}}\mathbf{X}) = 0, \tag{5.45}$$

$$\tag{5.46}$$

ve které symbol $\mathbf{p_i^{j^\mathsf{T}}}$ označuje j-tý řádek i-té matice kamery. Z této soustavy dokážeme sestavit soustavu

$$\mathbf{AX} = \begin{pmatrix} x_1\mathbf{p_1^{3^\mathsf{T}}} - \mathbf{p_1^{1^\mathsf{T}}} \\ y_1\mathbf{p_1^{3^\mathsf{T}}} - \mathbf{p_1^{2^\mathsf{T}}} \\ x_2\mathbf{p_2^{3^\mathsf{T}}} - \mathbf{p_2^{1^\mathsf{T}}} \\ y_2\mathbf{p_2^{3^\mathsf{T}}} - \mathbf{p_2^{2^\mathsf{T}}} \end{pmatrix} \mathbf{X} = \mathbf{o}. \tag{5.47}$$

Tuto soustavu je možné řešit například SVD rozkladem. Řešením je zrekonstruovaný bod scény $\mathbf{X}$ [14].

Obrázek 5.4 zobrazuje výsledek rekonstrukce scény ze dvou snímků. Použité páry významných bodů tohoto objektu použité ve výše uvedeném algoritmu vidíme na obrázku 5.2. Všimněme si, že zrekonstruovaná scéna obsahuje body, které leží mimo hlavní zrekonstruovanou skupinu. Těmto bodům budeme říkat outliery a jejich odstraněním se zabývá část 6.1.

Pro rekonstrukci z většího počtu snímků existuje několik přístupů. Jeden z nich využívá trifokální tenzor pro přímou rekonstrukci ze tří pohledů, existují i tenzory pro rekonstrukci z většího počtu pohledů. Jiný se snaží rekonstruované scény spojovat dohromady detekcí významných prvků dané scény. Nejjednodušší způsob využívá vztahy (3.10) a (3.11), které popisují relativní polohu kamery v souřadnicovém systému jiné kamery.

Zmiňme, že scéna rekonstruovaná výše uvedeným postupem je poměrně řídké mračno bodů, neboť obsahuje pouze body, u kterých jsme dokázali detekcí zjistit korespondenci.

Existují další techniky, které umožňují zvýšit množství bodů této scény. Rovněž existují i metody, které převádějí ryze bodová data na polygonové sítě. Tyto metody však nejsou v kontextu této práce uvažovány.

5.4. Rekonstrukce v knihovně OpenCV

Závěr této kapitoly věnujeme opět předvedení metod knihovny OpenCV, které lze použít pro rekonstrukci scény.

Pro nalezení fundamentální matice je v knihovně dostupná funkce:

```
cv::Mat findFundamentalMat(InputArray points1, InputArray points2, int
    method, double ransacReprojThreshold, double confidence, int maxIters,
    OutputArray mask)
```

Parametry této metody mají následující význam.

`points1` – vstupní pole bodů prvního obrazu;

`points2` – vstupní pole bodů druhého obrazu, podotkneme, že musí mít odpovídající pořadí vzhledem k bodům z prvního obrazu;

`method` – celočíselná hodnota reprezentující požadovanou metodu;

`ransacRepojThreshold` – hodnota použitá pouze v metodách založených na RANSACu, jedná se o maximální vzdálenost bodu od epipolární přímky v pixelech, za kterou je již bod považován za outlier a nepodílí se na výpočtu fundamentální matice, autoři knihovnu doporučují volbu mezi $1,0$ až $3,0$;

`confidence` – spolehlivost použitá v metodě RANSAC, typicky $0,99$;

`maxIters` – maximální počet iterací robustních metod;

`mask` – výstupní maska bodů, které vyhovují nalezené fundamentální matici.

Pro hledání esenciální matice lze použít funkci:

```
cv::Mat findEssentialMat(InputArray points1, InputArray points2, InputArray
     cameraMatrix, int method, double confidence, double
    ransacReprojThreshold, int maxIters, OutputArray mask)
```

Parametry této metody jsou velmi podobné, jediným parametrem navíc je `cameraMatrix`, což je matice kamery.

Z esenciální matice je možné spočítat rotaci $\mathbf{R}$ a translaci $\mathbf{t}$ vzhledem k první kameře metodou:

```
int cv::recoverPose(InputArray essential, InputArray points1, InputArray
    points2, InputArray cameraMatrix, OutputArray R, OutputArray t)
```

Výstupem je počet bodů, které splňují podmínku chirality. Parametry jsou opět velmi podobné jako v předchozích metodách, lišícími se parametry jsou pouze

`essential` – esenciální matice;

`R` – výstupní matice rotace $\mathbf{R}$ vzhledem k první kameře;

`t` – výstupní vektor translace $\mathbf{t}$ vzhledem k první kameře.

Triangulaci nabízí knihovna OpenCV v podobě funkce:

```
void cv::triangulatePoints(InputArray projMatr1, InputArray projMatr2,
    InputArray projPoints1, InputArray projPoints2, OutputArray points4D)
```

Parametry této funkce jsou

`projMatr1` – projekční matice první kamery;

`projMatr1` – projekční matice druhé kamery;

`projPoints1` – vstupní vektor bodů prvního obrazu v homogenních souřadnicích;

`projPoints2` – vstupní vektor bodů druhého obrazu v homogenních souřadnicích;

`points4D` – výstupní matice triangulovaných bodů v homogenních souřadnicích.

6. Skládání prostorových dat

Doteď jsme se až na výjimky zabývali obrazovými daty. Vysvětlili jsme, jak je možné využít obrazová data pořízená několika kamerami k rekonstrukci reálné scény v podobě řídkého mračna bodů. V této kapitole se budeme výhradně zabývat prostorovými daty v podobě mračen bodů. Popíšeme metody, které lze použít pro čištění těchto dat od nežádoucích outlierů a které lze využít pro hledání geometrických transformací, které převádějí jedno mračno do souřadného systému jiného.

Tato kapitola je zpracována s pomocí zdrojů [75, 76, 77, 78, 80, 79].

6.1. Příprava prostorových dat

Mračna rekonstruovaná z obrazových dat mohou obsahovat body, které vznikly chybou a ve skutečnosti vůbec nejsou součástí původní scény. Takovým bodům říkáme outliery. Někdy také o datech, které obsahují vysokou míru outlierů hovoříme jako o zašumělých. Tato část popisuje několik technik, které umožňují odstranit chybná data z mračen bodů a předpřipravit je pro zpracování v dalších algoritmech.

Odstranění outlierů na základě poloměru Nejjednodušší metoda pro odstranění outlierů předpokládá, že tyto body mají ve svém okolí málo sousedních bodů. Vstupem tohoto čistícího algoritmu je poloměr koule r, kterou popisujeme okolí bodu a minimální počet bodů n, který v každém takovém okolí bodu požadujeme. Následný algoritmus je velmi jednoduchý. Pro každý bod vstupního mračna vytvoříme kouli o poloměru r, spočítáme počet bodů mračna v této kouli a bod umístíme do výstupního mračna pouze za předpokladu, že počet bodů v takto definovaném okolí je větší než n.

Ačkoli je algoritmus velmi jednoduchý, pro dostatečnou rychlost takového algoritmu je nutné využívat oktálových stromů, kterými lze snadno omezit počet bodů nutných pro procházení, jinak je složitost tohoto algoritmu kvadratická.

Nevýhodou tohoto algoritmu je jeho naivita, která spočívá v konstantním poloměru koule r. Algoritmus předpokládá, že mračno bodů je ve všech svých částech víceméně stejně husté. Pokud tento předpoklad není splněn, lze jen těžko zvolit vhodné parametry, které budou dostatečně filtrovat data a zároveň nebudou snižovat kvalitu výsledného mračna.

V knihovně Open3D je tento algoritmus implementován jako následující metoda:

```
std::shared_ptr<PointCloud> open3D::geometry::PointCloud::
    RemoveRadiusOutliers(size_t nb_points, double radius)
```

Parametry této metody jsou

`nb_points` – minimální požadovaný počet bodů n;

`radius` – poloměr koule r.

Metoda na výstupu vrací ukazatel na nové mračno s odstraněnými outliery.

Statistické odstranění outlierů K sofistikovanější metodě pro odstranění outlierů lze dojít s použitím základní statistiky. Místo toho, abychom měli poloměr koule popisující okolí všude konstantní, je možné pro každý bod vzít jako okolí n nejbližších bodů a spočítat jeho průměrnou vzdálenost od těchto bodů. Za outliery následně prohlásíme všechny body, které v rámci stanovené povolené směrodatné odchylky s mají vyšší průměrnou vzdálenost od svých sousedních bodů než body v jejich okolí.

Open3D tuto metodu nabízí následujícím způsobem:

```
std::shared_ptr<PointCloud> open3D::geometry::PointCloud::
    RemoveStatisticalOutliers(size_t nb_neighbors, double std_ratio)
```

Metoda má tyto argumenty

`nb_neighbors` – počet bodů n, které se zohlední pro výpočet průměrné vzdálenosti;

`std_ratio` – parametr umožňující kontrolovat poměr směrodatné ochylky.

Vzorkování mračna bodů V některých aplikacích je výhodné pracovat s celkově menším počtem bodů v mračnu. Může se jednat o situace, kdy velký počet bodů zpomaluje užitou metodu, nebo když pracujeme s dvěma mračny, z nichž jedno je již na vstupu řídké a druhé husté. V takových případech je výhodně umět počet bodů v mračnu redukovat. Tento proces se nazývá vzorkování (anglicky *down-sampling*).

Základním přístupem ke vzorkování je vzorkování stejnoměrné. Při tomto typu vzorkování z mračna jednoduše vezmeme každý k-tý bod. Takto předzpracované mračno tedy obsahuje k-krát méně bodů, než původní mračno. Vlastností výstupního mračna je, že si zachovává původní rozložení hustoty.

Knihovna Open3D má pro stejnoměrné vzorkování metodu:

```
std::shared_ptr<PointCloud> open3D::geometry::PointCloud::UniformDownSample
    (size_t every_k_points)
```

Jediným parametrem této metody je

`every_k_points` – číslo k.

Jiný přístup ke vzorkování je vzorkování voxelové. Voxel (z anglického *volumetric elements*) je trojrozměrnou analogií dvojrozměrného pixelu a představuje hodnotu v nějaké pravidelné mřížce trojrozměrného prostoru. Toto vzorkování rozdělí prostor, ve kterém se mračno bodů nachází na pravidelnou mřížku tvořenou krychlemi. V každé takové krychli nahradí všechny body jediným bodem, který umístí do jejich společného těžiště. Výsledné mračno má tak v celém svém objemu konstantní hustotu, což může být výhodné v některých aplikacích. Parametrem pro tento algoritmus je délka hrany a krychle, která tvoří jeden voxel.

Toto vzorkování lze v Open3D provést zavoláním:

```
std::shared_ptr<PointCloud> open3D::geometry::PointCloud::VoxelDownSample(
    double voxel_size)
```

Do metody vstupuje parametr

`voxel_size` – délka strany krychle a.

Odhad normál Data získaná rekonstrukčními metodami se skládají pouze s množiny bodů v prostoru. Pro mnoho algoritmů je však výhodné znát a dokázat popsat chování celé plochy, které tyto body určují. Dobrým výchozím stavem tak může být znalost normálových vektorů v každém bodě získaného mračna.

Jedna z metod použitelná pro odhad normál již byla naznačena v části 4.2, kde jsme rozšiřovali Harrisův detektor významných bodů pro trojrozměrná data. V této části je popsáno nalezení aproximační roviny v každém bodě mračna pomocí algoritmu PCA. Pro odhad lze použít normálu takto nalezené roviny. Článek [81] navrhuje robustnější metodu využívající k-d stromy pro efektivnější výpočet.

Pro odhad normál lze použít metodu knihovny Open3D:

```
void open3d::geometry::PointCloud::EstimateNormals()
```

Volba vhodné metody přípravy dat může rozhodnout o úspěšnosti dalších složitějších úprav, pro které chceme prostorová data použít. Tuto volbu navíc nelze zobecňovat a je nutné ji vhodně zvolit pro konkrétní aplikaci a vzorek dat.

6.2. Registrace mračen

Registrace je proces transformace několika různých množin dat do společného souřadného systému. Tento klasický problém z oblasti počítačového vidění je ústředním prvkem v mnoha směrech dnešní vědy. Nachází své uplatnění například v zobrazovacích metodách v lékařství, ve zpracování satelitních dat a v neposlední řadě také v astronomii pro spojování snímků oblohy. Většina metod používaných pro registraci dat staví na korespondenci bodů, která je popsána v části 4.6.

Je evidentní, že své aplikace tento proces bude mít i při práci s mračny bodů. Problém registrace dvou mračen bodů lze zjednodušeně charakterizovat následujícím způsobem. Mějme dvě mračna bodů, které alespoň částečně zobrazují stejnou scénu. Našim cílem je nalézt rotační matici $\mathbf{R}$ a translační vektor $\mathbf{t}$ tak, abychom minimalizovali vzdálenost mezi překrývajícími se částmi obou mračen. Takto lze problém formulovat jako optimalizační úlohu s obecně šesti stupni volnosti. Bohužel bez žádných apriori známých informací o vzájemné poloze obou mračen většina běžných řešitelů na tomto problému selhává. Je to proto, že účelová funkce je vícerozměrná a mívá mnoho lokálních extrémů, které jsou blízko globálnímu.

Tradičně metody pro registraci rozdělujeme do dvou kategorií – lokální a globální. Lokální metody registrace na vstupu berou transformační matici, která alespoň přibližně převádí jedno z mračen do souřadného systému toho druhého, a pouze danou transformaci vylepšují. Globální metody obvykle používáme pro inicializaci lokálních metod, neboť se na vstupu obejdou bez znalosti vzájemné polohy. V dalších částech budou podrobněji rozebrány oba typy registračních metod.

6.2.1. Globální registrace

Dnes nejznámější metody globální registrace jsou obvykle založeny na globální stochastické optimalizaci s využitím genetických nebo evolučních algoritmů. Jejich častou nevýhodou je celkový výpočetní čas. Zde se zaměříme na využití FPFH (*Fast Point Feature Histograms*) [80] deskriptorů v metodě postavené na RANSACu.

Mějme mračno bodů, v němž každý bod P má již známou normálu $\mathbf{n_P}$. V každém bodě tohoto mračna vezměme všechny body v okolí o poloměru r a pro každou dvojici různých bodů P a Q z tohoto okolí a jejich normály $\mathbf{n_P}$ a $\mathbf{n_Q}$ definujme

$$\mathbf{u} = \mathbf{n_P}, \tag{6.1}$$
$$\mathbf{v} = (Q - P) \times \mathbf{u}, \tag{6.2}$$
$$\mathbf{w} = \mathbf{u} \times \mathbf{v}. \tag{6.3}$$

Trojici těchto vektorů se říká Darbouxův rám. Jako bod P ve dvojici vždy volíme bod, který má menší úhel mezi normálou a vektorem, který tyto dva body spojuje. Významné rysy těchto bodů nyní spočítáme ze vztahů

$$f_1 = \mathbf{v}\mathbf{n_Q}, \tag{6.4}$$
$$f_2 = \frac{\mathbf{u}(Q - P)}{\|Q - P\|}, \tag{6.5}$$
$$f_3 = \arctan_2(\mathbf{w}\mathbf{n_Q}, \mathbf{u}\mathbf{n_Q}). \tag{6.6}$$

Dále definujme indexovací funkci

$$j(P) = \sum_{i=1}^{3} \text{step}(s_i, f_i(P))2^{i-1}, \tag{6.7}$$

kde funkce $\text{step}(s, f)$ je 0, pokud $f < s$, jinak 1. A $f_i(P)$ je hodnota rysu f_i pro bod P. Ke každému bodu lze nyní přiřadit histogram obsahující procentuální zastoupení jeho okolních bodů ve stanovených intervalech.

Pro porovnávání těchto deskriptorů autoři doporučují používat Kullback-Leiblerovu divergenci [82], která popisuje jak se odlišují dvě rozdělení pravděpodobnosti.

Podotkneme, že autoři deskriptoru dříve používali ještě rys popisující vzdálenost dvou bodů okolí, který se ale později ukázal jako nepříliš významný, a tak jejich histogram původně obsahoval 16 tříd. Jsou také doporučeny dodatečné úpravy, které snižují výpočetní náročnost a zabraňují „skluzovému“ jevu, který způsobují chybně zvolené významné body na některých površích. Tyto body negativně ovlivňují hodnotu účelové funkce, protože se špatně párují.

Pro globální registraci užitím výše popsaných deskriptorů je doporučena následující metoda [80]:

1. zvolíme náhodně n bodů prvního mračna, přičemž zajistíme, aby jejich vzájemná vzdálenost byla větší než nějak zvolené $d_{\min}$;

2. pro každý zvolený bod prvního mračna nalezneme seznam bodů druhého mračna, jejichž deskriptory jsou podobné, z těch zvolíme jeden bod náhodně, který prohlásíme za jeho pár;

3. spočítáme eukleidovskou transformaci definovanou těmito páry a zhodnotíme kvalitu registrace;

4. opakujeme, dokud nezískáme dostatečně kvalitní transformaci.

6.2.2. Lokální registrace

Nejpopulárnější metodou lokální registrace je bezpochyby ICP (*Iterative Closest Point*). Klíčovou myšlenkou této metody je minimalizace rozdílu mezi oběma mračny za předpokladu, že nejbližší body si odpovídají. Rozdíl popisuje funkce

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} \|P_i - \mathbf{R}Q_i - \mathbf{t}\|^2 , \tag{6.8}$$

kde n je menší z počtu bodů v obou mračnech a body P_i a Q_i jsou po řadě body prvního a druhého mračna, jejichž vzdálenost je minimální.

Kroky algoritmu jsou následující:

1. pro každý bod prvního mračna stanovíme nejbližší bod druhého mračna;
2. odhadneme eukleidovskou transformaci, která minimalizuje chybu ve vztahu (6.8);
3. transformujeme všechny body prvního mračna nalezenou transformací;
4. opakujeme, dokud nezískáme dostatečně kvalitní transformaci.

Dodejme, že existují další přidružené varianty algoritmu ICP. Tyto obvykle mírně mění strategii párování bodů, přidávají váhy bodům nebo eliminují nevhodné páry bodů.

6.3. Obarvení mračna

Hlavním cílem naší práce je obarvit předem dané mračno bodů sadou barevných obrazů, které zobrazují stejnou scénu jako dané mračno. Metoda, kterou jsme pro tento úkol zvolili spočívá v rekonstrukci nového mračna užitím epipolární geometrie (kapitola 5) a následném nalezení geometrických transformací, které nové mračno správně složí dohromady s původním mračnem. K tomuto využijeme metod registrace. Díky tomu známe transformace nutné k promítnutí libovolného bodu daného mračna do obrazu libovolné kamery.

Posledním problémem, který je potřeba před obarvením vyřešit je, který snímek použijeme pro obarvení kterého bodu mračna. Pro obarvovaný bod jednak musí platit, že se nachází na daném snímku – tedy že dvojrozměrný bod, který vznikne promítnutím prostorového bodu na daný snímek neleží mimo oblast daného snímku a také, že je daný bod na daném, snímku viditelný. Viditelností v mračnu bodů se zabývá například článek [83].

Zde navrhujeme použití heuristické metody. Každý bod obarvíme pomocí snímku, z nějž zrekonstruované body jsou nejblíž obarvovanému bodu a zároveň daný snímek bod obsahuje.

7. Implementace navržené metody

Jedním z cílů této práce bylo implementovat metodu pro mapování textury získané zpracováním běžných fotografií na mračno bodů. Teoretická témata předložená v předchozích kapitolách budovala postupy, jež jsou základem navržené metody. V dalších částech nejdřív popíšeme navrženou metodu i její implementaci a následně předvedeme výsledky této metody na testovacích datech.

7.1. Navržená metoda

Před úplným představením navržené metody zopakujme zadání problému. Máme dánu sérii barevných obrazů i_1 až i_n, kalibrační matici použité kamery $\mathbf{K}$ a neobarvené mračno bodů G. Našim cílem je nalézt vhodné geometrické transformace, které body mračna zobrazí na jim odpovídající obrazové body na jednotlivých obrazech. Dále chceme obarvit mračno s využitím těchto transformací.

Metoda navržená pro řešení tohoto problému má tři kroky. Prvním krokem je rekonstrukce nového mračna G' z pořízených obrazů. Druhým krokem je pak nalezení transformací, které rekonstruované mračno G' převádějí na původní mračno G. Třetí krok se týká využití nalezených zobrazení pro obarvení mračna G.

Rekonstrukce Během rekonstrukce se obrazová data i_1 až i_n rekonstruují do 3D s využitím znalostí z epipolární geometrie, zejména pak znalostí o fundamentální matici. Fundamentální a esenciální matice je vypočtena užitím sedmibodové RANSAC metody. Vstupem do této metody jsou páry významných bodů získané detektorem ORB a párované hrubou silou s využitím Hammingovy metriky. Algoritmus ORB byl zvolen kvůli své výkonnosti a robustnosti. Párování hrubou silou bylo vybráno pro svou jednoduchost, ačkoli by bylo možné použít i jiné metody. Rovněž je potřeba poskytnout matici vnitřních parametrů kamer $\mathbf{K}$. Výstupem rekonstrukce je řídké mračno bodů G' a projekční matice $\mathbf{P_1}$ až $\mathbf{P_n}$ nutné pro promítnutí tohoto mračna na jednotlivé obrazy i_1 až i_n.

Sesazení Druhý krok používá globální registraci s FPFH deskriptory pro nalezení vhodné transformace mezi řídkým mračnem G' a hustým vstupním mračnem G. Pro FPFH deskriptory jsme se rozhodli kvůli tomu, že se jedná o moderní a rychlý přístup řešení problému registrace. Výslednou transformaci je možné zlepšit algoritmem ICP. Výstupem je transformační matice $\mathbf{H}$, která tuto transformaci popisuje.

Obarvení Pro obarvení nejdříve původní mračno G transformujeme maticí $\mathbf{H}^{-1}$ a následně jednotlivé body promítneme na poskytnuté snímky i_1 až i_n užitím projekčních matic $\mathbf{P_1}$ až $\mathbf{P_n}$. Bod mračna je obarven podle barvy pixelu, na který se promítne, pokud vyhovuje heuristice navržené v části 6.3.

Navržený algoritmus je implementován v jazyce C++, který je dnes stále dominantní v počítačovém vidění. V implementaci jsou použity vybrané metody knihoven OpenCV [7] a Open3D [8]. Tyto knihovny jsou standardem pro práci s obrazovými a prostorovými daty. Pro kalibraci a parametrizaci jsou dodány skripty naprogramované v Pythonu využívající stejné knihovny.

Při testování se osvědčilo skládat během rekonstrukce model vždy pouze ze dvou snímků a ten následně poslat do dalších kroků. Aplikace je proto naprogramovaná tak, aby fotky brala po dvojicích.

Implementace je poskytnuta ve zkompilované podobě pro 64bitový systém Windows a také v podobě zdrojového kódu, který pro sestavení využívá program CMake. Další popis se bude týkat pouze 64bitového systému Windows. Spuštění na jiných systémech vyžaduje kompilaci. Přesné použití knihoven OpenCV a Open3D pro sestavení je pak možné najít v [48, 75].

Pro spuštění aplikace je nutné mít všechny potřebné knihovny v podobě `.dll` souborů umístěné v adresáři s aplikací. Knihovny je z důvodu omezení velikosti příloh nutné stáhnout zvlášť z oficiálního zdroje. Potřebné knihovny jsou OpenCV (v. 3.4.3) a Open3D (v. 0.15.1).

Knihovnu OpenCV v požadované verzi je možné stáhnout z odkazu:
`https://github.com/opencv/opencv/releases/tag/3.4.3`
Stažený soubor `opencv-3.4.3-vc14_vc15.exe` je nutné rozbalit do libovolného adresáře. V tomto adresáři lze poté nalézt soubory:
`opencv/build/x64/vc15/bin/opencv_world343.dll`,
`opencv/build/x64/vc15/bin/opencv_world343d.dll`,
které je potřeba překopírovat do adresáře s poskytnutou zkompilovanou verzí programu.

Knihovnu Open3D v požadované verzi je možné stáhnout z odkazu:
`https://github.com/isl-org/Open3D/releases/tag/v0.15.1`
Stažený soubor `open3d-devel-windows-amd64-0.15.1.zip` je opět nutné rozbalit do libovolného adresáře. V tomto adresáři lze poté nalézt soubor:
`bin/Open3D.dll`,
který je potřeba překopírovat do adresáře se zkompilovanou verzí programu.

Výsledná adresářová struktura aplikace by měla vypadat přibližně takto:

```
root\
 | res\
    | test\
    | vase\
    | tea\
 | program.exe
 | Open3D.dll
 | opencv_world343.dll
 | opencv_world343d.dll
 | configuration.py
 | calibration.py
```

Dále budeme předpokládat, že se nacházíme v adresáři `root` a všechny příkazy budeme zadávat z tohoto adresáře. Takto nachystaný program je připravený k použití.

Mapovací program se používá zavoláním z příkazové řádky. Příkaz spouštějící obarvení je

```
program.exe <project_folder>
```

Povinným argumentem `<project_folder>` je cesta k složce obsahující všechny soubory projektu.

Složka projektu musí obsahovat soubor `project.txt`. Tento soubor poskytuje programu konfigurační informace. Konfigurační soubor se skládá z bloků, které jsou uvozeny názvem bloku. Ten musí být vyplněn velkými písmeny. Konfigurační soubor také umožňuje

používat komentáře. Ty začínají znakem # a jsou při načítání konfigurace automaticky ignorovány. Dále budou popsány jednotlivé konfigurační bloky.

IMAGE V tomto bloku se na samostatné řádky uvádí názvy jednotlivých obrazových souborů. Alespoň 2 snímky musí být uvedeny. Standardní formáty obrázků `.png` a `.jpg` jsou podporovány. Kromě jména souboru by na řádku neměl být žádný další znak. Příkladem bloku je:

```
IMAGE
01.png
02.png
03.png
04.png
```

POINT_CLOUD Tento blok poskytuje aplikaci název zdrojového mračna bodů. Mračno bodů musí být uloženo ve formátu `.ply`. Příkladem budiž:

```
POINT_CLOUD
target.ply
```

CAMERA V tomto bloku se definuje kalibrační matice kamery. Matice musí obsahovat pouze čísla a mezi jejími prkvy musí být pouze jedna mezera. Příkladem definice kamery je:

```
CAMERA
2667 0 960
0 2667 540
0 0 1
```

RECONSTRUCTION_PARAMS Tento blok umožňuje nastavení parametrů pro rekonstrukční krok programu. I zde je potřeba zapsat čísla oddělená jedinou mezerou. Tyto čísla po řadě reprezentují: počet významných bodů nalezených detektorem, maximální povolenou vzdálenost bodů ve výpočtu fundamentální matice, požadovanou spolehlivost RANSAC metody pro výpočet fundamentální i esenciální matice, maximální povolenou vzdálenost při filtraci outlierů během výpočtu matic kamer a maximální povolenou chyba při reprojekci pro jeden bod. Použití je následující:

```
RECONSTRUCTION_PARAMS
2000 1 0.99 1000 10
```

REGISTRATION_PARAMS Tento blok definuje nastavení registračního kroku algoritmu. Opět je potřeba zapsat čísla oddělená jedinou mezerou. Čísla po řadě reprezentují: velikost voxelu při down-samplingu rekonstruovaného mračna, velikost voxelu při down-samplingu původního mračna, počet vstupních bodů RANSAC metody při registraci, maximální počet iterací RANSAC metody při registraci, maximální vzdálenost odpovídajících si bodů při sesazování, požadovanou spolehlivost RANSAC metody, poloměr pro odhad normál, počet bodů použitý pro odhad normál, poloměr pro výpočet deskriptorů, počet bodů pro výpočet deskriptorů, minimální počet bodů při filtrování na základě

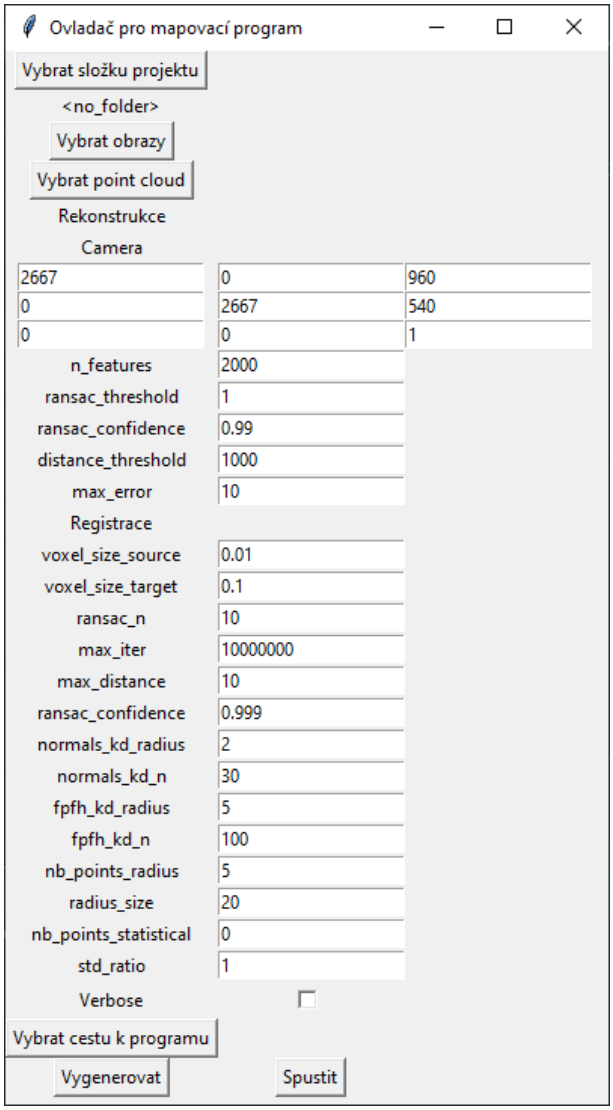
poloměru (lze zvolit 0 nebo méně, a tím zrušit filtrování), poloměr pro filtrování na základě poloměru, počet bodů pro statistické filtrování (lze zvolit 0 nebo méně, a tím zrušit filtrování) a směrodatnou odchylku pro statistické filtrování. Používá se způsobem uvedeným dále:

```
REGISTRATION_PARAMS
0.01 0.1 10 10000000 10 0.999 2 30 5 100 5 20 0 1
```

VERBOSE Tento blok je volitelný a slouží pouze jako instrukce pro generování více výstupů programu. Toho lze využít při ladění programu a hledání vhodných parametrů. Aplikace s tímto výstupem bude produkovat více dat do konzole a poskytovat obrazové mezivýsledky. Použití tohoto bloku je:

```
VERBOSE
```

V ladícím režimu se zobrazují i průběžné výsledky. Ty je možné přeskočit stisknutím klávesy `Esc`.



Obrázek 7.1: Konfigurační program.

Stanovená hranice h	Pokus 1	Pokus 2	Pokus 3	Průměr
$0,01$	$0,0119336$ %	$0,0159115$ %	$0,0318231$ %	$0,0198894$ %
$0,1$	$27,3479$ %	$21,9778$ %	$28,6964$ %	$26,0073666667$ %
$0,5$	$80,4447$ %	$81,9484$ %	$82,4575$ %	81.6168666667 %

Tabulka 7.1: Kvalita barev ve vzorku „test“.

Vzhledem ke složitosti těchto souborů je poskytnut také skript naprogramovaný v jazyce Python, který umí tyto soubory generovat. Ten se spouští příkazem:

```
python configurator.py
```

Pro jeho použití je potřeba mít nainstalován interpretr jazyka Python alespoň verze 3.10.0 [84]. Nastavení se provádí v klasické okenní aplikaci, která je zobrazena na obrázku 7.1. Nejdříve je potřeba zvolit složku projektu. Následně je možné zvolit, které snímky chceme použít a s jakým mračnem bodů chceme pracovat. Další parametry je možno ponechat ve výchozím stavu, případně je změnit. Stisknutím tlačítka „Vygenerovat“ se vytvoří v projektové složce soubor `project.txt`, který je možné použít pro spuštění hlavního programu. Je-li zvolena i cesta k mapovacímu programu, pak je možné stisknutím tlačítka „Spustit“ spustit mapování. V takovém případě není potřeba manuálně soubor generovat. Součástí spouštění je i generace.

7.2. Výsledky metody

V této části prezentujeme výsledky navržené metody. Předem podotkneme, že úspěšnost této metody je silně závislá na volbě vstupních parametrů, kvalitě poskytnutých dat i náhodě, která vystupuje v metodách RANSAC.

Algoritmus byl testován na třech datových vzorcích (*datasetech*). Dva vzorky jsou umělé a jeden pochází z reálných dat získaných 3D skenerem ATOS. Dále blíže popíšeme výsledky testování jednotlivých datových vzorků.

Vzorek „test“ Tato data byla speciálně vytvořena pro testování kvality navržené metody. Jedná se o mírně deformovanou krychli vytvořenou v programu Blender, na kterou byla nanesena barevná textura. Tento přístup nám umožnil vygenerovat mračno, které je již v původní podobě správně obarveno. Náš program tak může po svém doběhnutí posoudit kvalitu přiřazení barev.

Tento objekt byl následně programem Blender vyrenderován do 36 snímků, které jsou pořízeny kamerou orbitující kolem tělesa. Kamera používá perspektivní model s fokální vzdáleností 50mm a velikostí senzoru 36mm × 20,25mm. Vzhledem k velikosti snímků 1920px × 1080px byla stanovena matice vnitřních parametrů kamery

$$\mathbf{K} = \begin{pmatrix} 2667 & 0 & 960 \\ 0 & 2667 & 540 \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.1}$$

Kvalitu přiřazení barev zkoumáme jednoduchým testem. Pro všechny body provedeme rozdíl vektorů popisujících jejich barvu a spočítáme jeho normu. Následně počítáme, kolik procent bodů má velikost rozdílu pod nějakou stanovenou hranicí h. Vzhledem k ná-

Dateset	Počet snímků	Počet bodů mračna	Průměrný výpočetní čas
test	36	25139	4,35s
vase	100	295101	42,968s
tea	33	121389	10,551s

Tabulka 7.2: Přehled průměrných výpočetních časů mapování jednotlivých datasetů. Vypočteno jako průměr třech nezávislých pokusů.

hodnosti použitých metod je pokus opakován vícekrát, přičemž nastavení je ponecháno ve výchozím stavu. Kvalitu stanovenou pro různé hodnoty h je možné vidět v tabulce 7.1.

Výsledek je rovněž možné pozorovat na obrázku 7.2. Konkrétně obrázek 7.2c znázorňuje, které části objektu byly namapovány správně. Je vidět, že některým částem objektu byly nesprávně přiřazeny barvy pozadí. Filtrování těchto přiřazení by mohlo významně zlepšit kvalitu algoritmu.

Vzorek „vase“ Vzorek „vase“ má pohledově asi nejlepší výsledky. Tento vzorek pochází z velkého datasetu IVL-SYNTHSFM-v2 [85, 86, 87], který slouží pro testování metod *structure-from-motion*. Jedná se o renderované snímky různých 3D objektů, které jsou vytvořeny s různými světelnými podmínkami. Pro každý model tohoto datasetu a každé z 8 nastavení světelných podmínek je k dispozici 100 snímků.

V této práci používáme data „EmpireVase“ s fixní polohou slunce, které dataset označuje názvem „fs“. Vzhledem k velikosti souborů, není tento dataset součástí přílohy, ale je jej možné zdarma stáhnout na webu [85].

Snímky jsou pořízeny kamerou s ohniskovou vzdáleností 35mm a senzorem o velikosti 32mm × 18mm. Rozlišení jednotlivých obrazů je 1920px × 1080px. Stanovená kalibrační matice tak je

$$\mathbf{K} = \begin{pmatrix} 2100 & 0 & 960 \\ 0 & 2100 & 540 \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.2}$$

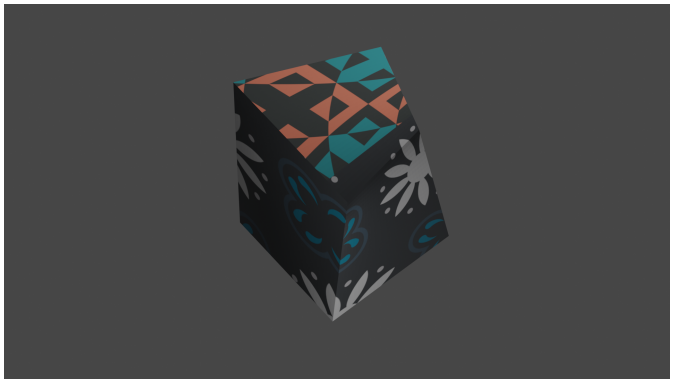
V tabulce 7.2 je vidět srovnání délky výpočetního času nutného pro zpracování všech použitých datasetů. Vzorek „vase“ obsahuje 100 snímků a výpočetní čas se pro něj pohybuje okolo 43 sekund. Obarvený model je možné vidět na obrázku 7.3.

Vzorek „tea“ Posledním testovaným vzorkem jsou reálná data získaná skenováním na 3D skeneru ATOS. Bohužel pořízení snímků bylo provedeno mobilním telefonem a nedošlo k pořízení vhodných kalibračních snímků. Kalibrační matice tak musela být odhadnuta z informací poskytnutými výrobcem. Snímky rovněž musely být zmenšeny, aby mohly být použity jako příloha této práce.

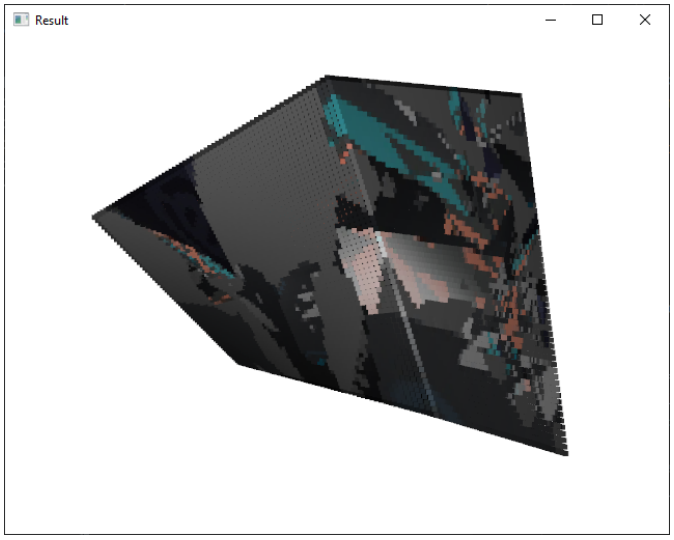
Odhadnutá matice vnitřních parametrů kamery tak byla stanovena na

$$\mathbf{K} = \begin{pmatrix} 864 & 0 & 576 \\ 0 & 864 & 432 \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.3}$$

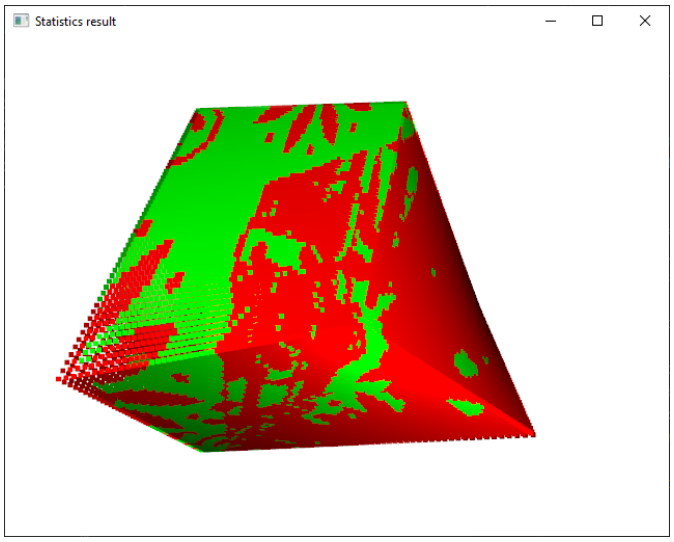
Lepších výsledků by však mohlo být dosaženo kvalitní kalibrací a použitím metod pro odstranění zkreslení.



(a) Příklad vstupního snímku.



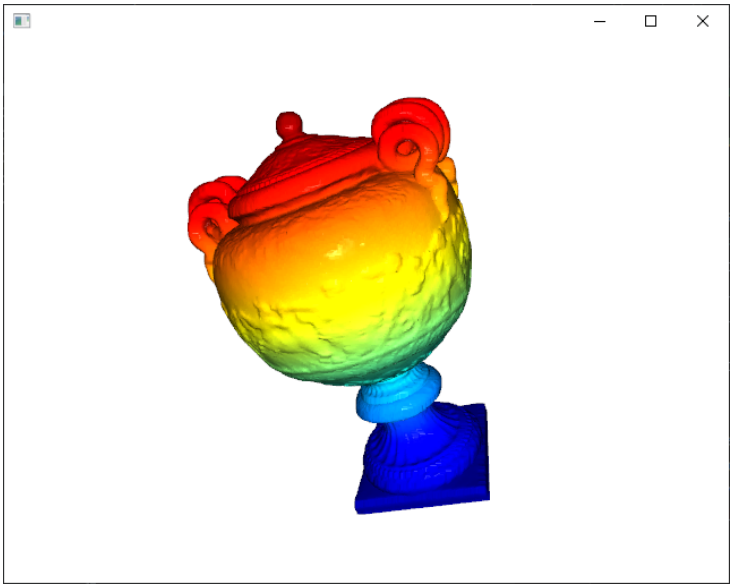
(b) Snímek obarveného modelu.



(c) Mapa kvality přiřazení barev s hranicí $h = 0.1$. Zelená místa byla mapována správně, červená špatně.

Obrázek 7.2: Vzorek „test“.

Obrázek 7.4 zobrazuje výsledky párování významných bodů před a po filtrování s využitím fundamentální matice. Je patrné, že páry jsou utvořeny správným způsobem a že se nacházejí na zkoumaném objektu.



(a) Neobarvený model.

(b) Příklad vstupního snímku.

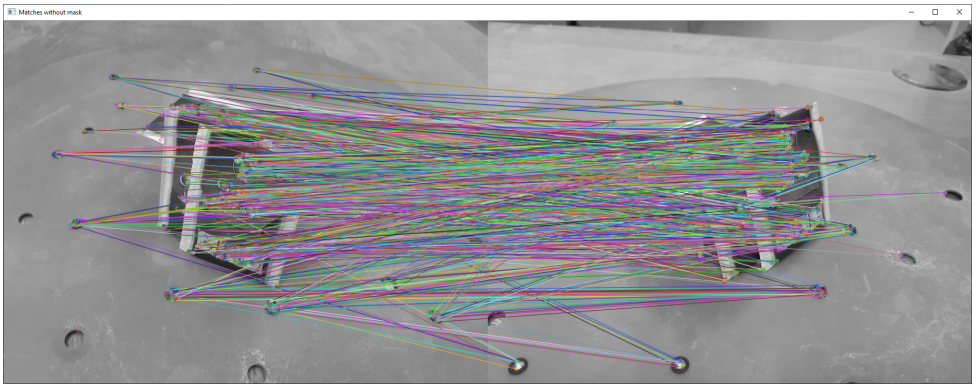
(c) Snímek obarveného modelu.

(d) Příklad vstupního snímku.

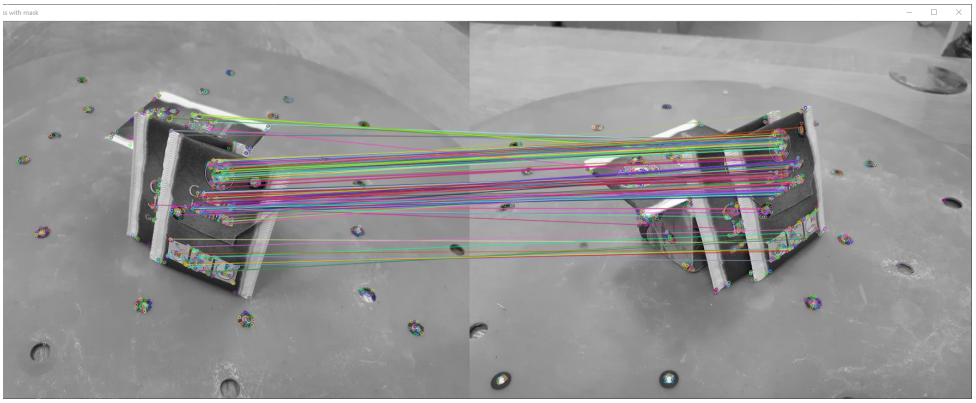
(e) Snímek obarveného modelu.

Obrázek 7.3: Vzorek „vase“.

Na obrázku 7.5 můžeme vidět zpětně promítnuté rekonstruované body objektu. Zpětné promítnutí v algoritmu používáme pro odstranění špatně rekonstruovaných bodů. Abychom bod prohlásili za správně rekonstruovaný, musí jeho vzdálenost od původního bodu být menší než stanovená povolená chyba.

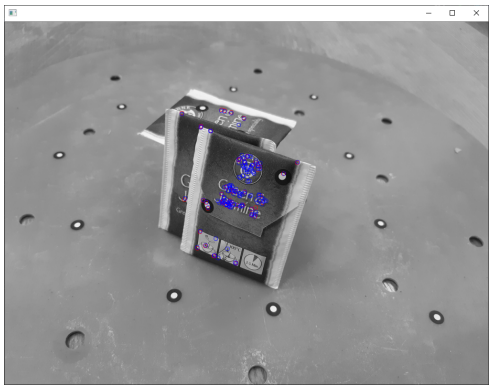


(a) Páry významných bodů algoritmu ORB, které jsou spárovány hrubou silou. Všimneme si, že tyto páry obsahují řadu chyb.



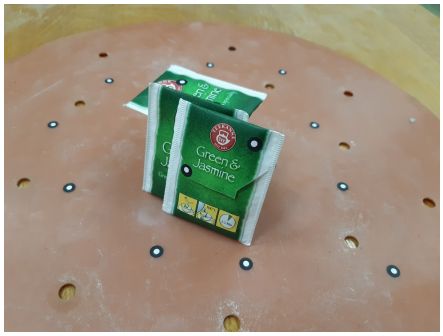
(b) Zbylé páry významných bodů po filtrování s využitím fundamentální matice. Většina chybných párů je odstraněna, protože nevyhovují nalezené fundamentální matici.

Obrázek 7.4: Párování významných bodů v datovém vzorku „tea“.

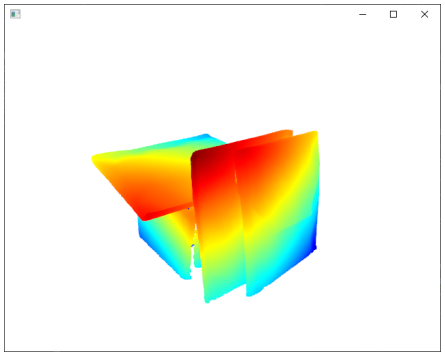


Obrázek 7.5: Reprojekce rekonstruovaných významných bodů. Modře jsou znázorněny všechny použité významné body. Červeně jsou znázorněny zpátky promítnuté body. Čím víc se jich překrývá s modrými body, tím lépe byly zjištěny projekční matice.

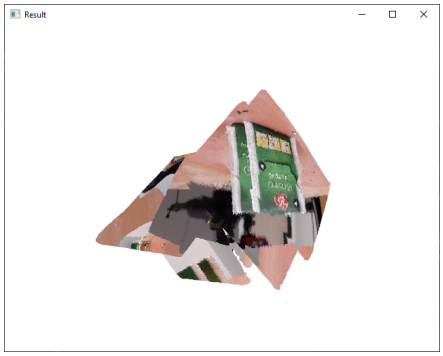
Obrázek 7.6 porovnává různé výsledky mapování textury na vstupní mračno. Nastavení pro oba výstupy bylo ponecháno stejné. Evidentně je výsledek velmi závislý na náhodě. Výsledek by mohlo být možné zlepšit nalezením optimálnějších parametrů. Použité parametry jsou obsahem projektového souboru v příloze.



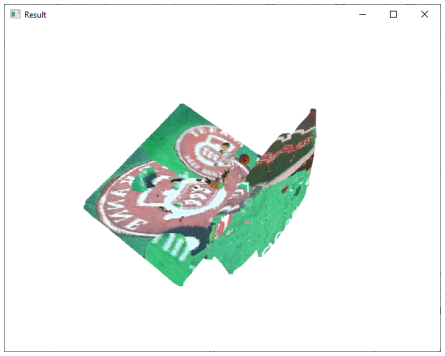
(a) Příklad vstupního snímku.



(b) Vstupní model.



(c) Špatný výsledek mapování textury.



(d) Lepší výsledek mapování textury.

Obrázek 7.6: Výsledky mapování textury v datasetu „tea“.

Z porovnání jednotlivých výsledků vyplývá, že nejslabším místem navržené metody je metoda globální registrace. V dalším zkoumání by bylo vhodné zaměřit se na nalezení optimální metody registrace pro řídká mračna bodů.

Závěr

Cílem této práce bylo nastudovat a popsat metody pro hledání korelací mezi mračnem bodů a fotografiemi. Dalším cílem bylo použít nastudované metody pro návrh vlastního algoritmu sloužícího k automatickému mapování textury na mračno bodů.

Na začátku práce byly zavedeny projektivní prostory a předvedeny základní geometrické transformace, které v těchto prostorech provádíme. Popsali jsme, jak lze v počítači reprezentovat obrazy a mračna bodů. Jedna z kapitol byla věnována detekci a párování významných bodů, což je klíčový proces v mnoha aplikacích počítačového vidění. Poté byl popsán model perspektivní kamery, který byl následně využit pro jako základ epipolární geometrie. V rámci popisu epipolární geometrie jsme ukázali, jak je možné použít fundamentální matice k rekonstrukci scény. Poslední část se věnovala sesazení dvou mračen bodů užitím globální a lokální registrace.

Hlavním přínosem této práce je vytvoření přehledu a podrobného zpracování řady moderních metod počítačového vidění. Některé z těchto metod jsou následně využity v návrhu vlastního algoritmu pro automatické texturování. Součástí práce je rovněž implementace tohoto algoritmu a jeho testování na vybraných umělých i reálných datech. Výsledky navržené metody byly hodnoceny z hlediska kvality a výpočetního času.

Tato práce slouží rovněž jako dobrý výchozí bod dalšího zkoumání. Je možné na ni navázat například hledáním optimálních parametrů představené metody, testováním jiných přístupů pro globální registraci nebo návrhem algoritmu pro automatické posouzení kvality mapování. Kvalita mapování by v navazujícím výzkumu mohla být porovnána s jinými existujícími metodami.

Literatura

[1] JACOBS, David W. Matching 3-D Models to 2-D Images. *International Journal of Computer Vision* [online]. **21**(1/2), 123-153 [cit. 2022-05-18]. ISSN 09205691. Dostupné z: doi:10.1023/A:1007927623619

[2] STAMOS, I. a P.K. ALIEN. Automatic registration of 2-D with 3-D imagery in urban environments. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* [online]. IEEE Comput. Soc, 2001, 731-736 [cit. 2022-05-18]. ISBN 0-7695-1143-0. Dostupné z: doi:10.1109/ICCV.2001.937699

[3] GEHUA YANG, Gehua Yang, Jacob BECKER a Charles V. STEWART. Estimating the Location of a Camera with Respect to a 3D Model. *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)* [online]. IEEE, 2007, 2007, 159-166 [cit. 2022-05-18]. ISBN 0-7695-2939-9. ISSN 1550-6185. Dostupné z: doi:10.1109/3DIM.2007.23

[4] SCHINDLER, Grant, Panchapagesan KRISHNAMURTHY, Roberto LUBLINERMAN, YANXI LIU a Frank DELLAERT. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. *2008 IEEE Conference on Computer Vision and Pattern Recognition* [online]. IEEE, 2008, 2008, 1-7 [cit. 2022-05-18]. ISBN 978-1-4244-2242-5. Dostupné z: doi:10.1109/CVPR.2008.4587461

[5] STAMOS, Ioannis. Automated registration of 3D-range with 2D-color images: an overview. *2010 44th Annual Conference on Information Sciences and Systems (CISS)* [online]. IEEE, 2010, 2010, 1-6 [cit. 2022-05-18]. ISBN 978-1-4244-7416-5. Dostupné z: doi:10.1109/CISS.2010.5464815

[6] ZHAO, W., D. NISTER a S. HSU. Alignment of continuous video onto 3D point clouds. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004* [online]. IEEE, 2004, 2004, II-II [cit. 2022-05-18]. ISBN 0-7695-2158-4. Dostupné z: doi:10.1109/CVPR.2004.1315269

[7] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* [online]. 2000 [cit. 2022-05-18]. Dostupné z: `https://opencv.org/`

[8] ZHOU, Qian-Yi, Jaesik PARK a Vladlen KOLTUN. *Open3D: A Modern Library for 3D Data Processing* [online]. 2018 [cit. 2022-05-18]. Dostupné z: `http://www.open3d.org/`

[9] EUCLID, FITZPATRICK, Richard, ed. *Euclid's Elements of Geometry*. 2008. ISBN 978-0-6151-7984-1.

[10] HILBERT, David. *Grundlagen der Geometrie*. Leipzig: Druck und Verlag von B. G. Teubner, 1909, 279 s.

[11] BEČVÁŘ, Jindřich. *Lineární algebra*. Vyd. 3. Praha: Matfyzpress, 2005. ISBN 80-86732-57-6.

[12] MARTIŠEK, Dalibor. *Počítačová geometrie a grafika (přednáška): 02 – Euklidovská geometrie* [online]. [cit. 2022-05-15]. Dostupné z: `https://mathonline.fme.vutbr.cz/Prednaska/sc-1245-sr-1-a-261/default.aspx`

[13] MARTIŠEK, Dalibor. *Počítačová geometrie a grafika (přednáška): 03 – Projektivní geometrie* [online]. [cit. 2022-05-15]. Dostupné z: `https://mathonline.fme.vutbr.cz/Prednaska/sc-1245-sr-1-a-261/default.aspx`

[14] HARTLEY, Richard a Andrew ZISSERMAN. *Multiple view geometry in computer vision*. 2nd ed. Cambridge: Cambridge University Press, 2003, 655 s. ISBN 0-521-54051-8.

[15] MRKVIČKA, Daniel. *Rekonstrukce 3D objektů z více pohledů*. Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2019.

[16] PAVLÍČEK, Jan B. *Základy neeuklidovské geometrie Lobačevského*. Praha: Přírodovědecké nakladatelství, 1953. Dostupné také z: `https://dml.cz/handle/10338.dmlcz/402750`

[17] KALMAN, Dan. A Singularly Valuable Decomposition: The SVD of a Matrix. *The College Mathematics Journal* [online]. 2018, **27**(1), 2-23 [cit. 2022-05-19]. ISSN 0746-8342. Dostupné z: doi:10.1080/07468342.1996.11973744

[18] ČERMÁK, Libor a HLAVIČKA, Rudolf. *Numerické metody*. Vydání třetí. Brno: Akademické nakladatelství CERM, s.r.o, 2016, 110 stran : černobílé ilustrace. ISBN 978-80-214-5437-8.

[19] SZELISKI, Richard. *Computer Vision: Algorithms and Applications*. 1st ed. London: Springer, 2010, 812 s. ISBN 978-1-84882-934-3. Dostupné také z: `https://szeliski.org/Book`

[20] GONZALEZ, Rafael C a Richard Eugene WOODS. *Digital image processing*. 3rd ed. Upper Saddle River: Pearson ; Prentice Hall, 2008, 954 s. ISBN 0-13-168728-X.

[21] SOJKA, Eduard. *Digitální zpracování a analýza obrazů*. Ostrava: VŠB - Technická univerzita, 2000, 133 s. ISBN 80-7078-746-5.Dostupné také z: `http://mrl.cs.vsb.cz/people/sojka/dzo/digitalni_zpracovani_obrazu.pdf`

[22] SOBEL, Irwin. *Camera models and machine perception*. Stanford University of California, Department of Computer Science. 1970. Dostupné také z: `http://www.dtic.mil/docs/citations/AD0708084`

[23] DRUCKMÜLLEROVÁ, Hana. *Registrace obrazů pomocí fázové korelace*. Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2010.

[24] MIKULÁŠEK, Karel. *Studijní materiály k přednáškám z Teorie grafů* [online]. [cit. 2022-05-15]. Dostupné z: `http://teaching.mikulasek.net/Lectures/graphs.html`

[25] KVĚTNÝ, Michal. *Detekce významných bodů v obrazu pomocí Harrisova detektoru*. Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2019.

[26] GRT. Princip diskrétní konvoluce In: *Wikipedia: the free encyclopedia* [online]. 2006 [cit. 15.5.2022]. Dostupný pod licencí CC BY-SA 3.0 Dostupné z: `https://commons.wikimedia.org/w/index.php?curid=969064`

[27] ALBAWI, Saad, Tareq Abed MOHAMMED a Saad AL-ZAWI. Understanding of a convolutional neural network. 2017 *International Conference on Engineering and Technology (ICET)* [online]. IEEE, 2017, 2017, 1-6 [cit. 2022-05-15]. ISBN 978-1-5386-1949-0. Dostupné z: doi:10.1109/ICEngTechnol.2017.8308186

[28] O'SHEA, Keiron a Ryan NASH. *An Introduction to Convolutional Neural Networks*. CoRR [online]. 2015, (abs/1511.08458) [cit. 2022-05-15]. Dostupné z: doi:10.48550/arXiv.1511.08458

[29] GUO, Tianmei, Jiwen DONG, Henjian LI a Yunxing GAO. Simple convolutional neural network on image classification. *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)* [online]. IEEE, 2017, 2017, 721-724 [cit. 2022-05-15]. ISBN 978-1-5090-3618-9. Dostupné z: doi:10.1109/ICBDA.2017.8078730

[30] LEE, D. T. a B. J. SCHACHTER. *Two algorithms for constructing a Delaunay triangulation* [online]. 1980, **9**(3), 219-242 [cit. 2022-05-15]. ISSN 0091-7036. Dostupné z: doi:10.1007/BF00977785

[31] GUO, Baining, Jai MENON a Brian WILLETTE. *Surface Reconstruction Using Alpha Shapes*. Computer Graphics Forum [online]. 1997, **16**(4), 177-190 [cit. 2022-05-15]. ISSN 0167-7055. Dostupné z: doi:10.1111/1467-8659.00178

[32] BERNARDINI, F., J. MITTLEMAN, H. RUSHMEIER, C. SILVA a G. TAUBIN. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* [online]. 1999, **5**(4), 349-359 [cit. 2022-05-15]. ISSN 1077-2626. Dostupné z: doi:10.1109/2945.817351

[33] CHRSCHN. Example of low poly triangle mesh representing a dolphin. In: *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-16]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/f/fb/Dolphin_triangle_mesh.png`

[34] JENSEN, Tommy R a Bjarne TOFT. *Graph Coloring Problems*. New York: John Wiley & Sons, Incorporated, 1994. ISBN 0471028657.

[35] PLEJ. Strom(informatika). In: *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-16]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/6/63/Strom%28informatika%29.jpg`

[36] RUŽICKÝ, Eugen a Andrej FERKO. *Počítačová grafika a spracovanie obrazu*. Bratislava: SAPIENTIA, 1995. ISBN 80-967180-2-9. Dostupné také z: `https://flurry.dg.fmph.uniba.sk/webog/SuboryOG/ferko/PocitacovaGrafikaSpracovanieObrazu.pdf`

[37] VŠELICHA, Radovan. *Laserové skenování: bakalářská práce = Laser scanning*. Praha, 2017. Bakalářská práce (Bc.). České vysoké učení technické v Praze. Dopravní fakulta, Ústav letecké dopravy. Vedoucí práce Jakub Kraus.

[38] MOUNT, Dave. *CMSC 420: Data Structures* [online]. 2001 [cit. 2022-05-16]. Dostupné z: `http://www.cs.umd.edu/~mount/420/Lects/420lects.pdf`. Lecture Notes.

[39] WHITETIMBERWOLF. Left: Recursive subdivision of a cube into octants. Right: The corresponding octree. In: *Wikipedia: the free encyclopedia* [online]. 2010 [cit. 2022-05-16]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/2/20/Octree2.svg`. Dostupný pod licencí CC BY-SA 3.0.

[40] ZHOU, Kun, Qiming HOU, Rui WANG a Baining GUO. Real-time KD-tree construction on graphics hardware. *ACM Transactions on Graphics* [online]. 2008, 27(5), 1-11 [cit. 2022-05-16]. ISSN 0730-0301. Dostupné z: doi:10.1145/1409060.1409079

[41] HU, Linjia, Saeid NOOSHABADI a Majid AHMADI. Massively parallel KD-tree construction and nearest neighbor search algorithms. *2015 IEEE International Symposium on Circuits and Systems (ISCAS)* [online]. IEEE, 2015, 2015, 2752-2755 [cit. 2022-05-16]. ISBN 978-1-4799-8391-9. Dostupné z: doi:10.1109/ISCAS.2015.7169256

[42] PINKHAM, Reid, Shuqing ZENG a Zhengya ZHANG. QuickNN: Memory and Performance Optimization of k-d Tree Based Nearest Neighbor Search for 3D Point Clouds. *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)* [online]. IEEE, 2020, 2020, 180-192 [cit. 2022-05-16]. ISBN 978-1-7281-6149-5. Dostupné z: doi:10.1109/HPCA47549.2020.00024

[43] KIWISUNSET. K-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2). In: *Wikipedia: the free encyclopedia* [online]. 2006 [cit. 2022-05-16]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/b/bf/Kdtree_2d.svg`. Dostupný pod licencí CC BY-SA 3.0.

[44] MYGUEL. The resulting k-d tree. In: *Wikipedia: the free encyclopedia* [online]. 2008 [cit. 2022-05-16]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/2/25/Tree_0001.svg`

[45] History of the camera. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-05-16]. Dostupné z: `https://en.wikipedia.org/wiki/History_of_the_camera`

[46] Camera Calibration. *OpenCV Docs* [online]. [cit. 2022-05-16]. Dostupné z: `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`

[47] EICHHORN, G. N., A. BOWMAN, S. K. HAIGH a S. STANIER. Low-cost digital image correlation and strain measurement for geotechnical applications. *Strain* [online]. 2020, 56(6) [cit. 2022-05-16]. ISSN 0039-2103. Dostupné z: doi:10.1111/str.12348. Dostupný pod licencí CC BY 4.0.

[48] *OpenCV Docs* [online]. [cit. 2022-05-16]. Dostupné z: `https://docs.opencv.org/4.x/index.html`

[49] Rodrigues' Rotation Formula. *Wolfram MathWorld* [online]. [cit. 2022-05-16]. Dostupné z: `https://mathworld.wolfram.com/RodriguesRotationFormula.html`

[50] HARRIS, Charles R., K. Jarrod MILLMAN, Stéfan J. VAN DER WALT, et al. Array programming with NumPy. *Nature* [online]. 2020, **585**(7825), 357-362 [cit. 2022-05-19]. ISSN 0028-0836. Dostupné z: doi:10.1038/s41586-020-2649-2

[51] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* [online]. 2004, **60**(2), 91-110 [cit. 2022-05-16]. ISSN 0920-5691. Dostupné z: doi:10.1023/B:VISI.0000029664.99615.94

[52] BAY, Herbert, Tinne TUYTELAARS a Luc VAN GOOL. SURF: Speeded Up Robust Features. *Computer Vision – ECCV* 2006 [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 2006, 404-417 [cit. 2022-05-16]. Lecture Notes in Computer Science. ISBN 978-3-540-33832-1. Dostupné z: doi:10.1007/11744023_32

[53] ROSTEN, Edward a Tom DRUMMOND. Machine Learning for High-Speed Corner Detection. *Computer Vision – ECCV* 2006 [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 2006, 430-443 [cit. 2022-05-16]. Lecture Notes in Computer Science. ISBN 978-3-540-33832-1. Dostupné z: doi:10.1007/11744023_34

[54] RUBLEE, Ethan, Vincent RABAUD, Kurt KONOLIGE a Gary BRADSKI. ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision* [online]. IEEE, 2011, 2011, 2564-2571 [cit. 2022-05-16]. ISBN 978-1-4577-1102-2. Dostupné z: doi:10.1109/ICCV.2011.6126544

[55] CALONDER, Michael, Vincent LEPETIT, Christoph STRECHA a Pascal FUA. BRIEF: Binary Robust Independent Elementary Features. *Computer Vision – ECCV 2010* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 2010, 778-792 [cit. 2022-05-16]. Lecture Notes in Computer Science. ISBN 978-3-642-15560-4. Dostupné z: doi:10.1007/978-3-642-15561-1_56

[56] HARRIS, Chris a Mike STEPHENS. A Combined Corner and Edge Detector. *Alvey Vision Conference*. 1988, 147-151.

[57] MORAVEC, Hans. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Pittsburgh, PA: Carnegie Mellon University, 1980.

[58] COLLINS, Robert. *Lecture 06: Harris Corner Detector* [online]. [cit. 2022-05-16]. Dostupné z: `https://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf`. Prezentace.

[59] TYAGI, Deepanshu. Introduction To Feature Detection And Matching. In: *Medium* [online]. 3.1.2019 [cit. 2022-05-16]. Dostupné z: `https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d`

[60] TYAGI, Deepanshu. Introduction to Harris Corner Detector. In: *Medium* [online]. 16.3.2019 [cit. 2022-05-16]. Dostupné z: `https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6`

[61] TYAGI, Deepanshu. Introduction to FAST (Features from Accelerated Segment Test). In: *Medium* [online]. 2.1.2019 [cit. 2022-05-16]. Dostupné z: `https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65`

[62] TYAGI, Deepanshu. Introduction to BRIEF(Binary Robust Independent Elementary Features). In: *Medium* [online]. 19.3.2019 [cit. 2022-05-16]. Dostupné z: https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6

[63] TYAGI, Deepanshu. Introduction to ORB (Oriented FAST and Rotated BRIEF). In: *Medium* [online]. 1.1.2019 [cit. 2022-05-16]. Dostupné z: https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

[64] ORB (Oriented FAST and Rotated BRIEF). *OpenCV Docs* [online]. [cit. 2022-05-16]. Dostupné z: https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

[65] SIPIRAN, Ivan a Benjamin BUSTOS. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer* [online]. 2011, **27**(11), 963-976 [cit. 2022-05-16]. ISSN 0178-2789. Dostupné z: doi:10.1007/s00371-011-0610-y

[66] SMITH, Lindsay I. *A tutorial on Principal Components Analysis* [online]. 2002 [cit. 2022-05-19]. Dostupné z: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[67] Bresenham's circle drawing algorithm. In: *Geeks for Geeks* [online]. 18.1.2021 [cit. 2022-05-16]. Dostupné z: https://www.geeksforgeeks.org/bresenhams-circle-drawing-algorithm/

[68] BROWN, M., R. SZELISKI a S. WINDER. Multi-Image Matching Using Multi-Scale Oriented Patches. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05* [online]. IEEE, 2005, 510-517 [cit. 2022-05-16]. ISBN 0-7695-2372-2. Dostupné z: doi:10.1109/CVPR.2005.235

[69] WANG, Xiaolong. *Learning and Reasoning with Visual Correspondence in Time.* 2019.

[70] SHILKROT, Roy. Exploring Structure from Motion Using OpenCV. In: Packt [online]. 9.1.2017 [cit. 2022-05-17]. Dostupné z: https://hub.packtpub.com/exploring-structure-motion-using-opencv/

[71] CWOO. Moore-Penrose generalized inverse. In: *PlanetMath* [online]. 22.3.2013 [cit. 2022-05-17]. Dostupné z: https://planetmath.org/MoorePenroseGeneralizedInverse

[72] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus. *Communications of the ACM* [online]. 1981, **24**(6), 381-395 [cit. 2022-05-19]. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

[73] SUGAYA, Yasuyuki a Kenichi KANATI. High Accuracy Computation of Rank-Constrained Fundamental Matrix. *BMVC* [online]. 2007 [cit. 2022-05-17]. Dostupné z: https://www.dcs.warwick.ac.uk/bmvc2007/proceedings/CD-ROM/papers/paper-81.pdf

[74] DE FRANÇA, José A., Maria B. DE M. FRANÇA, Robinson HOTO a M. R. STEMMER. *A Simple Linear and Iterative LMS Algorithm for Fundamental Matrix Estimating*. Dostupné také z: `http://www.uel.br/pessoal/josealexandre/publications/congress/franca04simple.pdf`

[75] Open3D Docs. *Open3D (C++ API)* [online]. [cit. 2022-05-18]. Dostupné z: `http://www.open3d.org/docs/release/cpp_api/index.html`

[76] Global registration. In: *Open3D* [online]. [cit. 2022-05-18]. Dostupné z: `http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html`

[77] ICP registration. In: *Open3D* [online]. [cit. 2022-05-18]. Dostupné z: `http://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html`

[78] RUSU, R.B., N. BLODOW, Z.C. MARTON a M. BEETZ. Aligning point cloud views using persistent feature histograms. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* [online]. IEEE, 2008, 2008, 3384-3391 [cit. 2022-05-18]. ISBN 978-1-4244-2057-5. Dostupné z: doi:10.1109/IROS.2008.4650967

[79] KOSECKA, Jana. *Introduction to Mobile Robotics: Iterative Closest Point Algorithm* [online]. [cit. 2022-05-18]. Dostupné z: `https://cs.gmu.edu/~kosecka/cs685/cs685-icp.pdf`. Prezentace.

[80] RUSU, Radu Bogdan, Nico BLODOW a Michael BEETZ. Fast Point Feature Histograms (FPFH) for 3D registration. *2009 IEEE International Conference on Robotics and Automation* [online]. IEEE, 2009, 2009, 3212-3217 [cit. 2022-05-18]. ISBN 978-1-4244-2788-8. Dostupné z: doi:10.1109/ROBOT.2009.5152473

[81] ZHANG XIMIN, LIU RAN, ZHOU YIYUAN, WAN WANGGEN a LU LIBING. Normal estimation algorithm for point cloud using KD-tree. *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)* [online]. Institution of Engineering and Technology, 2013, 2013, 286-289 [cit. 2022-05-16]. ISBN 978-1-84919-707-6. Dostupné z: doi:10.1049/cp.2013.1978

[82] KULLBACK, S. a R. A. LEIBLER. On Information and Sufficiency. *The Annals of Mathematical Statistics* [online]. 1951, **22**(1), 79-86 [cit. 2022-05-18]. ISSN 0003-4851. Dostupné z: doi:10.1214/aoms/1177729694

[83] KATZ, Sagi a Ayellet TAL. On the Visibility of Point Clouds. 2015 *IEEE International Conference on Computer Vision (ICCV)* [online]. IEEE, 2015, 2015, 1350-1358 [cit. 2022-05-16]. ISBN 978-1-4673-8391-2. Dostupné z: doi:10.1109/ICCV.2015.159

[84] Python [online]. [cit. 2022-05-19]. Dostupné z: `https://www.python.org/`

[85] MARELLI, Davide, Simone BIANCO a Gianluigi CIOCCA. *IVL-SYNTHSFM-v2: a dataset for the evaluation of 3D reconstruction pipelines* [online]. [cit. 2022-05-19]. Dostupné z: doi:10.17632/fnxy8z8894.1

[86] BIANCO, Simone, Gianluigi CIOCCA a Davide MARELLI. Evaluating the Performance of Structure from Motion Pipelines. *Journal of Imaging* [online]. 2018, **4**(8) [cit. 2022-05-19]. ISSN 2313-433X. Dostupné z: doi:10.3390/jimaging4080098

[87] MARELLI, Davide, Simone BIANCO, Luigi CELONA a Gianluigi CIOCCA. A Blender plug-in for comparing Structure from Motion pipelines. *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)* [online]. IEEE, 2018, 2018, 1-5 [cit. 2022-05-19]. ISBN 978-1-5386-6095-9. Dostupné z: doi:10.1109/ICCE-Berlin.2018.8576196

Seznam použitých zkratek a symbolů

BRIEF	*Binary Robust Independent Elementary Features*
FAST	*Features from Accelerated Segment Test*
FPFH	*Fast Point Feature Histograms*
ICP	*Iterative Closest Point*
MOPS	*Multi-scale Oriented PatcheS*
ORB	*Oriented FAST and Rotated BRIEF*
PCA	*Principal Component Analysis* – Analýza hlavních komponent
RANSAC	*RAndom SAmple Consensus*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded-Up Robust Features*
SVD	*Singular Value Decomposition* – Singulární rozklad

x, y	proměnné
$\mathbf{x}, \mathbf{y}$	vektory
A, B	množiny
X, Y	body
$\mathbf{X}, \mathbf{Y}$	homogenní souřadnice bodů
$\mathbf{A}, \mathbf{B}$	matice
f, g	obrazová funkce
i	obrazová funkce, digitální obraz
r, g, b	barevné kanály digitálního obrazu
$\mathbb{E}^2, \mathbb{E}^3$	eukleidovská rovina, eukleidovský prostor
$\mathbb{P}^2, \mathbb{P}^3$	projektivní rovina, projektivní prostor

Seznam příloh

Přiložený adresář má následující strukturu.

```
root\
 | src\
 | res\
    | tea\
        |  project.txt
        |  tea.ply
        |  01.jpg
        |  ...
        |  33.jpg
    | test\
        |  project.txt
        |  test.ply
        |  01.jpg
        |  ...
        |  36.jpg
    | vase\
        |  project.txt
 | program.exe
 | configuration.py
 | calibration.py
```

Adresář obsahuje tyto soubory a složky.

`program.exe` – zkompilovaný mapovací program;

`calibration.py` – skript pro zjištění vnitřních parametrů kamery;

`configuration.py` – skript pro konfiguraci souborů projektu a spouštění mapování;

`src` – adresář obsahující zdrojový kód mapovacího programu;

`res` – adresář obsahující datasety;

`tea` – adresář obsahující dataset „tea“;

`test` – adresář obsahující dataset „test“;

`vase` – adresář obsahující projektový soubor pro dataset „vase“.