

# **Metody pro navigaci mobilního robota pomocí kamery**

Methods of camera-based navigation of mobile robot

Jan Vaňhara

---

Bakalářská práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Jan VAŇHARA  
Osobní číslo: A09167  
Studijní program: B3902 Inženýrská informatika  
Studijní obor: Informační a řídicí technologie  
Forma studia: prezenční

Téma práce: Metody pro navigaci mobilního robota pomocí kamery

Zásady pro vypracování:

1. Prostudujte základní metody zpracování obrazu – např. filtrace v prostorové a frekvenční oblasti, korelace, detekce hran atd.
2. Prostudujte metody 3D rekonstrukce obrazu, implementované v knihovnách Matlabu, OpenCV a dalších.
3. Prostudujte a analyzujte metody pro detekci objektů a překážek v obraze.
4. Prostudujte metody pro navigaci mobilního robota v prostoru pomocí na něm umístěné stereo kamery. Pro začátek zvolte jednoduché případy – např. průchod po přímce s jedinou překážkou.
5. Navrhněte architekturu funkčního systému pro navigaci autonomního robota pomocí obrazu z kamer, který v budoucnosti implementujete v rámci navazující inženýrské diplomové práce.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. BREZNAN, M. a R. HRONEC. Reconstruction of 3-D image from stereo pictures. 2005. ISBN 953-7044-01-4.
2. SCHARSTEIN, D., R. SZELISKI a R. ZABIH. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Kauai, HI, 2001. ISBN 0-7695-1327-1.
3. YING-YUAN HUANG a MEI-YUNG CHEN. 3D object model recovery from 2D images utilizing corner detection. Macao, 2011. ISBN 978-1-61284-472-5.
4. ZHENGYOU ZHANG. Flexible camera calibration by viewing a plane from unknown orientations. Kerkyra, 1999. ISBN 0-7695-0164-8.
5. BREZNAN, M. Hybrid Method of 3-D Image Reconstruction from Stereo Pictures. Istanbul, 2008. ISBN 978-1-4244-1755-1.
6. CHUNG, R. a CHI-KIN HO. Using 2D active contour models for 3D reconstruction from serial sections. Vienna, 1996. ISBN 0-8186-7282-X.
7. LI QIAN. Image 3D reconstruction system for indoor environment based on JAVA 3D. Dalian, 2010. ISBN 978-1-4244-6893-5.

Vedoucí bakalářské práce:

**Ing. Tomáš Dulík, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

**24. února 2013**

Termín odevzdání bakalářské práce:

**14. června 2013**

Ve Zlíně dne 24. února 2013

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## ABSTRAKT

Tato bakalářská práce se zabývá metodami zpracování informací z kamery, popř. více kamer za účelem rekonstrukce 3D modelu scény, detekce objektů a jejich sledování v sekvenci obrazů.

Práce popisuje metody snímání obrazu jako je stereo kamera, LIDAR a strukturované světlo, které jsou nejpoužívanější při rekonstrukci 3D scény. Dále popisuje metody pro kalibraci obrazu, využívající epipolární geometrii a početní metody rekonstrukce bodů v prostoru, které využívají triangulaci.

Pro detekci objektů jsou v práci popsány metody segmentace obrazu, prahování a detekce hran.

Část práce je také věnována sledování pohybu kamery metodou hledání jedinečných oblastí v sekvencích obrazů.

### **Klíčová slova:**

Stereo kamera, LIDAR, strukturované světlo, triangulace, 3D rekonstrukce, detekce objektů, sledování objektů, OpenCV, Matlab.

## **ABSTRACT**

These bachelor's thesis deal with methods of information processing from the camera or more cameras for reconstruction of 3D scenes model, object detection and their tracking in the sequence of images.

This work describes methods of capturing like a stereo camera, LIDAR, structured light which are used for reconstruction of 3D scenes. It also describes methods for calibration image, using epipolar geometry and numerical methods of reconstruction points in area, which use triangulation.

In the thesis for the object detection are described methods of image segmentation, thresholding and edge detection.

Part of work is devoted to tracking the movement of cameras with methods of search unique area in the sequence of images.

### **Keywords:**

Stereo camera, LIDAR, structured light, triangulation, 3D reconstruction, object detection, tracking objects, OpenCV, Matlab.

## **PODĚKOVÁNÍ, MOTTO**

Děkuji svému vedoucímu bakalářské práce, Ing. Tomáši Dulíkovi, PhD., za cenné rady při konzultacích. Dále chci poděkovat svým rodičům a přátelům za trpělivost a podporu při studiu.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD .....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>12</b>
<b>1 REKONSTRUKCE 3D OBRAZU .....</b>	<b>13</b>
1.1 STEREO KAMERA.....	13
1.1.1 O stereo vidění .....	13
1.1.2 2D triangulace.....	14
1.1.3 Hloubka a rozlišitelnost objektů .....	15
1.1.4 Zarovnání kamer, rektifikace (rectification) .....	17
1.1.5 Epipolární geometrie .....	19
1.1.6 Esenciální a fundamentální matice .....	21
1.1.7 Matematika esenciální matice.....	22
1.1.8 Matematika fundamentální matice .....	23
1.1.9 Stereo korespondence (stereo matching) .....	23
1.1.10 Souhrn .....	24
1.2 STRUKTUROVANÉ SVĚTLO .....	24
1.2.1 Princip strukturovaného světla a 3D triangulace .....	26
1.2.2 3D rekonstrukce a 3D skener .....	27
1.2.3 Microsoft Kinect.....	28
1.2.4 Souhrn .....	29
1.3 LIDAR (LIGHT DETECTOR AND RANGING) .....	29
1.3.1 Princip .....	30
1.3.2 Google Car .....	30
1.3.3 Souhrn .....	33
<b>2 DETEKCE OBJEKTŮ V OBRAZE.....</b>	<b>34</b>
2.1 PRAHOVÁNÍ (THRESHOLDING).....	34
2.2 DETEKCE HRAN .....	35
2.3 SEGMENTACE ZALOŽENÁ NA HLEDÁNÍ OBLASTÍ – DĚLENÍ A SPOJOVÁNÍ OBLASTÍ (SPLIT AND MERGE) .....	36
2.4 DETEKCE OBJEKTŮ .....	37
<b>3 POHYB A SLEDOVÁNÍ .....</b>	<b>38</b>
3.1 HLEDÁNÍ ROHŮ (CORNER FINDING) .....	38
3.2 POHYB OPTIKY .....	39
<b>II PRAKTICKÁ ČÁST .....</b>	<b>41</b>
<b>4 MAPA STEREO DISPARITY – PŘÍKLADOVÉ DEMO V MATLABU .....</b>	<b>42</b>
4.1 ZOBRAZENÍ A ZÁKLADNÍ FUNKCIONALITA.....	42
4.2 IMPLEMENTACE.....	43
4.2.1 Vstupy a GUI.....	43
4.2.2 Výpočet disparity, funkce stereo().....	44
4.2.3 Funkce slide_images().....	44
4.2.4 Funkce winner_take_all().....	46
4.2.5 Funkce modefilt2() .....	47
<b>5 OPTICAL FLOW PŘÍKLADOVÉ DEMO – KNIHOVNA OPENCV .....</b>	<b>48</b>



5.1	FUNKCIONALITA A ZOBRAZENÍ.....	48
5.2	IMPLEMENTACE.....	50
<b>ZÁVĚR .....</b>		<b>52</b>
<b>ZÁVĚR V ANGLIČTINĚ.....</b>		<b>53</b>
<b>SEZNAM POUŽITÉ LITERATURY .....</b>		<b>54</b>
<b>SEZNAM OBRÁZKŮ.....</b>		<b>58</b>
<b>SEZNAM PŘÍLOH .....</b>		<b>60</b>

## ÚVOD

Tato práce popisuje možnosti a způsoby zpracování informací z kamery, popř. více kamer za účelem rekonstrukce 3D modelu scény, detekce objektů a jejich sledování v sekvenci obrazů.

Zpracováním obrazů na mobilní robotické platformě se zabývá mnoho výzkumných týmů v akademické i komerční sféře. Na obrázku 1 je zobrazen autonomní robotický automobil Stanley [28]. Zorné pole kamery tohoto automobilu snímá obraz ve vzdálenosti 80 a více metrů od vozu, který doplňuje obraz ze střešních LIDARů, detailně mapujících nejbližší okolí auta.



Obrázek 1: Autonomní robotický automobil Volkswagen Touareg, Stanley [29]

Úkolem metod počítačového vidění je zpracování obrazu tak, aby počítač dokázal vnímat okolí automobilu a vyhýbat se překážkám na cestě.

Tato bakalářská práce je členěna následujícím způsobem:

- Kapitola 1 popisuje nejběžněji používané metody rekonstrukce 3D obrazu za použití kamery. U metody stereo kamery se využívá dvou a více kamer. Za pomoci jejich vzájemné geometrie lze počítat hloubku. Metoda strukturovaného světla využívá nasvícení scény přes mřížku s určitým vzorem. Díky deformaci tohoto vzoru na

objektech v prostoru se pak dá počítat hloubková mapa obrazu. Naproti tomu LIDARová metoda vyšle laserový impuls a počítá dobu, kdy se jako odražený vrátí. Protože známe rychlost šíření světla, můžeme počítat, jak daleko se nasvícený bod od LIDARu nachází.

- Kapitola 2 se zabývá detekcí a rozpoznáním objektů v obraze. Metoda prahování je založena na určení úrovně jasu, od které se odvíjí vyhodnocená maximální a minimální nová úroveň. Metoda detekce hran hledá hranici tam, kde je největší rozdíl v jasových hodnotách, využívá gradientního počtu. A metoda segmentace obrazu dělí obraz na menší část se stejnou úrovní určitého kritéria homogenity (jas, barva, textura, apod.).
- Kapitola 3 popisuje určení pohybu kamery ze sledu obrazových scén. Zabývá se hledáním unikátních bodů v jednom obraze s cílem najít tyto body i v obraze druhém.
- Kapitola 4 je zaměřena na příklad získání mapy disparity programem, napsaným v Matlabu.
- Kapitola 5 je příklad určení pohybu kamery z obrazu. Program využívá knihovnu OpenCV.

## **I. TEORETICKÁ ČÁST**

## 1 REKONSTRUKCE 3D OBRAZU

### 1.1 Stereo kamera

Stereo kamera je tvořena dvěma vzájemně paralelně umístěnými kamerami na jedné společné vodorovné ose, ze kterých získáváme stereo pár (dva obrazy stejné scény). Obě kamery mají identickou ohniskovou vzdálenost  $f$  a jsou rovnoběžně přemísťovány v prostoru. Obrázek 2 ukazuje, jak stereo kamera může vypadat.



Obrázek 2: Takto může vypadat stereo kamera. [19]

#### 1.1.1 O stereo vidění

Úkolem stereo vidění je nalezení odpovídajících si bodů na jednom obrázku v obrázku druhém. Této schopnosti říkáme korespondence (z *angl. slova correspondence*). Díky korespondujícím bodům a se známou vzdáleností mezi kamerami (*baseline*, nebo-li *parallax*), můžeme odhadnout 3D umístění určitého bodu. Aby počítání nebylo příliš nákladné, využijeme znalost geometrie a zúžíme prohledávaný prostor jak je to jen možné. Při použití dvou kamer, počítačové stereo vidění v praxi využívá čtyř kroků:

1. Matematicky odstraníme distorzi (zkreslení) čočky. Musíme získat nezkreslený obrázek. Tento proces se nazývá z angličtiny *undistortion process*.
2. Nastavíme úhel a vzdálenost mezi kamerami (*angl. rectification*). Získáme tak řádkově zarovnaný (*angl. row-aligned*) a usměrněný (*angl. rectified*) obrázek. To znamená, že jsou plochy obou obrázků navzájem rovnoběžné a jednotlivé řady v obrázku jsou naprosto přesně zarovnaný (ve stejném směru souřadnice  $x$ , máme stejné souřadnice  $y$  pro oba obrázky).

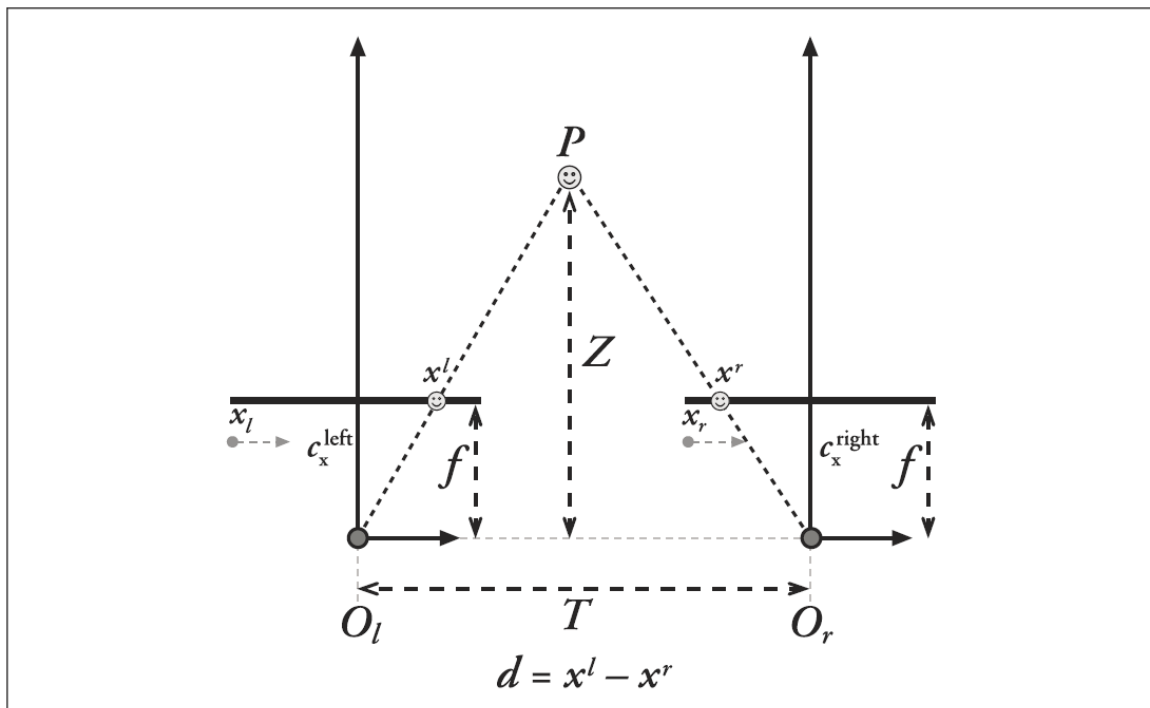
3. Korespondenci najdeme shodné výrazné rysy či oblasti v levém i pravém pohledu kamer. Získáme tak mapu disparity, kde disparity jsou rozdíly v x-souřadnicích na ploše obrázku nalezených výrazných oblastí v pohledech levé a pravé kamery:  $x^l - x^r$ .
4. Pokud známe geometrické rozmístění kamer, můžeme použít mapu disparity na výpočet vzdálenosti pomocí triangulace (*reprojection process*). Získáme tak hloubkovou mapu (*depth map*). [1]

### 1.1.2 2D triangulace

Předpokládejme, že máme perfektní, nezkreslenou, zarovnanou měrnou stereo výbavu, jak můžeme vidět na obrázku 3. Jde o dvě kamery vedle sebe, směr čoček a jejich optických os je navzájem rovnoběžný. Známe taky vzdálenost mezi optickými osami.

Optická osa je paprsek, který vede ze středu promítání  $O$  skrze hlavní bod  $c$  (*angl. principal point*), optickou osu nazýváme také jako hlavní paprsek (*angl. principal ray*). Hlavní bod je obraz středu druhé kamery v obrázku z kamery první. Více v kapitole 1.1.5. V ideálním případě se dva rovnoběžné hlavní paprsky setkávají v nekonečnu. Ohnisková vzdálenost je u obou kamer shodná, tedy  $f_l = f_r$ . Předpokládejme, že hlavní body jsou kalibrovány – máme souřadnice pixelu na levém obrázku, který se vztahuje k souřadnicím na pravém obrázku. Jinak řečeno, našli jsme bod na levém obrázku, který odpovídá bodu na pravém obrázku.

Dále předpokládejme, že jsou řady pixelů kalibrovány, že každá řada pixelů z jedné kamery přesně odpovídá s korespondující řadou z druhé kamery. Takovéto zarovnání kamer můžeme nazvat čelně rovnoběžné (*angl. frontal parallel*). Také předpokládáme, že můžeme fyzicky najít bod  $P$  v získaném levém i pravém obrazu  $p_l$  a  $p_r$ , kde mají příslušné vodorovné souřadnice  $x^l$  a  $x^r$ .



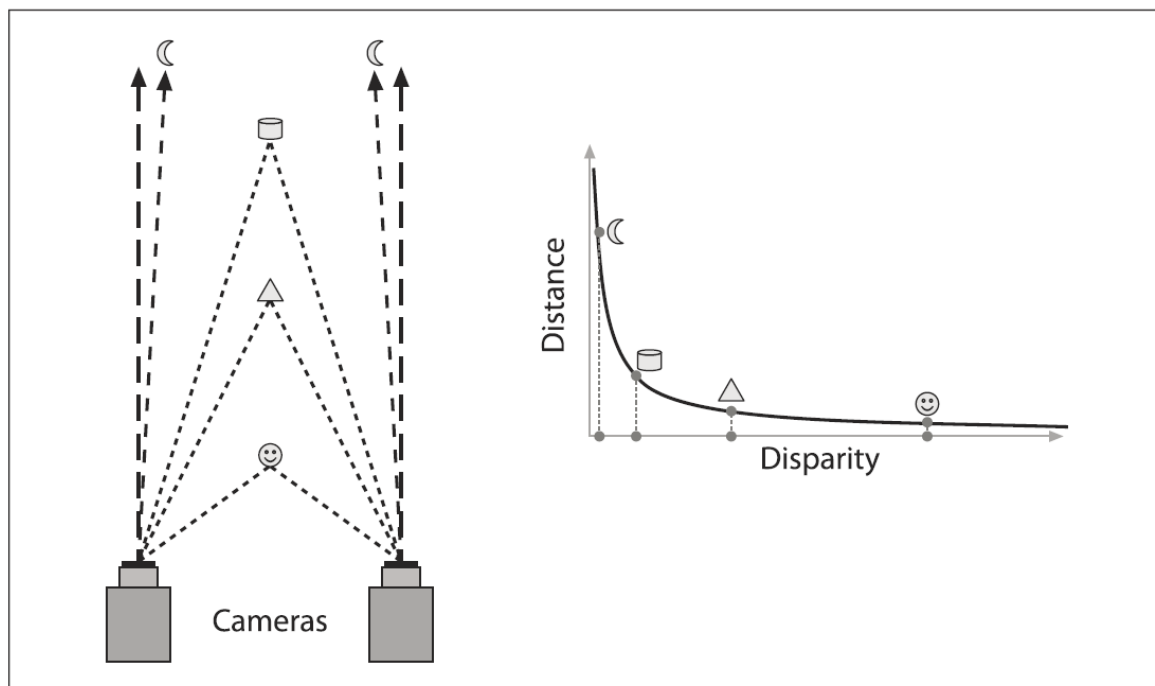
Obrázek 3: Geometrické zobrazení ideální stereo výbavy. [1]

V tomto jednoduchém případě, můžeme ze získaných souřadnic  $x^l$  a  $x^r$  zjistit, že hloubka je nepřímo úměrná disparitě mezi těmito pohledy, kde disparitu získáme jednoduše  $d = x^l - x^r$ . Z obrázku 3 můžeme lehce odvodit hloubku  $Z$  použitím trojúhelníku.

$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x^l - x^r} \quad (1.1)$$

### 1.1.3 Hloubka a rozlišitelnost objektů

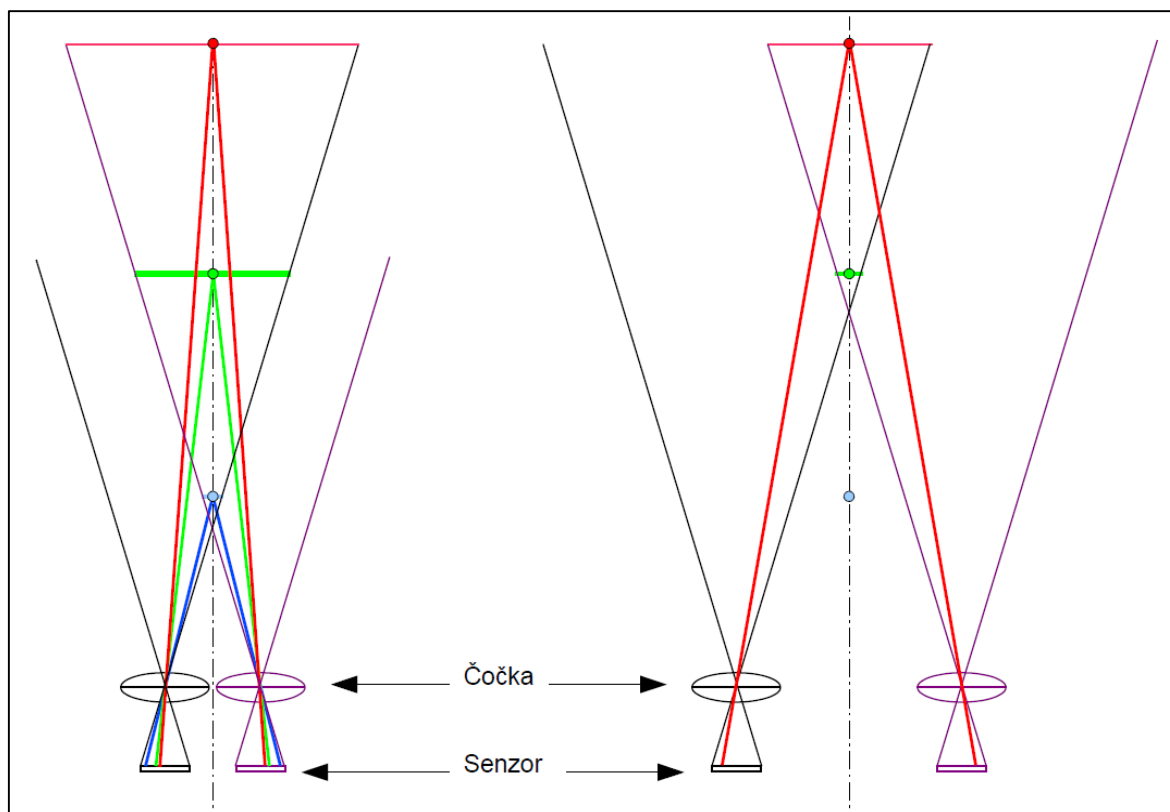
Hloubka je nepřímo úměrná disparitě. Jinak řečeno, když je disparita blízká 0, malé rozdíly disparity znamenají velké rozdíly v hloubce. Když má disparita vysoké hodnoty, její malé rozdíly v těchto číslech pak nemají skoro žádný vliv na hloubku. Takže ze stereo vidění můžeme získat vysoké rozlišení hloubky pouze pro objekty relativně blízko kamer. Obrázek 4 ukazuje vztah mezi změnou disparity a vzdáleností objektů. [1]



Obrázek 4: Nepřímá úměra mezi hloubkou a disparitou. [1]

Vzájemná vzdálenost kamer se označuje jako **parallax**. Pro rozpoznávání vzdálenějších objektů můžeme využít větší parallax. Na obrázku 5 můžeme vidět tři kuličky v různé vzdálenosti. Pro bližší vzdálenost kamer (parallax je nejmenší) vidíme modrou kuličku v levé kameře na levém kraji senzoru, v pravé kameře na pravém kraji senzoru. Při počítání vzdálenosti kuličky (její hloubky v obraze) budeme mít největší přesnost, právě kvůli velké disparitě. Naproti tomu u červené kuličky budeme s malou parallaxou velmi obtížně odhadovat její vzdálenost. Pro lepší rozlišitelnost vzdálenosti u červené kuličky tak bychom měli oddálit kamery od sebe nebo prodloužit ohniskovou vzdálenost.

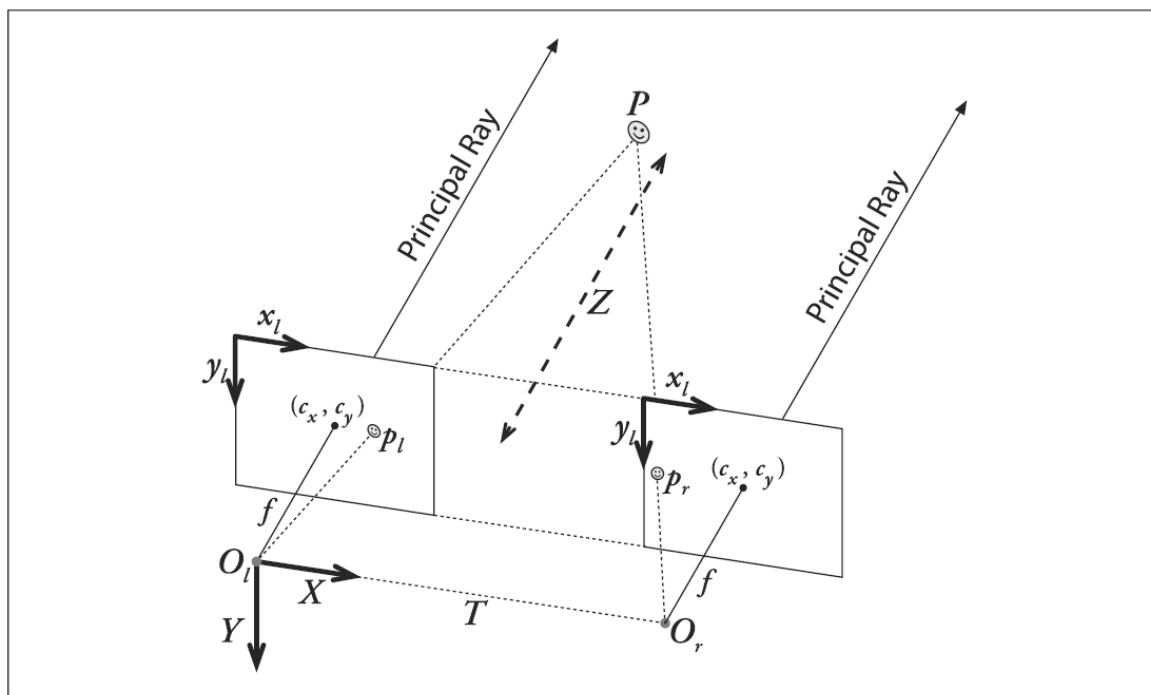




Obrázek 5: Vlevo nastavení kamer pro bližší objekty, vpravo nastavení kamer pro vzdálenější objekty.

#### 1.1.4 Zarovnání kamer, rektifikace (rectification)

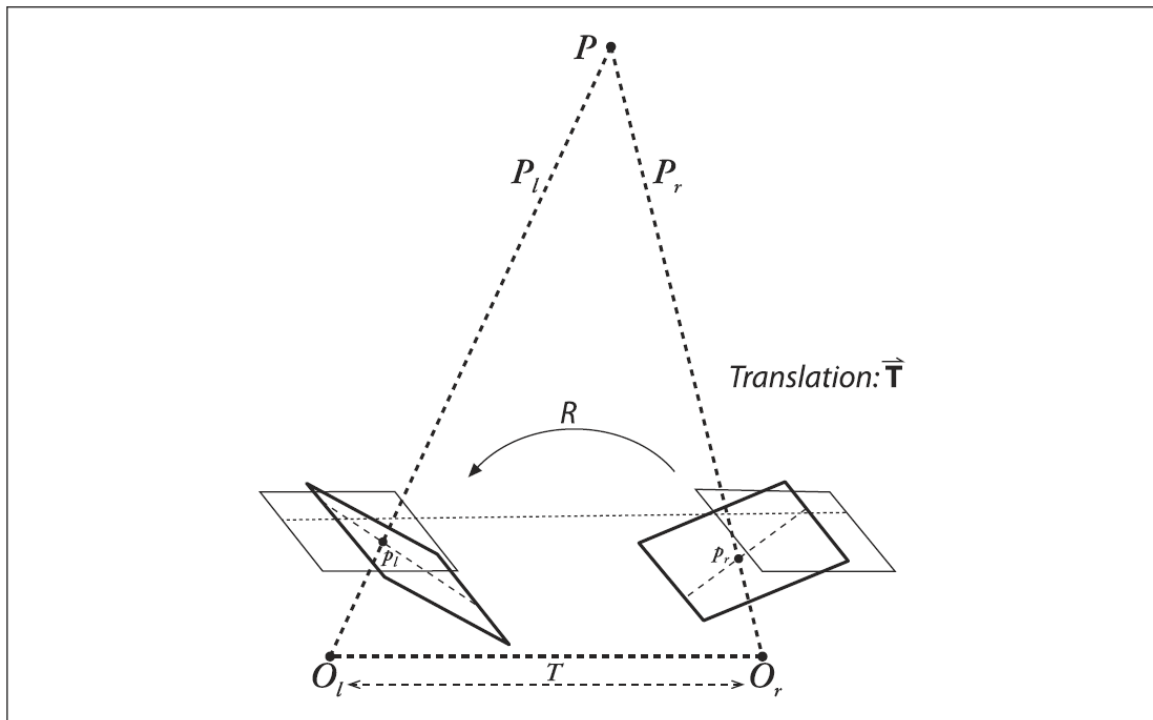
Na obrázku 6 je zobrazena geometrie v prostoru při používání stereo vidění. Začátek souřadnicového systému je nahoře vlevo a pixely jsou na levém obrázku označeny souřadnicemi  $(x_l, y_l)$  a na pravém obrázku  $(x_r, y_r)$ . Střed projekce je v  $O_l$  a  $O_r$  s hlavními paprsky (*principal rays*), protínajícími obrazovou plochu v hlavním bodě (*principal point*)  $(c_x, c_y)$ . Zobrazená sestava je již po matematické korekci a kamery jsou řádkově zarovnaný (rovnoběžně a horizontálně). Kamery jsou od sebe ve vzdálenosti  $T$  a mají stejnou ohniskovou vzdálenost  $f$ . S touto sestavou je pak relativně jednoduché vypočítat vzdálenost.



Obrázek 6: Nejčastěji používaný souřadnicový systém: počátek souřadnicového systému je vlevo nahoře, plochy obrazů jsou řádkově zarovnány. [1]

Reálně nejsou kamery téměř nikdy zarovnány a kalibrovány tak, jak ukazuje obrázek 6. Musíme matematicky opravit obrazy, abychom dosáhli tohoto uspořádání. Také bychom měli synchronizovat kamery tak, aby nám dávali obraz ve stejný čas, zvláště při pohyblivé scéně. [1]

Obrázek 7 nám přibližuje reálnou situaci mezi dvěma kamerami a ukazuje, jakého matematického zarovnání chceme dosáhnout. K matematickému zarovnání využijeme epipolární geometrii.

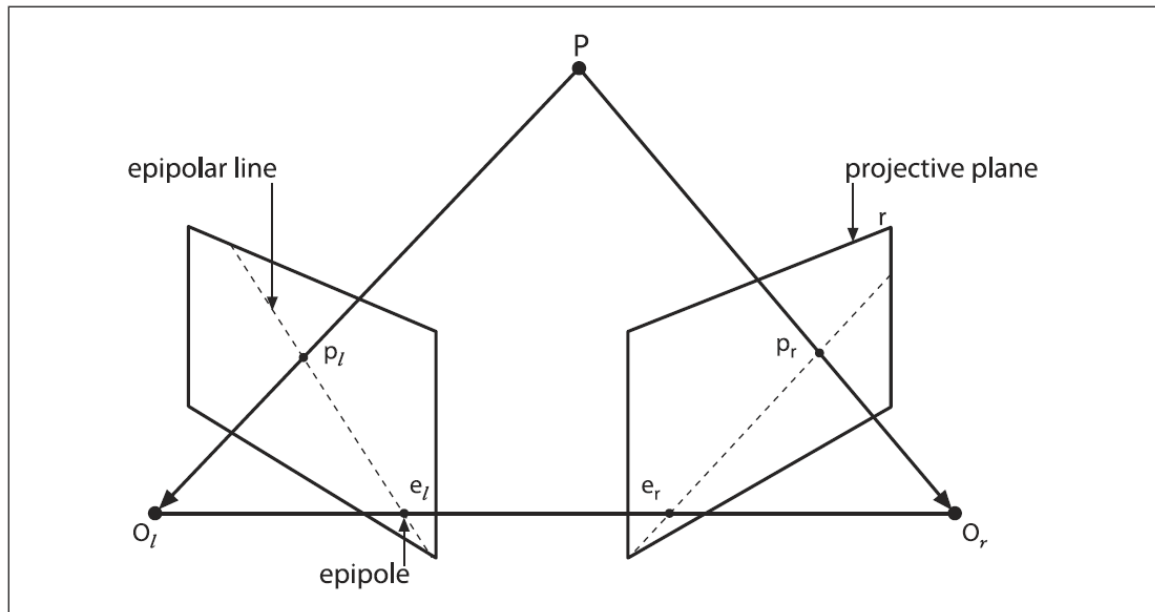


Obrázek 7: Matematicky potřebujeme zarovnat obrazové plochy do jedné roviny tak, aby řádky pixelů obou kamer si odpovídali. Téměř nikdy se nám nepodaří získat ideální obrazy z kamer fyzicky. [1]

### 1.1.5 Epipolární geometrie

Geometrie, popisující stereo zobrazování se nazývá *epipolární geometrie*. Ve své podstatě epipolární geometrie kombinuje dva tzv. „*pinhole models*“ a body nazvané *epipóly*. Obrázek 8 v jednoduchosti ukazuje, jak epipolární geometrie vypadá.

Pro každou kameru máme oddělený střed projekce  $O_l$  a  $O_r$ , a jeden pár korespondujících projekčních rovin  $\Pi_l$  a  $\Pi_r$ . Bod  $P$  má projekci fyzicky na každé projekční ploše  $p_l$  a  $p_r$ . Epipól  $e_l$  (resp.  $e_r$ ) je obraz středu kamery (obraz báze) na projekční rovině druhého pohledu v pohledu prvním. Plocha v prostoru, tvořená aktuálním bodem  $P$  a dvěma epipóly  $e_l$  a  $e_r$  se nazývá epipolární rovina (*epipolar plane*) a přímky  $p_l e_l$  a  $p_r e_r$  nazýváme epipolární přímky (*epipolar line*).



Obrázek 8: Epipolární rovina je definována pozorovaným bodem  $P$  a dvěma středy kamery  $O_l$ ,  $O_r$  (bázemi). Epipóly jsou v bodě průniku přímky, spojující středy kamer, s rovinou projekce. [1]

Abychom pochopili užitek epipólů, vzpomeňme si, když vidíme fyzický bod, který promítáme na levou či pravou rovinu promítání, tento bod by mohl být ve skutečnosti umístěný kdekoli podél celé přímky bodů, která jde z  $O_r$  skrze  $p_r$  (nebo  $O_l$  skrze  $p_l$ ), protože právě z jediné kamery nemůžeme znát vzdálenost bodu, na který se díváme. Protože pravá kamera vidí pouze  $p_r$  (promítaný bod  $P$  na  $\Pi_r$ ), skutečný bod  $P$  může být kdekoli na přímce mezi  $p_r$  a  $O_r$ . Tato přímka nejenže obsahuje bod  $P$ , ale taky obsahuje spoustu dalších bodů. Zajímavé je, že obraz všech možných umístění bodu, který vidíme na jedné promítané ploše, je přímka, jdoucí skrze korespondující bod a epipól na druhé promítané ploše.

V bodech epipolární geometrie zahrnuje:

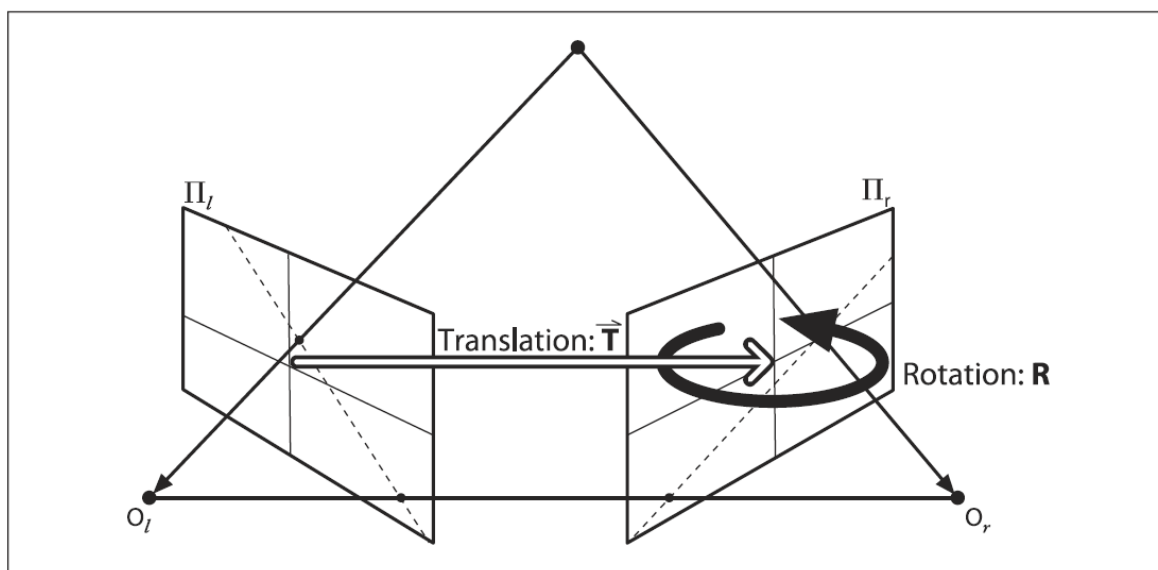
- Každý 3D bod, který kamery vidí, je obsažen v epipolární rovině. Na jednotlivých obrázcích je bod obsažen v odpovídajících si epipolárních přímkách.
- Získaný bod v jednom obrázku musí ležet na korespondující epipolární přímce v druhém obrázku. Tomuto jevu říkáme epipolární vymezení (*epipolar constraint*).
- Epipolární vymezení znamená, že dvoudimenzionální hledání odpovídajících si bodů na dvou obrazech se stává jednodimenzionální hledání po epipolární přímce.

Nejenom to ohromně snižuje náročnost výpočetního hledání, ale můžeme si i dovolit vypustit spoustu bodů, které by mohli vést ke špatným shodám.

- Pořadí nalezených bodů zůstává zachováno. Pokud body  $A$  a  $B$  vidíme na obou obrázcích a nachází se v jedné horizontální řadě na jednom obrázku, pak se taky budou nacházet ve stejném pořadí v jedné horizontální řadě na obrázku druhém. [1]

### 1.1.6 Esenciální a fundamentální matice

Esenciální matice  $E$  obsahuje informace o posunutí a rotaci, vztahujících se fyzicky ke kamerám. Fundamentální matice  $F$  obsahuje stejné informace jako  $E$  a navíc jsou v ní přidány informace o vnitřních parametrech obou kamer. Protože v  $F$  jsou vloženy informace o vnitřních parametrech kamer, spojuje dvě kamery k pixelovým souřadnicím. Obrázek 9 vizuálně ukazuje, co esenciální a fundamentální matice obsahují.



Obrázek 9: Ideální geometrie stereo zobrazování je zachycena esenciální maticí  $E$ , který obsahuje veškeré informace o posunutí  $T$  a rotaci  $R$ . Popisuje vzájemné umístění druhé kamery k první v globálních souřadnicích. [1]

Esenciální matice  $E$  je čistě geometrická a nejsou v ní obsaženy žádné informace o kamerách. Závisí na umístění bodu  $P$  (ve fyzických souřadnicích), dále je označujeme jako  $p_l$  a  $p_r$ . Fundamentální matice naproti tomu závisí na bodech v zobrazené rovině z jedné kamery (v pixelech) k bodům na zobrazené rovině druhé kamery (v pixelech), které označujeme jako  $q_l$  a  $q_r$ . [1]

### 1.1.7 Matematika esenciální matice

Odvodíme vztah, který spojuje zobrazované umístění  $p_l$  a  $p_r$  bodu  $P$  na dvou kamerách.

Pro počítání si vybereme si jedno nastavení souřadnic (levý nebo pravý obraz a kameru). Nezáleží které si přesně vybereme. Pro příklad jsme vybrali souřadnice se středem v  $O_l$  na levé kameře. V těchto souřadnicích je umístění pozorovaného bodu v  $P_l$  a umístění v druhé kameře je  $T$ . Bod  $P$  můžeme pozorovat na pravém obrázku v  $P_r$ , kde

$$P_r = R(P_l - T) \quad (1.2)$$

Důležitým krokem je zavedení epipolární roviny. Pro náš záměr nám pomůže si vzpomenout, že rovnice pro všechny body  $\mathbf{x}$  v rovině s normálovým vektorem  $\mathbf{n}$  a procházející skrze bod  $\mathbf{a}$  se řídí následujícím omezením:

$$(\mathbf{x} - \mathbf{a}) \cdot \mathbf{n} = 0 \quad (1.3)$$

Epipolární rovina zahrnuje vektory  $P_l$  a  $T$ . Normálu  $\mathbf{n}$  z nich získáme pomocí vektorového součinu  $P_l \times T$ . Rovnice pro všechny možné body  $P_l$  přes bod  $T$  bude:

$$(P_l - T)^T (T \times P_l) = 0 \quad (1.4)$$

Vykreslíme  $P_r$  do obrázku pomocí rovnosti  $P_r = R(P_l - T)$ , kterou můžeme přepsat jako  $P_r R^{-1} = (P_l - T)$ . Použijeme substituci  $R^{-1} = R^T$  a dostaneme:

$$(R^T P_r)^T (T \times P_l) = 0 \quad (1.5)$$

V této rovnici můžeme vektorový součin přepsat jako násobení matic:

$$T \times P_l = S P_l \Rightarrow S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \quad (1.6)$$

A dostáváme se k našemu prvnímu výsledku. Pro vektorový součin zavedeme substituci a dostáváme:

$$(P_r)^T R S P_l = 0 \quad (1.7)$$

Součin  $RS$  je právě naše esenciální matice  $E$ , takže naše rovnice bude vypadat takto:

$$(P_r)^T E P_l = 0 \quad (1.8)$$

Přirozeně jsme hledali vztah mezi body  $p_l$  a  $p_r$ , jednoduše můžeme počítat substituce pomocí projekčních rovnic  $p_l = f_l P_l / Z$  a  $p_r = f_r P_r / Z$ . Dosadíme tedy do rovnice a dostaneme:

$$p_r^T E p_l = 0 \quad (1.9)$$

Tato rovnost platí pouze pro přímku (závisí totiž na hodnotě esenciální matice). V esenciální matici je pět parametrů, tři pro otočení (rotaci) a dva pro směr posunutí, spolu s dalšími omezeními.

1. Determinant esenciální matice je 0
2. Dvě jednotlivé nenulové hodnoty jsou shodné, protože matice  $S$  je asymetrická a  $R$  je matice rotace. [1]

### 1.1.8 Matematika fundamentální matice

Esenciální matice  $E$  obsahuje všechny informace o vzájemné geometrii dvou kamer, ale neobsahuje žádné informace o kamerách samotných. Ve skutečnosti se většinou bavíme o souřadnicích pixelů. Abychom našli vztah mezi pixelem v prvním obrázku a korespondující epipolárou v druhém obrázku, musíme dosadit skutečné informace o kamerách. Uděláme to tak, že za  $p$  (souřadnice pixelu) nahradíme  $q$  a přidáme náležitou matici skutečných informací o kamerách  $M$ , dostaneme  $p = M^{-1}q$ . Dosadíme do původní rovnice a dostaneme:

$$q_r^T (M_r^{-1})^T E M_l^{-1} q_l = 0 \quad (1.10)$$

Abychom rovnici zjednodušili, definujeme fundamentální matici  $F$  jako:

$$F = (M_r^{-1})^T E M_l^{-1} \quad (1.11)$$

Po dosazení dostáváme výsledek:

$$q_r^T F q_l = 0 \quad (1.12)$$

Fundamentální matice  $F$  je podobná esenciální matici  $E$ , až na to, že  $F$  operuje v pixelových souřadnicích obrázku, zatímco  $E$  operuje ve fyzických souřadnicích. Obě matice  $E$  i  $F$  mají hodnotu matice 2. Fundamentální matice má 7 parametrů, dva pro každý epipól a tři pro homografii, která náleží dvěma obrázkovým rovinám. [1]

### 1.1.9 Stereo korespondence (stereo matching)

Korespondenci a odpovídající body můžeme hledat pouze v místech, kde se oba dva obrázky překrývají. Pokud známe fyzické souřadnice kamer nebo velikost objektů ve scéně, můžeme určit hloubku pomocí triangulace disparity mezi korespondujícími body ve dvou pohledech.

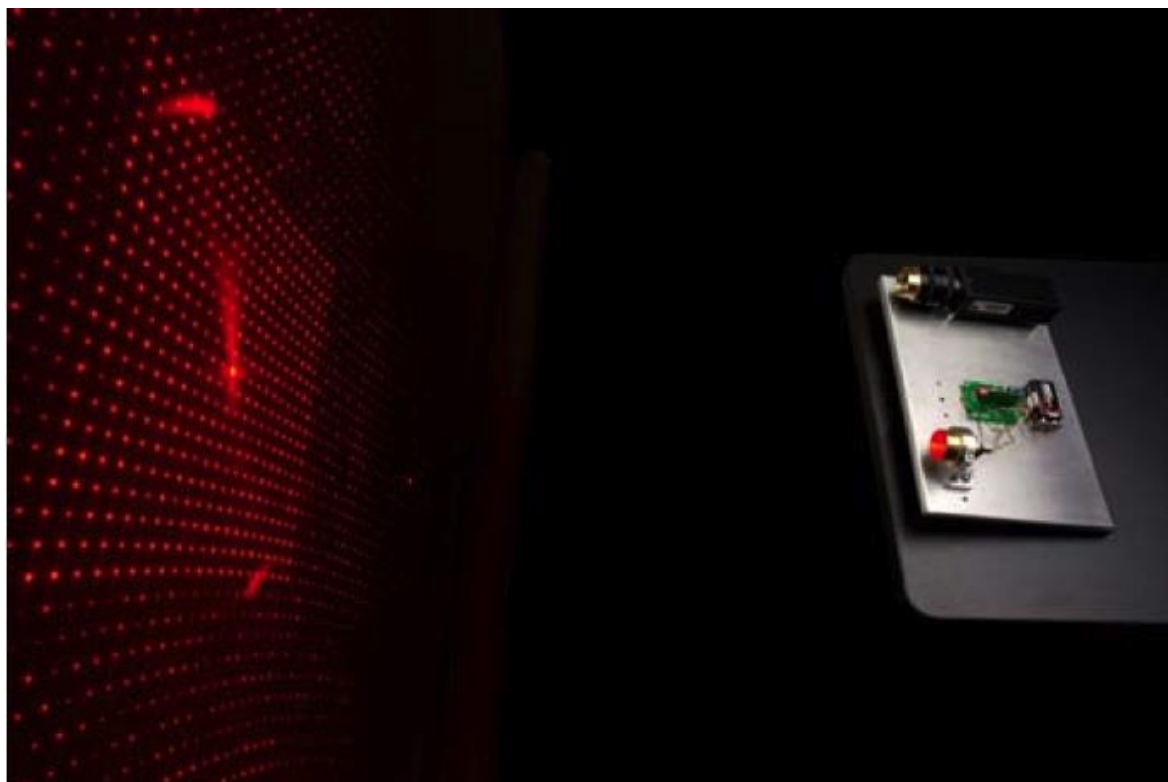
Pokud neznáme fyzické souřadnice, můžeme určit pouze hloubkový poměr. Tomuto procesu se říká *stereo matching*.

#### 1.1.10 Souhrn

U případu stereo kamery nemusíme znát dopředu polohu kamery, přímo ze snímků lze určit relativní umístění kamery vzhledem ke scéně. Taky zde nepotřebujeme znát geometrické uspořádání osvětlení. Pro tyto účely je vhodné vložit do scény kalibrační předmět (předmět známých rozměrů většinou doplněn o definovaný vzor). Tento předmět je pak třeba nalézt v jednotlivých snímcích a z natočení a změny měřítka předmětu jsou určeny všechny potřebné parametry pro měření. [2]

### 1.2 Strukturované světlo

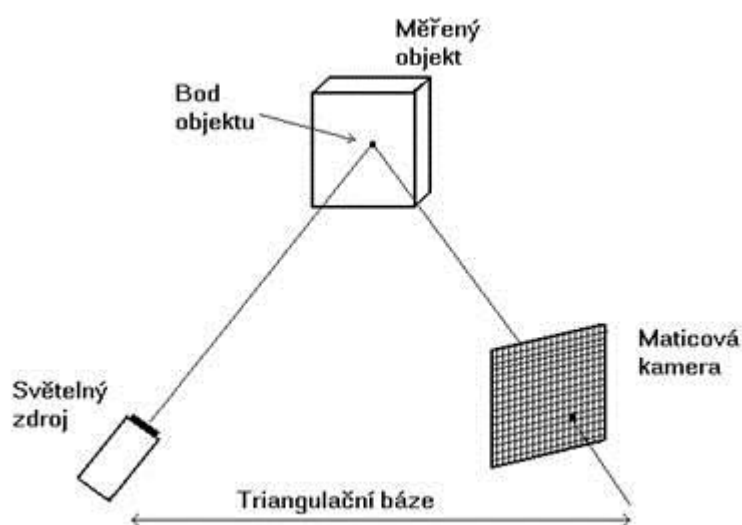
Strukturované světlo je v podstatě světlo, promítnuté přes šterbinu známého tvaru do snímaného prostoru. Obrázek 10 a obrázek 12 ukazuje, jak strukturované světlo může vypadat.



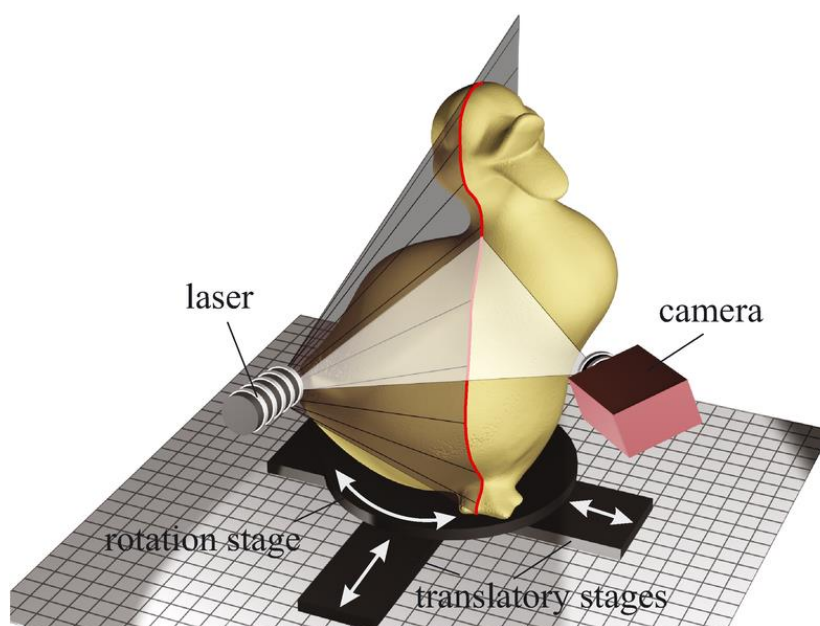
Obrázek 10: Promítané strukturované světlo na plochu [4]



Na obrázku 11 lze vidět princip strukturovaného světla. Využívá se triangulace dvou známých referenčních vektorů. První vektor tvoří paprsek (reference) dobře identifikovatelného úzkého světla, tj. například laserový paprsek. Druhý vektor se získává kalibrovanou kamerou, která hledí na referenční paprsek. [13] Oba vektory jsou kalibrovány a známe jejich vzájemnou polohu, jednoduše pak geometricky vypočítáme a rekonstruujeme 3D souřadnice snímaného povrchu.



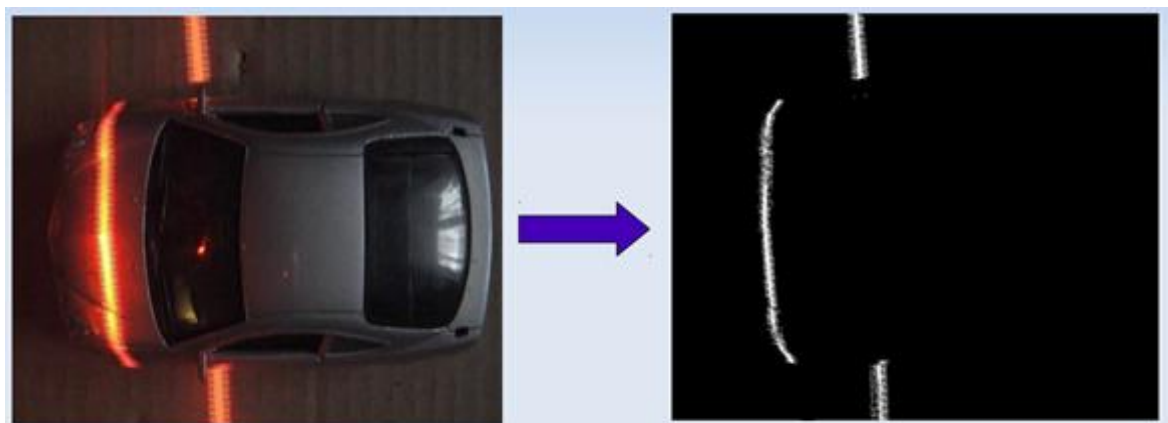
Obrázek 11: Zdroj světla spolu se snímačem a osvětleným bodem na zkoumaném objektu tvoří triangulační trojúhelník, pomocí něhož můžeme vypočítat hloubku [2]



Obrázek 12: Příklad strukturovaného světla [3]

### 1.2.1 Princip strukturovaného světla a 3D triangulace

Jakmile nasvítíme objekt či scénu, vyfiltrujeme obraz kamery tak, abychom viděli právě jenom tohle světlo. Tuto situaci zobrazuje obrázek 13.



Obrázek 13: Příklad filtrace obrázku tak, abychom viděli pouze strukturované světlo [4]

K získání hloubky využíváme geometrii 3D triangulace, kterou můžeme vidět na obrázku 14. Úhel natočení světla od kamery  $\alpha$  a vzdálenost mezi kamerou a světlem  $b$  známe. Víme, že poměry vzdáleností se rovnají, tedy  $\frac{x}{x} = \frac{y}{y} = \frac{z}{z}$ . Pomocí výpočtu pravého trojúhelníku můžeme odvodit, že  $\tan \alpha = \frac{z}{b-x}$ . [5] Můžeme tedy říci:

$$Z = \frac{x}{x} \cdot f = \tan \alpha (b - x) \quad (1.13)$$

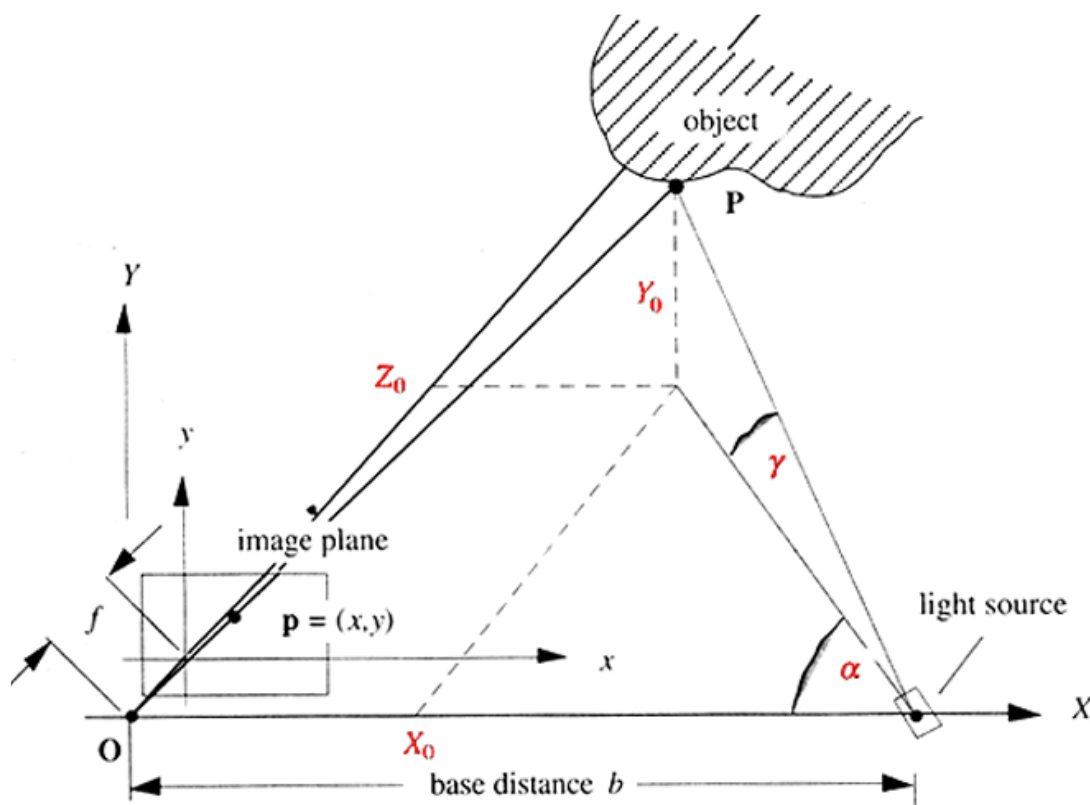
$$x \cdot \left( \frac{f}{x} + \tan \alpha \right) = \tan \alpha \cdot b \quad (1.14)$$

A můžeme počítat jednotlivé souřadnice skutečného bodu  $P$ :

$$X = \frac{\tan \alpha \cdot b \cdot x}{f + x \cdot \tan \alpha} \quad (1.15)$$

$$Y = \frac{\tan \alpha \cdot b \cdot y}{f + x \cdot \tan \alpha} \quad (1.16)$$

$$Z = \frac{\tan \alpha \cdot b \cdot f}{f + x \cdot \tan \alpha} \quad (1.17)$$



Obrázek 14: Geometrie 3D triangulace strukturovaného světla [5]

### 1.2.2 3D rekonstrukce a 3D skener

Na obrázku 15 vidíme příklad kalibrovaného 3D skeneru, který z laseru vysílá strukturované světlo, a na monitoru vidíme výstup z kamery. Tato sestava je připravena k výpočtu hloubky.

Promítneme např. pruh laserového světla na objekt, což vidíme na obrázku 16, a využijeme znalost roviny [5]:

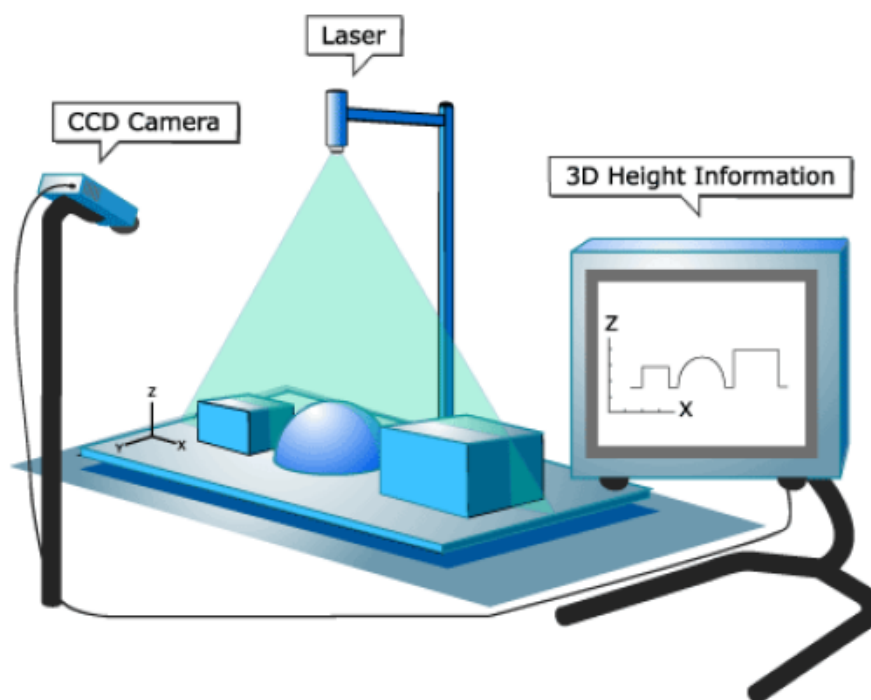
$$AX + BY + CZ + D = 0 \quad (1.18)$$

V příkladu z obrázku 17 můžeme vypočítat souřadnice a tudíž i hloubku bodu:

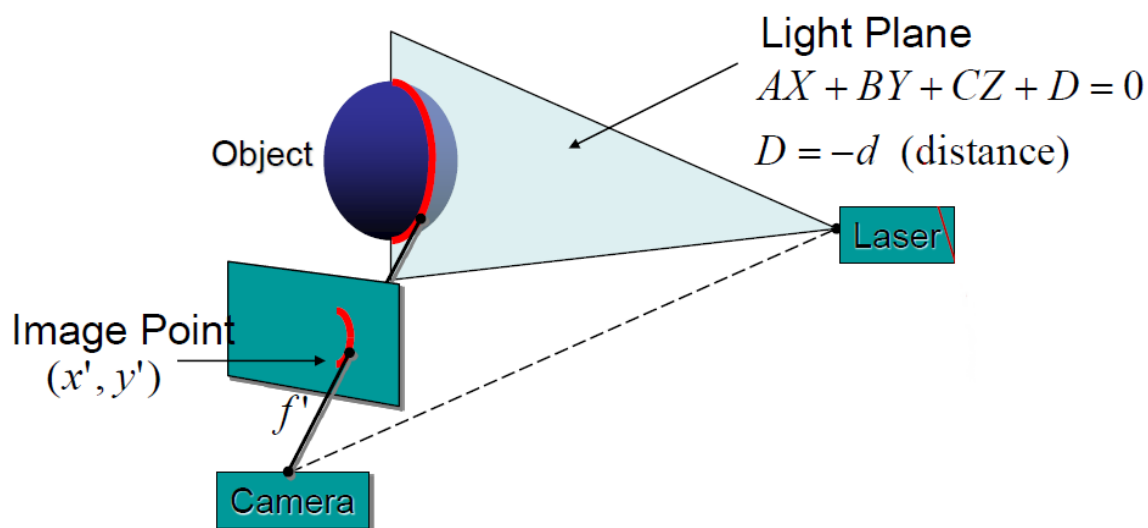
$$X = \frac{x'Z}{f'} \quad (1.19)$$

$$Y = \frac{y'Z}{f'} \quad (1.20)$$

$$Z = \frac{-Df'}{x'X + By' + Cf'} \quad (1.21)$$



Obrázek 15: Geometricky kalibrovaný 3D skener. Na obrazovce monitoru sestavy vidíme vyfiltrovaný obraz strukturovaného světla. [5]

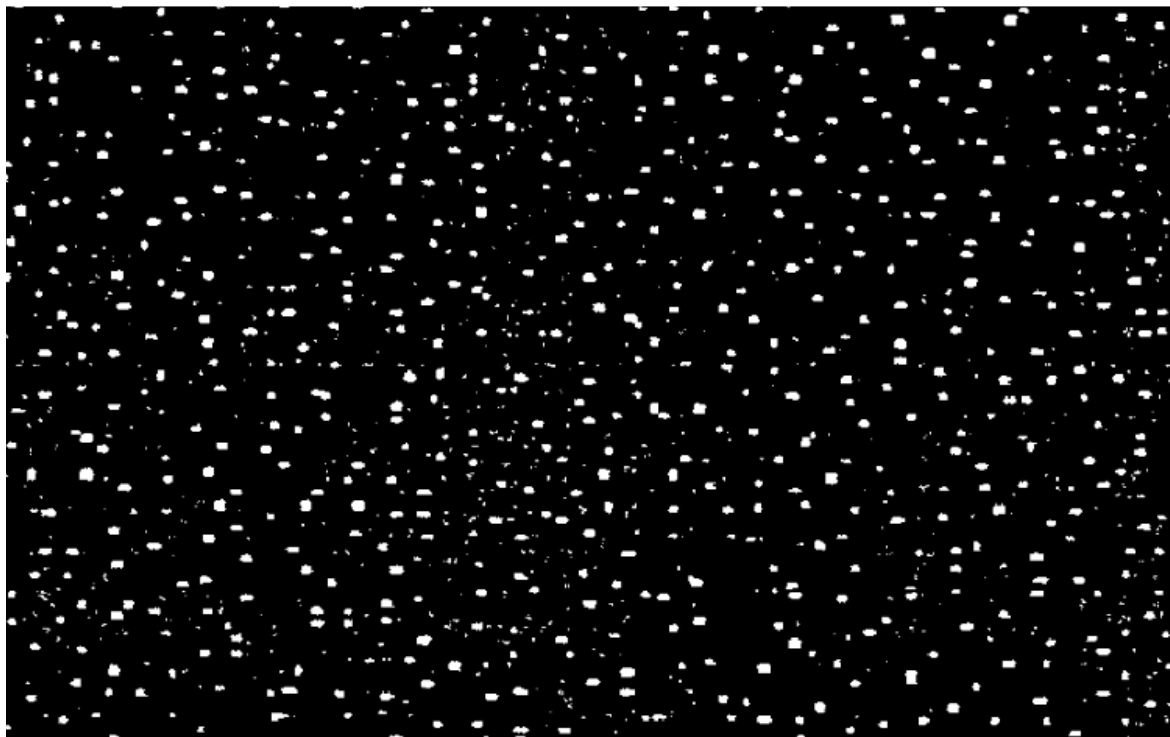


Obrázek 16: Výpočet triangulace. [5]

### 1.2.3 Microsoft Kinect

Kinect kombinuje strukturované světlo s dalšími dvěma technikami. V případě strukturovaného světla Kinect využívá světlo v infračerveném spektru se známým

tečkovaným vzorem (*speckle pattern*). [6] Obrázek 17 zobrazuje vzor tohoto strukturovaného světla.



Obrázek 17: Vzor strukturovaného světla, vysílaný Kinectem. [6]

#### 1.2.4 Souhrn

Strukturované světlo se používá při fotogrammetrické rekonstrukci snímaného objektu nasvícením jeho povrchu světelným zdrojem a současným snímáním CCD snímačem. [2]

Použití strukturovaného světla je vhodné při detekci periodických objektů, nebo objektů bez ostrých jasových přechodů. Poskytuje vcelku kvalitní detekci. Při jednoduchém vzoru mřížky je přesnost malá nebo při opakování posunu vzoru je detekce pomalá, při složitém vzoru se ztrácí „originální“ pohled. Nehodí se na neodrazné plochy (např. černá). Strukturované světlo se musí kalibrovat.

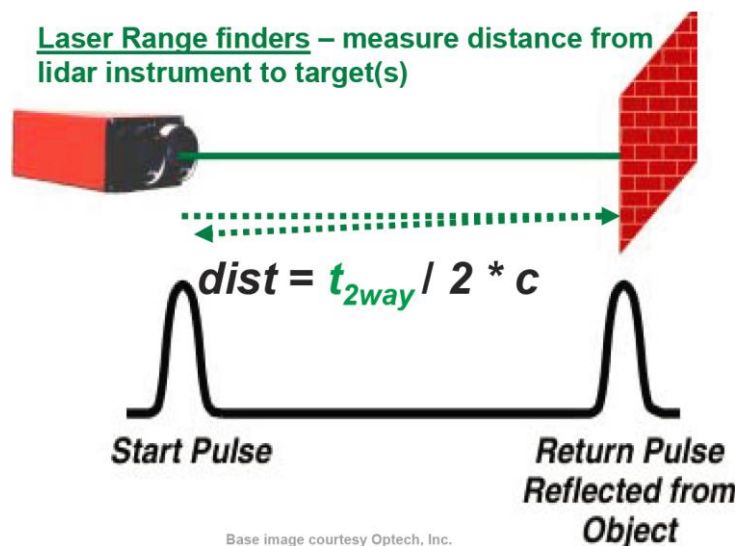
### 1.3 LIDAR (Light Detector And Ranging)

Jedná se o měření vzdálenosti pomocí impulsu laserového paprsku.

Celý přístroj obsahuje zdroj laserového záření, optickou soustavu, detektor elektromagnetického záření (kamera) a velmi přesné hodiny. Hodiny slouží k měření času od vyslání svazku paprsků po jejich detekci na detektoru. Ze znalosti rychlosti šíření světla

lze určit vzdálenost lidarů od objektu. [14] Z této určené vzdálenosti a ze znalosti směru vysílaného svazku paprsků můžeme určit polohu každého měřeného bodu.

### 1.3.1 Princip



Obrázek 18: LIDAR pro měření vzdálenosti [7]

Na obrázku 18 LIDAR do prostoru vyšle krátký optický impuls a měří dobu návratu, která je přímo úměrná dvojnásobné vzdálenosti objektu od lidarů. Vzdálenost  $d$  se vypočítá z poloviny naměřené doby vyslaného optického impulsu  $t$  (v tomto čase urazí impuls vzdálenost dvakrát – tam i zpět), vynásobené rychlostí světla  $c = 299\,792\,458\text{ m/s}$  [7]:

$$vzdálenost = \frac{t}{2} c \quad (1.22)$$

### 1.3.2 Google Car

Firma Google vyvíjí auta bez řidiče již od roku 2005. K mapování okolí a orientaci využívají zabudovaný LIDAR, který můžeme vidět na obrázku 19. Jeho výrobní označení je Velodyne LIDAR HDL-64E a HDL-32E.

Na obrázku 20 je vidět umístění LIDARu na střeše vozidla. LIDAR je tvořen rotačním modulem, díky kterému může kolem sebe rozmítat laserový paprsek. Rotační modul LIDARu umožňuje otáčet se 10 krát za sekundu. Obrázek 21 ukazuje mapování prostoru tímto LIDAREm, který je složen ze 64 nebo 32 laserů. U 64 laserového LiDARu je každý laser posunut o  $26,8^\circ$ , u 32 laserového je to  $40^\circ$ . Díky tomu dosahuje rozlišení 1 cm i na

vzdálenost 80-100m. LIDAR vytváří datový tok 750 MB/s a využívá Ethernetového rozhraní. Velodyne LIDAR se také snaží rozeznat intenzitu odraženého světla a tím lépe rozpoznat objekty. Využití intensity světla zobrazuje obrázek 22. Díky tomu dokáže rozpoznat mlhu nebo kaluže či různou vegetaci.

Mapování LIDAREm také doplňuje mapový software od Googlu a GPS data, která pomáhají s orientací auta na silnici. [26]

Cena použitého LIDARu se aktuálně pohybuje kolem \$70.000. [25]

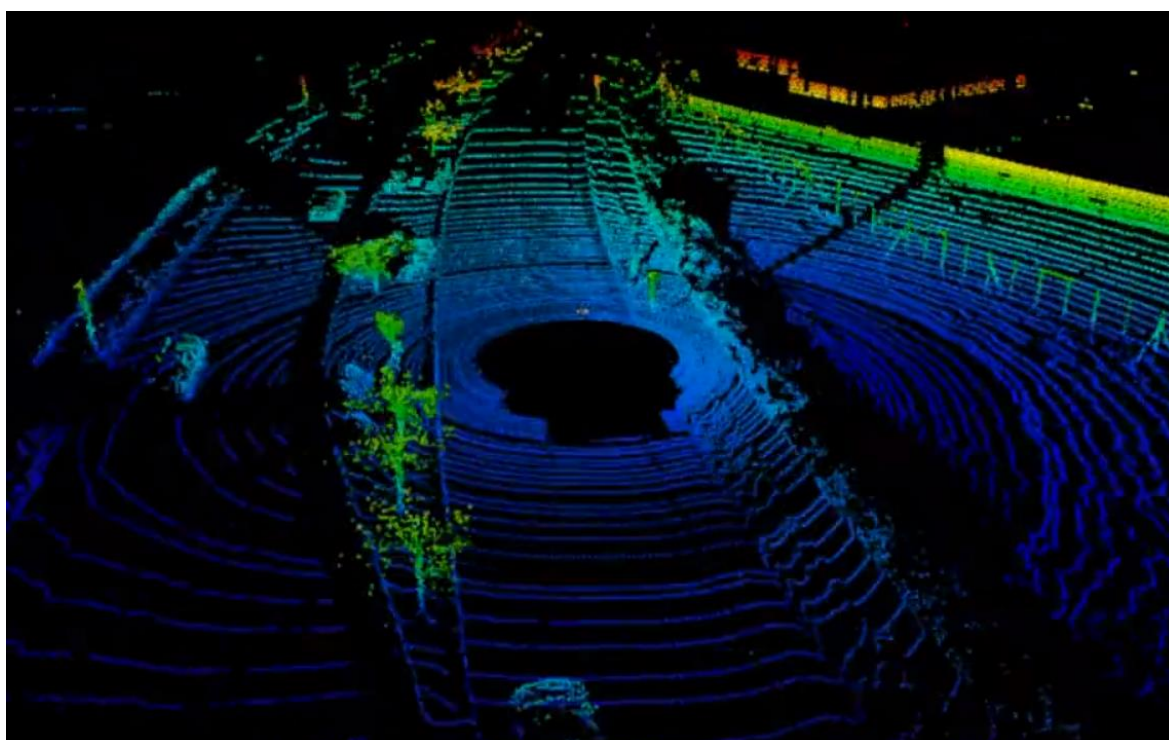


Obrázek 19: LIDAR pro Google Car. [26]





Obrázek 20: Umístění LIDARu na střeše auta Google Car. [27]



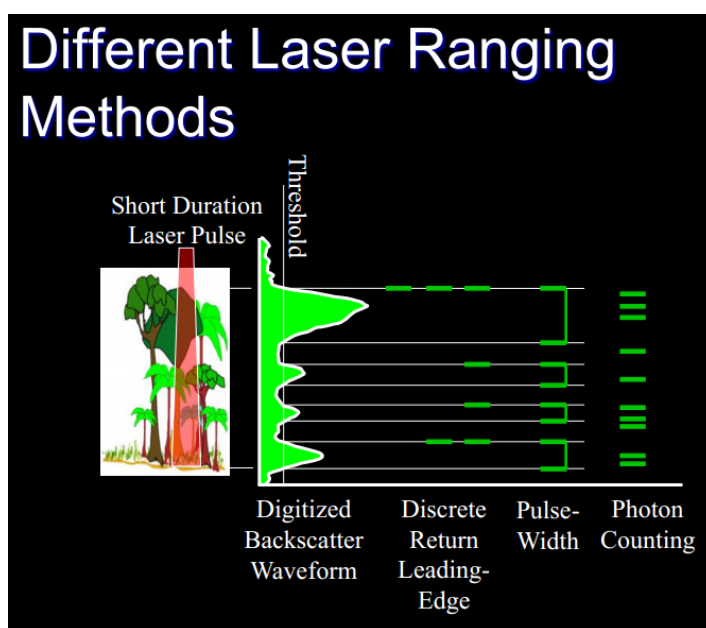
Obrázek 21: Ukázka, jak LIDAR na Google Car mapuje své okolí. [26]



### 1.3.3 Souhrn

LIDAR se využívá v rozmanitých oblastech, pro přesné bezdotykové měření vzdáleností, v meteorologii pro měření srážek, jako radar pro měření rychlosti a hojně se využívá pro mapování velkých ploch a terénu.

Na obrázku 23 je zajímavé praktické využití propustnosti světla na některých objektech. LIDAR díky tomu může detekovat jednotlivé vrstvy např. porostu a vegetace. Přičemž nás zajímá hlavně první a poslední odražený impuls. První nám udává vzdálenost od vegetace a poslední vzdálenost od země.



Obrázek 22: Příklad využití propustnosti světla u lidarů, každá vrstva odráží impuls zpět. [9]

## 2 DETEKCE OBJEKTŮ V OBRAZE

Abychom byli schopni rozpoznat objekt a jeho tvar v obrazové scéně, potřebujeme získat z této scény tzv. kandidáty. Ty posléze podrobíme analýze, zda je kandidát námi hledaným objektem. [15]

Segmentace obrazu jsou metody digitálního zpracování obrazu, které slouží k automatickému rozdělení vlastního obrazu na oblasti se společnými vlastnostmi a které obvykle mají nějaký smysluplný význam. [16] Získáme vzájemně se nepřekrývající oblasti obrazu, které by měli korespondovat s objekty, související s předměty v reálném světě.

### 2.1 Prahování (thresholding)

Je nejjednodušší metoda segmentace obrazu založená na hodnocení jasu každého pixelu. Většinou jde o převod obrazu ve stupních šedi na obraz černobílý. V podstatě jde o to, že se určí hodnota prahu  $T$  a s tímto prahem se porovná každý pixel v obrazu. Pokud je hodnota jasové úrovně kontrolovaného pixelu menší než prahová hodnota, přiřadíme tomuto pixelu novou jasovou hodnotu bílé barvy (většinou 0), pokud je jasová úroveň větší, přiřadíme jasovou hodnotu černé barvy (většinou 255) [10]:

$$f(x, y) = \begin{cases} 255 & \text{pro } g(x, y) \geq T \\ 0 & \text{pro } g(x, y) < T \end{cases} \quad (2.1)$$

Kde  $g(x, y)$  je porovnávaná informace pixelu na pozici  $(x, y)$ .

Můžeme použít i více prahů:

$$f(x, y) = \begin{cases} 1 & \text{pro } g(x, y) \in T_1 \\ 2 & \text{pro } g(x, y) \in T_2 \\ \dots & \\ n & \text{pro } g(x, y) > T_n \\ 0 & \text{jinak} \end{cases} \quad (2.2)$$

Kde  $T_i$  jsou podmnožiny jasových úrovní.

Výsledek prahování zobrazuje obrázek 23.



Obrázek 23: Příklad prahování. [10]

## 2.2 Detekce hran

Hranou myslíme oblast, kde dochází k prudké změně hodnoty jasu. Využívá se první nebo druhé derivace intenzity jasu. K tomuto výpočtu používáme výpočet na bázi gradientu. Popisuje rychlost změny a směr největšího růstu obrazové funkce. [16] Pro zpracování obrazů se používá zjednodušený tvar [15]:

$$\nabla f \approx |G_x| + |G_y| \quad (2.3)$$

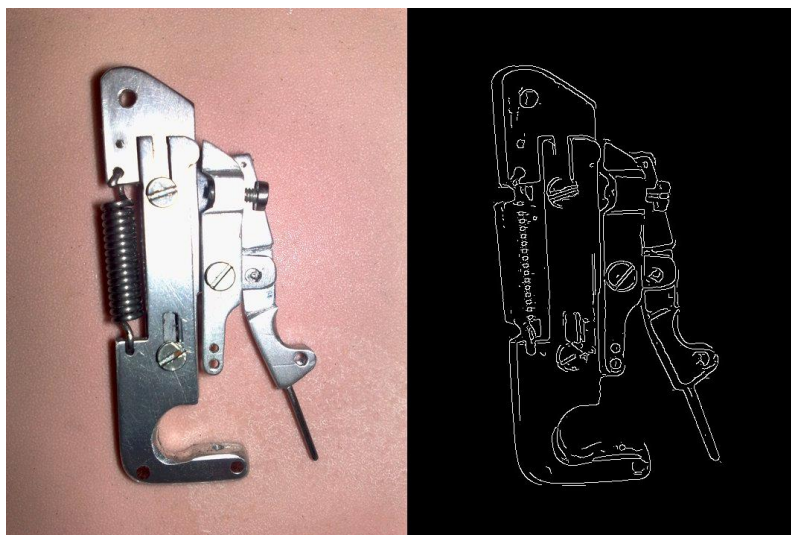
V diskretním obraze se derivace  $G_x$  a  $G_y$  počítají z rozdílu dvou sousedních hodnot jasu pixelů:

$$G_x \approx f(x, +1, y) - f(x, y) \quad (2.4)$$

Popřípadě:

$$G_x \approx f(x, +1, y) - f(x - 1, y) \quad (2.5)$$

Principu gradientu využívají téměř všechny operátory sloužící pro detekci hran. Tyto operátory využívají konvoluční masky. Dle použití masky můžeme detekovat hrany ve vertikálním, horizontálním a diagonálním směru [15]. Výsledek detekce hran ukazuje obrázek 24.



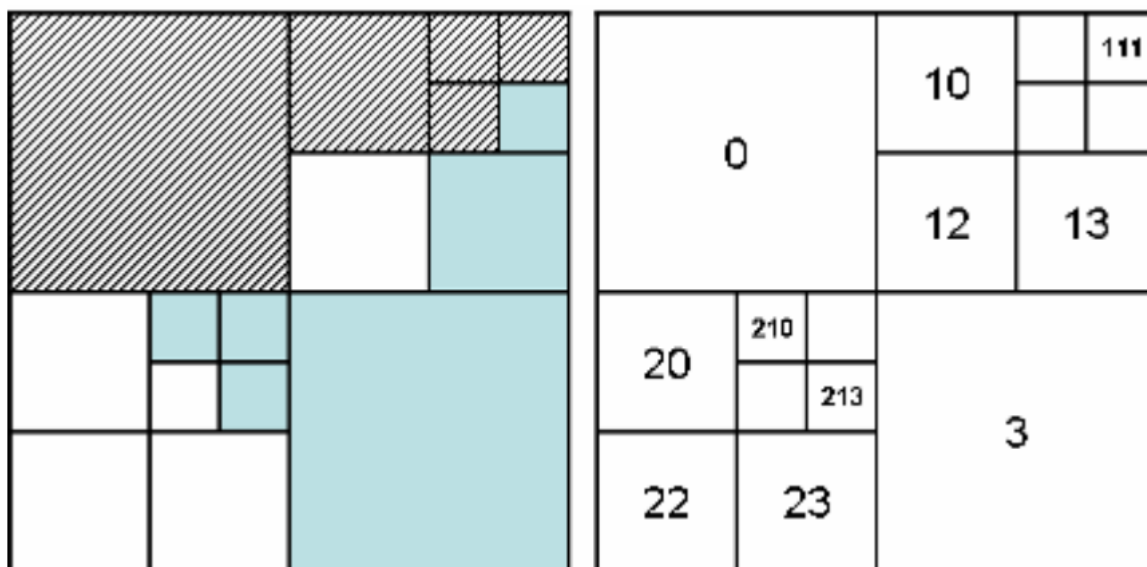
Obrázek 24: Detekce hran. Vlevo původní obrázek, vpravo obrázek po aplikování detekce hran. [11]

Po detekci hran většinou následuje sledování hranice (boundary tracking). Cílem těchto metod je v gradientním obraze najít právě skutečné hrany.

Poslední krok v detekci hran je většinou prahování. Je třeba rozhodnout, jak silná odezva už znamená hranu. Dobré nastavení prahu rozhoduje o kvalitě detektoru, příliš nízká hodnota označí za hrany i šum, příliš vysoká zas zahodí i některé podstatné hrany. [18]

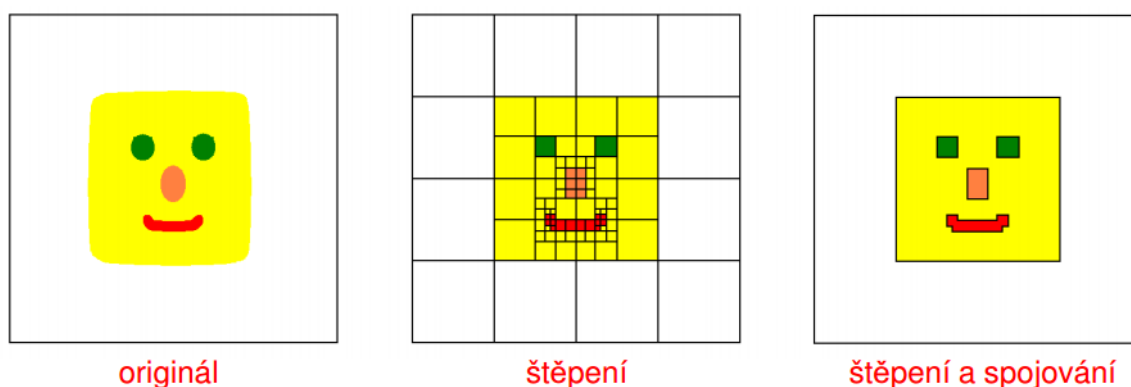
### **2.3 Segmentace založená na hledání oblastí – dělení a spojování oblastí (split and merge)**

Na obrázku 25 je zobrazena segmentace split and merge. Obraz se postupně dělí na menší a menší oblasti do předem dané struktury a sousední oblasti se naopak spojují, pokud splňují určité kritérium homogenity (podobný jas, barva, textura, apod.). Nejčastěji používaná struktura pro správu oblastí je quart tree. Stromová struktura, která se dělí na čtyři nové oblasti při přechodu do další úrovně. [12]



Obrázek 25: Quart tree pro split and marge. [12]

Po segmentaci je každý objekt tvořený souvislou oblastí o stejné hodnotě.



Obrázek 26: Využití segmentace s quart tree. [11]

Používá se k určení skupiny pixelů, které jsou na sobě závislé. Průběh a výsledek této segmentace zobrazuje obrázek 26.

Výhoda segmentace split and merge je mnohem lepší odolnost proti šumu. To znamená, že pro zašuměné obrazy vrací lepší výsledek než detekce hran.

## 2.4 Detekce objektů

V první řadě musíme nejprve rozdělit obraz pomocí segmentace, potom již většinou máme určeny hranice objektu a je jednoduché pak vybrat náš objekt. Můžeme ho vybrat manuálně, nebo pomocí šablony s kterou se pak objekty budou porovnávat, což je nejčastější řešení.

### 3 POHYB A SLEDOVÁNÍ

Jakmile jsme detekovali objekt na jediném obraze, můžeme se zaměřit na jeho sledování ve více obrazech či v určité sekvenci obrazů (videu).

Techniky pro sledování neidentifikovaných objektů se zaměřují spíše než sledování celého objektu, většinou na sledování vizuálně klíčových bodů [1].

#### 3.1 Hledání rohů (corner finding)

Jestliže vybereme v obraze bod, který je unikátní, je docela pravděpodobné, že tento bod najdeme i na druhém obraze. Vybíraný bod by měl vypadat z celého obrázku výjimečně, nebo se výjimečnému přibližovat. Tento bod by měl být tak charakteristický, že by neměl být srovnáván ani s jinými body na jiných obrázcích. Příklad takových charakteristických bodů můžeme vidět na obrázku 27, kde jsou tyto body označeny kolečkem [1].



Obrázek 27: Označené body kolečkem jsou dobré body pro sledování, zatímco body ve čtvercích jsou špatnou volbou, dokonce i když jsou vybrány hranice objektu, je to špatně. [1]

Můžeme se pokusit vyhledat takové body, které mají významnou šanci být zachyceny v obou obrázcích (například hranice objektu), ale ukazuje se, že to nestačí. Body, které mají tuto šanci, mohou být například okraje objektu. Problém je, že tyto okrajové body mohou vypadat jako jakékoli další body podél stejného okraje, které můžeme vidět také na obrázku 26 ve čtvercích.

Pokud jsou odvozené body sledovány ve dvou pravoúhlých směrech, můžeme doufat, že je tento bod pravděpodobně unikátní. Pravoúhle body nazýváme rohy. Rohy jsou body, které obsahují dost informací, aby mohli být převedeny z jednoho obrázku do druhého.

Všeobecně rohy definoval Harris [Harris88] [1]. Spoléhá na matici derivace druhého řádu intenzity obrázku  $(\partial^2 x, \partial^2 y, \partial x, \partial y)$ . Využívá terminologii z Hessovy matice kolem bodu, který je definován ve dvou rozměrech:

$$H(p) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial y \partial x} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \quad (3.1)$$

Pro Harrisovy rohy uvažujeme autokorelační matici druhé derivace obrázku pro malé okno kolem každého bodu. Tato matice je definována:

$$M(x, y) = \begin{bmatrix} \sum_{-K \leq i, j \leq K} w_{i,j} I_x^2(x+i, y+j) & \sum_{-K \leq i, j \leq K} w_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) \\ \sum_{-K \leq i, j \leq K} w_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) & \sum_{-K \leq i, j \leq K} w_{i,j} I_y^2(x+i, y+j) \end{bmatrix} \quad (3.2)$$

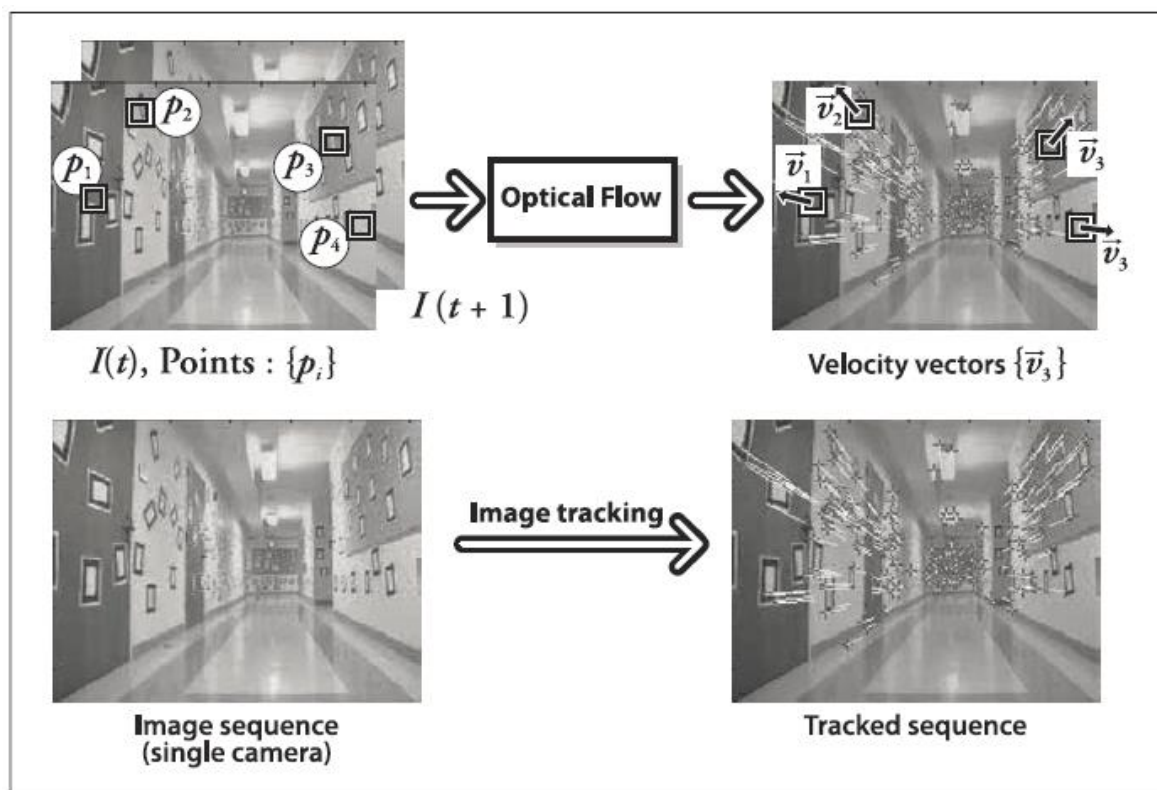
Kde  $w_{i,j}$  je výraz pro váhování, který může být uniformní, ale často se používá na vytvoření kruhového nebo Gaussova váhovacího okna.

Z této definice jsou rohy umístěné v obrázku tam, kde autokorelační matice druhé derivace má dvě velké vlastní hodnoty. V podstatě to znamená, že tam, kde se setkává jedna textura (nebo hrana) jdoucí nejméně ve dvou různých směrech, je roh. Využíváme druhé derivace, protože neodpovídají homogenním gradientům. To znamená, že když je první derivace homogenní (konstantní), pak druhá derivace se rovná 0. Výhoda je v tom, že bereme v úvahu vlastní čísla autokorelační matice, zvažujeme množství, které je neměnné při otáčení, což je důležité, protože sledované objekty se mohou při pohybu otáčet. Pozorujeme taky, že dvě vlastní čísla nám nejenom určují, zda je bod dobrý ke sledování, ale taky poskytuje identifikaci označení bodu.

Harrisova původní definice zahrnuje výpočet determinantu  $H(p)$ , vypočítává dráhu  $H(p)$  (s určitým váhovým koeficientem) a pak porovnává tyto rozdíly k určení možného prahu. [1]

### 3.2 Pohyb optiky

V případě, že se pohybuje optika, většinou chceme odhadnout pohyb mezi dvěma obrázky (nebo sekvence obrázků) bez předchozích znalostí obsahu této scény (objektů, atd.).



Obrázek 28: Pohyb optiky a sledování. [1]

Detekci pohybu optiky zobrazuje obrázek 28. Můžeme spojit rychlost s každým pixelem v obraze nebo posunutí, které reprezentuje vzdálenost pixelů, které se přesunuly mezi předchozím a nynějším obrazem. Takové provedení je obvykle uváděné jako *dense optical flow*, kterému můžeme připojit rychlost každému pixelu v obrázku. Pokusu porovnat plochu kolem každého pixelu jednoho obrazu s následujícím obrazem se říká porovnávání bloků (*block matching*).

Druhý přístup se nazývá *sparse optical flow*. Tyto algoritmy se přirozeně spoléhají na průměry specifických předem podmnožiny bodů, které mají být sledovány. Pokud tyto body mají určité žádoucí vlastnosti – „rohy“ [1].



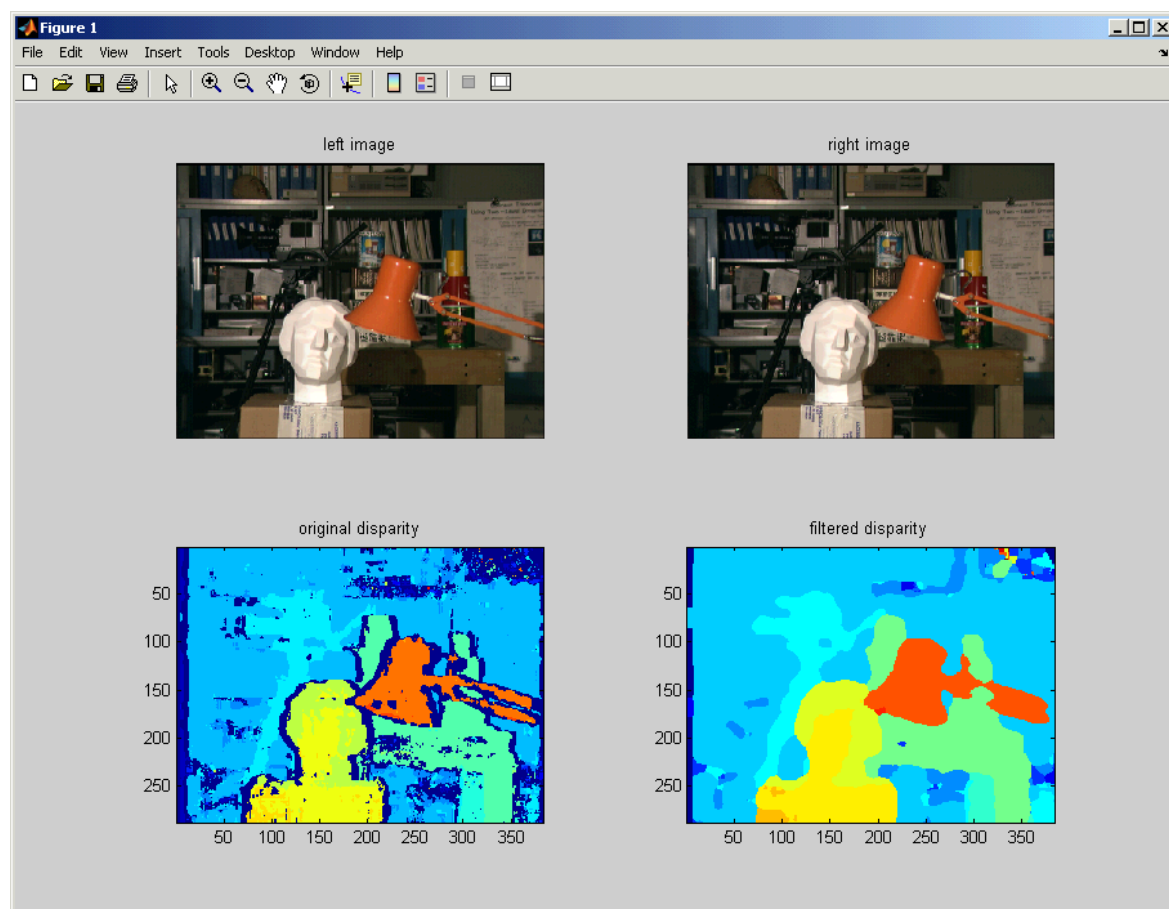
## **II. PRAKTICKÁ ČÁST**

## 4 MAPA STEREO DISPARITY – PŘÍKLADOVÉ DEMO V MATLABU

Podívejme se na příklad stereo vidění a získání hloubkové mapy v programu, napsaném v Matlabu. [20]

### 4.1 Zobrazení a základní funkcionalita

Základní zobrazení je definováno velmi jednoduše (Obrázek 29). Jakmile spustíme příkladové demo, zobrazí se nám čtyři obrázky. Nahoře vlevo je zarovnaný levý obrázek ze stereo páru. Nahoře vpravo je pravý obrázek ze stereo páru. Dolu vlevo je zobrazen obrázek s vypočítanou mapou disparity, což je matice o velikosti obrázku ze stereo páru, jejíž hodnoty jsou odhadnuté vzdálenosti hloubky jednotlivých pixelů. Jednotlivé barvy ukazují odhadnutou hloubku. Přičemž tmavě červená, představující nejbližší předměty, přechází plynule do tmavě modré, která představuje nejvzdálenější předměty. Vpravo dole se mapa disparity vypočítává za pomoci filtrační funkce z knihovny v c++, což dává lepší výsledek.



Obrázek 29: Vzhled okna v příkladovém demu na výpočet disparity.

## 4.2 Implementace

### 4.2.1 Vstupy a GUI

```
%% Run this to demo stereo disparity extraction

i1 = imread('tsuR.png'); %right image
i2 = imread('tsuL.png'); %left image

maxs = 20; %maximum disparity between the two images

%-- here's the main call
[d p] = stereo(i1,i2, maxs);

%-- run this instead if filtering causes problems
%   d = stereo_nofilter(i1,i2, maxs); p = d;

%-- Display stuff
subplot(2,2,1), imshow(i2); title('left image');
subplot(2,2,2), imshow(i1); title('right image');
subplot(2,2,3), imagesc(p); title('original disparity'); axis image;
subplot(2,2,4), imagesc(d); title('filtered disparity'); axis image;
```

Obrázek 30: Základní program [20]

Na začátku se pomocí `imread()` načtou obrázky do paměti, určí se maximální disparita `maxs`, kterou budeme brát v úvahu. Samotný výpočet a funkcionality se provádí ve funkci `stereo()`. Vstupy jsou obrázky ze stereo páru a maximální disparita. Výstup této funkce je obrazová mapa disparity `p` a obrazová mapa disparity s použitím gradientních filtrů `d`.

Následně se stereo pár a výsledné mapy disparity nechají zobrazit v okně.

Pozn.: Matlab používá `i1`, `i2` obrázky jako matice ve tvaru ( $obrazek_{výška} \times obrazek_{šířka} \times obrazek_{RGB}$ ), tzn. v našem příkladu ( $288 \times 384 \times 3$ ).

#### 4.2.2 Výpočet disparity, funkce stereo()

```
function [fdsp dsp] = stereo(i1,i2, maxs)
    win_size = 7; %-- size of window used when smoothing
    tolerance = 2; %-- how close R-L and L-R values need to be
    weight = 5; %-- weight on gradients opposed to color

    %--determine pixel correspondence Right-to-Left and Left-to-Right
    [dsp1, diff1] = slide_images(i1,i2, 1, maxs, win_size, weight);
    [dsp2, diff2] = slide_images(i2,i1, -1, -maxs, win_size, weight);

    %--keep only high-confidence pixels
    dsp = winner_take_all(dsp1,diff1,dsp2,diff2,tolerance);

    %--try to eliminate bad pixels
    fdsp = modefilt2(dsp,[win_size,win_size],2);
```

Obrázek 31: Funkce stereo() - výpočet mapy disparity

Funkce `stereo()` přebírá tři parametry. Levý obrázek ze stereo páru `i1`, pravý obrázek ze stereo páru `i2` a hodnotu maximální disparity `maxs`, která se bude hledat. Vrací mapu disparity `dsp` a mapu disparity, na kterou je použitý filtr pro lepší vyhodnocení disparity `fdsp`.

#### 4.2.3 Funkce slide\_images()

Nejprve najdeme kandidáty na korespondující body pomocí funkce `slide_images()`. Potom vybereme takové, které jsou nejvhodnější, a eliminujeme vadné body.

```
function [disparity mindiff] = slide_images(i1,i2,mins,maxs,win_size,weight)
    [dimy,dimx,c] = size(i1);
    disparity = zeros(dimy,dimx); %-- init outputs
    mindiff = inf(dimy,dimx);

    h = ones(win_size)/win_size.^2; %-- averaging filter

    [g1x g1y g1z] = gradient(double(i1)); %-- get gradient for each image
    [g2x g2y g2z] = gradient(double(i2));

    step = sign(maxs-mins); %-- adjusts to reverse slide

    for i=mins:step:maxs
        s = shift_image(i2,i);
        sx = shift_image(g2x,i);
        sy = shift_image(g2y,i);
        sz = shift_image(g2z,i);
```

Obrázek 32: Funkce slide\_images() – 1.část

```

%--CSAD is Cost from Sum of Absolute Differences
%--CGRAD is Cost from Gradient of Absolute Differences
diffs = sum(abs(i1-s),3);           %-- get CSAD and CGRAD

gdifffx = sum(abs(g1x-sx),3);
gdifffy = sum(abs(g1y-sy),3);
gdifffz = sum(abs(g1z-sz),3);
gdifff = gdifffx+gdifffy+gdifffz;

CSAD = imfilter(diffs,h);
CGRAD = imfilter(gdifff,h);
d = CSAD+weight*CGRAD;

idx = find(d<mindiff);              %-- put corresponding disparity
disparity(idx) = abs(i);            %   into correct place in image
mindiff(idx) = d(idx);
end

```

Obrázek 33: Funkce `slide_images()` – 2.část

Funkce `slide_images()` posouvá obrázky přes sebe a tak porovnává disparitu. Vrací odhadnutou disparitu `disparity` a matici minimálních rozdílů, na kterou použil gradientní filtr `mindiff`. Jako vstupy jsou předávány levý `i1` a pravý `i2` stereo obrázek, minimální `mins` a maximální `maxs` disparita, velikost filtrujícího okna `win_size`, které obrázky zjemní (rozmaže) a váhu pro gradientní filtr `weight`.

Ve smyčce `for` se vodorovně posouvá druhý obrázek ze stereo páru a jeho gradienty `shift_image()` o `i` pixelů (sloupců matice). V dalším kroku si do `diffs` vloží součet všech barev z rozdílu mezi prvním obrázkem ze stereo páru `i1` a právě posunutým druhým obrázkem. Totéž udělá pro rozdíl gradientů. Výsledné rozdíly gradientů ve všech směrech sečte do jedné matice `gdifff`.

Funkcí `imfilter(diffs,h)` na obrázky použije filtr `h`, což je matice velikosti `win_size` s hodnotami všech polí  $\frac{1}{win\_size^2}$ , pro `win_size = 7` je to matice  $7 \times 7$  s hodnotami 0,0204. Jednoduše to obrázek rozmaže. Následně do `d` přiřadí součet takto vyfiltrovaných obrázků – jak rozmazaný `diffs`, tak rozmazaný `gdifff`, vynásobený váhou `weight`.

Následně si do `idx` zapíše, v kterém poli platí podmínka `d<mindiff`. Na začátku, před smyčkou, je `mindiff` definováno jako matice velikosti obrázku s hodnotami nekonečno.

Tyto pole potom využije, když do mapy disparity na těchto pozicích zapíše hodnotu  $i$ , tedy velikost posunu:

```
disparity(idx) = abs(i);
```

Do `mindiff(idx)` pak vloží do stejných polí vyfiltrovaného a posunutého obrázku `d(idx)`.

Tato smyčka projede tolikrát, kolik je maximální hodnota disparity. Výsledné hodnoty disparity pixelů pak vrátí v matici `disparity`. Číselné hodnoty rozdílů pak vrátí v `mindiff`.

Funkci `slide_images()` vykoná program dvakrát. Jednou pro překrytí pravého obrázku přes levý a podruhé levého obrázku přes pravý. Získá tak dvojce matice disparity `disp1`, `disp2` a dvojce matice rozdílů `diff1`, `diff2`.

#### 4.2.4 Funkce `winner_take_all()`

```
function pd = winner_take_all(d1,m1,d2,m2,tolerance,maxs)
    pixel_dsp = zeros(size(d1));           %-- initialize output
    idx1 = find(abs(d1-d2)<tolerance & m1<m2); %-- find where d1 is best
    idx2 = find(abs(d1-d2)<tolerance & m2<m1); %-- find where d2 is best
    pixel_dsp(idx1) = d1(idx1);             %-- fill with d1
    pixel_dsp(idx2) = d2(idx2);             %-- fill with d2
    pd = shift_image(pixel_dsp,5);          %-- shift to match i1
```

Obrázek 34: Funkce `winner_take_all`

Vrací optimalizovanou mapu disparity `pd`. Přebírá parametry získaných map disparity `d1`, `d2` a rozdílových map `m1`, `m2`, které jsme získali ze `slide_image()` a určitou toleranci `tolerance`.

Nejprve si připraví matici `pixel_dsp` o velikosti získané mapy disparity `d1`, kterou naplní nulami. Do `idx1` vloží pozice, kde rozdíl v mapách disparity je menší než `tolerance` `abs(d1-d2)<tolerance` a zároveň, kde první rozdílová mapa je menší, než druhá rozdílová mapa `m1<m2`. Do `idx2` vloží podobné pozice s rozdílem důležitosti rozdílových map. Do `pixel_dsp` na pozici `idx1` vloží hodnoty z první mapy disparity `d1`, a na pozici `idx2` vloží hodnoty z druhé mapy disparity `d2`. Funkce pak vrátí výslednou mapu disparity `pixel_dsp`, posunutou o 5 pixelů doprava.

#### 4.2.5 Funkce modefilt2()

```
function f = modefilt2(img,win,ignore)

%-- check input
if(~exist('ignore')) ignore = 0; end
if(~exist('win')) win = [5 5]; end
if(~isfloat(img)) img = double(img); end

%-- this could be removed if you are sure you're
%   passing whole, positive numbers
img = abs(round(img));

f = modefilt2_mex(img,win,ignore);
```

Obrázek 35: Funkce modefilt2(),

Funkce volá kód v c++ modefilt2\_mex, která porovnává každý pixel obrázku se sousedními. Oblast porovnávání je dána rozměry win. ignore udává, jakou minimální vzdálenost v oblasti porovnávání má ignorovat. Funkce vrací vyfiltrovanou 2D matici disparity.

Tento příklad pochází od autora: Shawn Lankton, April 2008 [20].

## 5 OPTICAL FLOW PŘÍKLADOVÉ DEMO – KNIHOVNA OPENCV

Knihovna OpenCV je koncipována jako volná univerzální infrastruktura počítačového vidění, kterou podporuje Intel's Performance Library Team. OpenCV spadá pod volnou licenci BSD. Knihovna je psaná v C,C++, lze ji najít na <http://opencv.org/> [20].

OpenCV obsahuje třídy a funkce pro práci se *Stereo correspondence*, *View point morphing of cameras*, *3D tracking in stereo*, atd.. OpenCV také poskytuje oba dva hlavní přístupy pro sledování trasy jak pro *Dense Tracking Techniques*, tak pro *Sparse Tracking Techniques*. *Dense Tracking Techniques* jsou náročnější na počítání, protože můžeme připojit určitou rychlost každému pixelu v obraze nebo posunutí, které reprezentuje vzdálenost, o kterou se pixel posunul od předcházejícího obrazu. Naproti tomu *Sparse Tracking Techniques*, kde se algoritmus přirozeně spoléhá na určitou podmnožinu bodů, kterou si vytipoval předem. Když mají tyto body určité žádoucí vlastnosti, jako například „rohy“, potom je počítání těchto bodů poměrně spolehlivé a méně nákladné na výpočetní výkon.

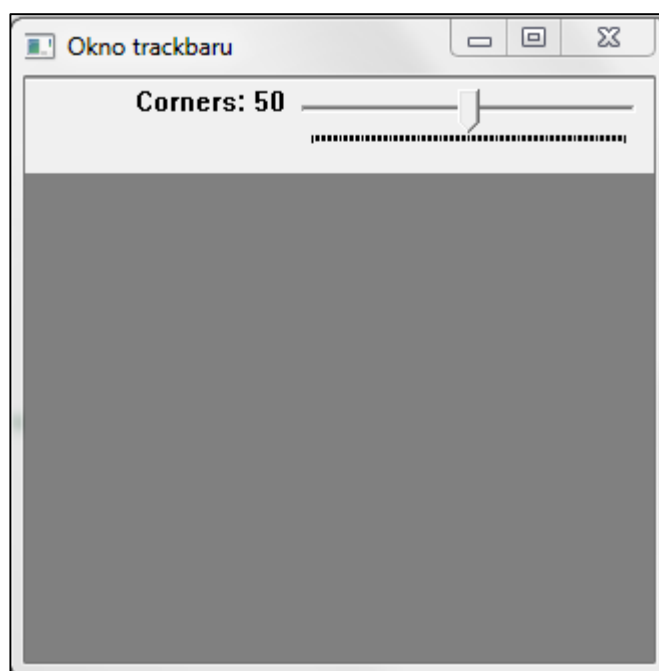
### 5.1 Funkcionalita a zobrazení

Příkladový program pro je rozdělen do dvou oken. Jedno okno zobrazuje záznam z kamery (Obrázek 36) a druhé posuvník, kterým nastavujeme, kolik vhodných bodů pro sledování bude v obrázcích hledat (Obrázek 37). Pokud se nastaví, že se bude hledat





Obrázek 36: Okno, které zobrazuje obrázek



Obrázek 37: Okno s trackbarem

Program zobrazuje vypočítané vhodné body pro sledování a jejich posun od posledního referenčního obrázku. Referenční obrázek se obnovuje po určitém intervalu.

## 5.2 Implementace

```
36 int main(int argc, char** argv) {
37
38     VideoCapture cap(0); // otevře defaultní kameru
39     if (!cap.isOpened()) // test, zda je kamera úspěšně otevřená
40         return -1;
41     createTrackBarsWindow(); // připraví okno s trackbarem
42     namedWindow("OpticalFlow", CV_WINDOW_AUTOSIZE);
43
44     Mat imageCapture;
45     cap >> imageCapture; // získá obrázek z kamery
46     Mat imageReferencni;
47     Mat imageAktualni;
48     cvtColor(imageCapture, imageReferencni, CV_BGR2GRAY);
49
50     int loop = 0; // abych mohl určit po jaké době budu obnovovat referenční obrázek
```

Obrázek 38: VanharaOpticalFlow.cpp – 1.část

Na začátku se připraví kamera, otestuje se její funkčnost, připraví se okna a inicializují se třídy Mat, které nesou informaci obrázků.

```
52     while (1) {
53
54         cap >> imageCapture;
55         cvtColor(imageCapture, imageAktualni, CV_BGR2GRAY);
56
57         // počká přibližně 450 ms
58         if (loop > 15) {
59             cap >> imageCapture;
60             cvtColor(imageCapture, imageReferencni, CV_BGR2GRAY);
61             loop = 0;
62         }
63
64         vector<Point2f> corners1;
65         corners1.reserve(CORNERS);
66         vector<Point2f> corners2;
67         corners2.reserve(CORNERS);
68         vector<uchar> nalezene_body;
69         nalezene_body.reserve(CORNERS);
70         vector<float> chyby;
71         chyby.reserve(CORNERS);
72
73         // zjištění možných bodů, které by měli být hledány
74         goodFeaturesToTrack(imageReferencni, corners1, CORNERS, UROVEN_KVALITY,
75                             MIN_DISTANCE, Mat());
76         goodFeaturesToTrack(imageAktualni, corners2, CORNERS, UROVEN_KVALITY,
77                             MIN_DISTANCE, Mat());
```

Obrázek 39: VanharaOpticalFlow.cpp – 2.část

```

79 // ze získaných kandidátů vybere takové, které se podobají rohům
80 cornerSubPix(imageReferencni, corners1, Size(WIN_SIZE, WIN_SIZE), Size(-1, -1),
81 TermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 20, 0.03));
82
83 cornerSubPix(imageAktualni, corners2, Size(WIN_SIZE, WIN_SIZE), Size(-1, -1),
84 TermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 20, 0.03));
85
86 // zjistí odpovídající si body
87 calcOpticalFlowPyrLK(imageReferencni, imageAktualni, corners1, corners2, nalezene_body,
88 chyby, Size(WIN_SIZE, WIN_SIZE), 5,
89 cvTermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 20, 0.3), 0);
90
91 // Do obrázku vykreslí přímky, kde byli sledované body nalezeny
92 // Abych rozeznal směr, koncový bod je označen kružnicí
93 Mat image3(imageReferencni.size(), 1);
94 imageCapture.copyTo(image3);
95 for (int i = 0; i < (int)nalezene_body.size(); i++) {
96     Point p0(ceil(corners1[i].x), ceil(corners1[i].y));
97     Point p1(ceil(corners2[i].x), ceil(corners2[i].y));
98     circle(image3, p1, 5, CV_RGB(0,0,255), 1, 2, 0);
99     line(image3, p0, p1, CV_RGB(255,255,255), 1);
100 }
101
102 imshow("OpticalFlow", image3);
103
104 loop++;
105 if (waitKey(30) >= 0)
106     break;
107 }
108 destroyAllWindows();
109
110 return 0;
111 }

```

Obrázek 40: VanharaOpticalFlow.cpp – 3.část

Ve smyčce se vždy sbírá aktuální obrázek z kamery, který se přefiltruje na obrázek ve stupních šedi. Referenční obrázek se sbírá až po 15. projetí smyčky. Pomocí funkce *goodFeaturesToTrack*, z knihovny OpenCV, se předběžně najdou vhodné body, které by mohly být použity pro hledání na dalších snímcích. Prozkoumáme okolí těchto bodů a určíme, které body mají charakteristiku „rohů“ a jsou tudíž jedinečné – *cornerSubPix*.

Potom pomocí Lucas-Kanade metody najde odpovídající si body v obrázcích. [20] – *calcOpticalFlowPyrLK*.

Nakonec do aktuálního obrázku vykreslí přímku mezi bodem v referenčním obrázku a stejným bodem, nalezeným na aktuálním obrázku. Abychom poznali směr na první pohled, v bodě, nalezeném na aktuálním obrázku, se vykreslí kružnice.

Program se dá zastavit stisknutím jakékoli klávesy.

## ZÁVĚR

Oblast počítačového vidění je rozsáhlé téma a obsahuje mnoho oblastí.

Nejdůležitějším a základním úkolem počítačového vidění je ze získaných obrazů scény rozpoznat jednotlivé objekty. Rozpoznání objektů zahrnuje úpravy obrazu. Metody úpravy obrazu pro rozpoznání objektů zahrnují filtrace jako např. prahování, detekce hran a segmentace. Tyto úpravy mají za cíl naprosto rozlišit objekt od okolí tak, že bude naprosto jasné, kde má objekt hranice a kde objekt není. Například se obraz vyfiltruje tak, že objekt bude bílé barvy a zbytek obrazu černé. Potom už je naprosto jednoduché zjistit, kde se objekt nachází.

Dalším úkolem počítačového vidění je sledování rozpoznaných objektů ve sledu obrazových scén. Jinak řečeno nalezení objektu z jednoho obrazu v druhém obrazu. Využívá se hledání unikátních oblastí v obraze (tzv. „rohů“). Tato oblast je tak unikátní, že je v obraze pouze jedna. Když se potom podíváme na další obraz, zjistíme, kde se tato oblast vyskytuje, a můžeme pak zjistit, kam se objekt posunul.

Počítačové vidění také zahrnuje mapování prostoru a objektů v prostoru. Stereo kamera využívá dvou a více kamer pro zjištění vzdálenosti získaných bodů od sebe. Díky triangulaci a epipolární geometrii jsme do jisté míry schopni vypočítat 3D pozici bodu. Strukturované světlo využívá promítanou mřížku určitého vzoru, kterou nasvítí scénu a kamera citlivá právě na tuto vlnovou délku lehce odfiltruje šum a následně z deformovaného vzoru za pomoci triangulace můžeme vypočítat 3D souřadnice bodu. LiDAR naproti tomu využije laserového impulsu a počítá čas, za jak dlouho se odražený impuls detekuje na detektoru (kameře). Potom díky znalosti šíření světla dokážeme vypočítat vzdálenost bodu.

Díky zmapování prostoru a zjištění objektů se můžeme pokusit počítač naučit pohybovat v tomto prostoru. Právě na toto téma se chci zaměřit v další práci.

## ZÁVĚR V ANGLIČTINĚ

The section of computer vision is an extensive theme and it includes many sections.

The most important method and basic task of computer vision is to recognize individual objects from the obtained image of the scene. Methods of image processing to detect objects include filtration for example, thresholding, edge detection and segmentation. These filtration are intended to completely distinguish an object from its surroundings so that it will be absolutely clear, where the object has borders and where the object is not. For example, the image is filtered such that the object should be white and the rest of image black. After this process it is easy to find where the object is located.

Another task of computer vision is tracking of detected objects in a sequence of image scenes. In other words, find an object from one image to another image. It uses looking for unique area in the image (means “corners”). This area is so unique that it is in the frame only once. When then we look on other images, we find out, where this area exist and we can then find out where the object is moved.

Computer vision includes mapping of the space and objects in the space. Stereo camera uses two and or more cameras to determine the distance of earning points from each. Through triangulation and epipolar geometry we are pretty able to calculate a 3D position of the point. Structured light uses projected grid of a certain pattern. The scene is illuminated a camera, which is wavelength sensitive, will lightly filter out the noise and then we are able to calculate the 3D coordinates of the point from the deformed pattern by using triangulation. In contrast, LIDAR uses a laser impulse and it calculates the time for how long the reflected pulse is detected at the detector (camera). Then, through propagation of light knowledge we are able to calculate the distance of the point.

Through mapping of the space and finding objects we can try to teach the computer how to move in this area. I want to focus on this theme in the other work.

## SEZNAM POUŽITÉ LITERATURY

- [1] BRADSKI, Gary R. *Learning OpenCV*. Sebastopol: O'Reilly, c2008, xvii, 555 s. ISBN 978-0-596-51613-0.
- [2] KALOVÁ, Ilona a Karel HORÁK. ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY, Fakulta elektrotechniky a komunikačních technologií, Vysoké učení technické v Brně. Optické metody měření 3D objektů. *Elektrorevue* [online]. 2005, č. 23, 12.4.2005 [cit. 2013-05-30]. ISSN 1213-1539. Dostupné z: <http://www.elektrorevue.cz/clanky/05023/index.html>
- [3] 3D scanner. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-05-30]. Dostupné z: [http://en.wikipedia.org/wiki/3D\\_scanner](http://en.wikipedia.org/wiki/3D_scanner)
- [4] NIKOROVIC, Matej. *Získavanie priestorových obrazov*. 2012, 45 s. Dostupné z: <http://s.ics.upjs.sk/~mnikorovic/files/bac-slides.pdf>
- [5] GERIG, Guido. *Structured Lighting: CS 6320, 3D Computer Vision*. Spring 2012, 51 s. Dostupné z: <http://www.sci.utah.edu/~gerig/CS6320-2012/Materials/CS6320-CV-S2012-StructuredLight.pdf>
- [6] MACCORMICK, John. *How does the Kinect work?*. 2011, 52 s. Dostupné z: <http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf>
- [7] NAYEGANDHI, Amar. ETI –US GEOLOGICAL SURVEY, Florida Integrated Science Center, St. Petersburg, FL 33701. *LidarTechnology Overview*. Baton Rouge, LA.: Airborne LidarTechnology and Applications, June 20-21, 2007, 66 s. Dostupné z: [http://lidar.cr.usgs.gov/downloadfile2.php?file=Nayegandhi\\_Lidar\\_Technology\\_Overview.pdf](http://lidar.cr.usgs.gov/downloadfile2.php?file=Nayegandhi_Lidar_Technology_Overview.pdf)
- [8] Marine Pollution: Supplement 2.19: The Laser Fluorosensor (1/5). SEOS PROJECT. *Science Education through Earth Observation for High Schools* [online]. [2013] [cit. 2013-05-30]. Dostupné z: [http://lms.seos-project.eu/learning\\_modules/marinepollution/marinepollution-c02-s19-p01.html](http://lms.seos-project.eu/learning_modules/marinepollution/marinepollution-c02-s19-p01.html)
- [9] NAYEGANDHI, Amar. JACOBS TECHNOLOGY CONTRACTED TO US GEOLOGICAL SURVEY, St. Petersburg Coastal and Marine Science Center, St. Petersburg, FL 33701. *Green, waveform lidar in topo-bathy mapping – Principles*

- and Applications*. [2007], 26 s. Dostupné z:  
[http://www.ngs.noaa.gov/corbin/class\\_description/Nayegandhi\\_green\\_lidar.pdf](http://www.ngs.noaa.gov/corbin/class_description/Nayegandhi_green_lidar.pdf)
- [10] KALOVÁ, Ilona. *Segmentace a detekce geometrických primitiv: Počítačové vidění*. [2005]. Dostupné z:  
<http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/05%20-%20Segmentace%20a%20detekce%20geometrickych%20primitiv.pdf>
- [11] Deriche edge detector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-05-30]. Dostupné z:  
[http://en.wikipedia.org/wiki/Deriche\\_edge\\_detector](http://en.wikipedia.org/wiki/Deriche_edge_detector)
- [12] STRAKA, Stanislav. *Segmentace obrazu* [online]. Brno, 2009 [cit. 2013-05-30]. Dostupné z: [http://is.muni.cz/th/72784/fi\\_m/dp.pdf](http://is.muni.cz/th/72784/fi_m/dp.pdf). Diplomová práce. Masarykova Univerzita, Fakulta Informatiky. Vedoucí práce Mgr. Radka Pospíšilová.
- [13] 3D "scannery" pro každého!. *PIXEL* [online]. 2011, č. 178, s. 2 [cit. 2013-05-30]. Dostupné z: <http://www.pixel.cz/pdf/178-20-3d-scannery-pro-kazdeho.pdf>
- [14] DOLANSKÝ, Tomáš. *Lidary a letecké laserové skenování*. Vyd. 1. Ústí nad Labem: Univerzita J.E. Purkyně, 2004. 100 s. Acta Universitatis Purkynianae. ISBN 80-704-4575-0. Dostupné z:  
<http://wvc.pf.jcu.cz/ki/data/files/160lidaryweb.pdf>. Výzkum. Univerzita J.E.Purkyně v Ústí nad Labem.
- [15] ŠEBELA, Miroslav. *Detekce objektu ve videosekvencích*. Brno, 2010. Dostupné z:  
[https://www.vutbr.cz/www\\_base/zav\\_prace\\_soubor\\_verejne.php?file\\_id=27336](https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=27336). Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Petr Číka, Ph.D.
- [16] Segmentace obrazu. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 30.3.2013 v 16:49 [cit. 2013-05-30]. Dostupné z: [http://cs.wikipedia.org/wiki/Segmentace\\_obrazu](http://cs.wikipedia.org/wiki/Segmentace_obrazu)
- [17] ČÍRTEK, Jiří. *Sledování malých změn objektů*. Brno, 2008. Dostupné z:  
[https://dspace.vutbr.cz/bitstream/handle/11012/1964/Diplomova\\_prace\\_Jiri\\_Cirtek.pdf?sequence=1](https://dspace.vutbr.cz/bitstream/handle/11012/1964/Diplomova_prace_Jiri_Cirtek.pdf?sequence=1). Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Ing. Miroslav Richter, Ph.D.

- [18] Detekce hran. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 8.3.2013 v 17:58 [cit. 2013-05-30]. Dostupné z: [http://cs.wikipedia.org/wiki/Detekce\\_hran](http://cs.wikipedia.org/wiki/Detekce_hran)
- [19] Technical Infrastructure. *ENTERFACE 2012: The 8th International Summer Workshop on Multimodal Interfaces* [online]. 2012 [cit. 2013-06-05]. Dostupné z: <http://enterface12.metz.supelec.fr/technical.php>
- [20] *OpenCV* [online]. 2013 [cit. 2013-06-05]. Dostupné z: <http://opencv.org/>
- [21] 3D Vision with Stereo Disparity. LANKTON, Shawn. *Shawn Lankton Online: life, business, consulting, and computer vision* [online]. © 2002-2013, 19.12.2007 [cit. 2013-06-05]. Dostupné z: <http://www.shawnlankton.com/2007/12/3d-vision-with-stereo-disparity/>
- [22] SHPUNT, Alexander a Zeev ZALEVSKY. *Three-dimensional sensing using speckle patterns* [patent]. Přihláška, US20090096783 A1. Uděleno 16.4.2009. Dostupné z: <http://www.google.com/patents/US20090096783>
- [23] Image Processing (ICIP), 2012 19th IEEE International Conference on. In: ZHANG, Yueyi, Zhiwei XIONG a Feng WU. *Hybrid structured light for scalable depth sensing*. Orlando, FL: Conference Management Services, Inc., 2012, s. 17-20. ISBN 978-1-4673-2534-9 ISSN 1522-4880. DOI: 10.1109/ICIP.2012.6466784. Dostupné z: <http://research.microsoft.com/en-us/people/fengwu/depth-icip-12.pdf>
- [24] How Kinect works with PrimeSense. *Optoelectronic notes: Note some news about optics, technology*. [online]. 2010, 2.12.2010 [cit. 2013-06-10]. Dostupné z: <http://ntuzhchen.blogspot.cz/2010/12/how-kinect-works-prime-sense.html>
- [25] Google discloses costs of its driverless car tests. *DRIVE ON: blog* [online]. 2012, 14.6.2012 [cit. 2013-06-10]. Dostupné z: [http://content.usatoday.com/communities/driveon/post/2012/06/google-discloses-costs-of-its-driverless-car-tests/1#.UbXPvflM\\_l8](http://content.usatoday.com/communities/driveon/post/2012/06/google-discloses-costs-of-its-driverless-car-tests/1#.UbXPvflM_l8)
- [26] 'Eyes' of Google's Self-Driving Car May Bust Crooks. ZIFF DAVIS, Inc. *PC Magazine* [online]. 2012, 3.4.2012 [cit. 2013-06-10]. Dostupné z: <http://www.pcmag.com/article2/0,2817,2402516,00.asp>
- [27] Google Cars Drive Themselves, in Traffic. *The New York Times* [online]. 2010, 9.10.2010 [cit. 2013-06-10]. Dostupné z: <http://www.nytimes.com/2010/10/10/science/10google.html?pagewanted=all>



- [28] Stanley (vehicle). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 28.2.2013 [cit. 2013-06-12]. Dostupné z: [http://en.wikipedia.org/wiki/Stanley\\_\(vehicle\)](http://en.wikipedia.org/wiki/Stanley_(vehicle))
- [29] *Stanford Racing* [online]. 2006 [cit. 2013-06-12]. Dostupné z: <http://cs.stanford.edu/group/roadrunner//old/index.html>

## SEZNAM OBRÁZKŮ

Obrázek 1: Autonomní robotický automobil Volkswagen Touareg, Stanley [29] .....	10
Obrázek 2: Takto může vypadat stereo kamera. [19] .....	13
Obrázek 3: Geometrické zobrazení ideální stereo výbavy. [1] .....	15
Obrázek 4: Nepřímá úměra mezi hloubkou a disparitou. [1] .....	16
Obrázek 5: Vlevo nastavení kamer pro bližší objekty, vpravo nastavení kamer pro vzdálenější objekty. ....	17
Obrázek 6: Nejčastěji používaný souřadnicový systém: počátek souřadnicového systému je vlevo nahoře, plochy obrazů jsou řádkově zarovnány. [1] .....	18
Obrázek 7: Matematicky potřebujeme zarovnat obrazové plochy do jedné roviny tak, aby řádky pixelů obou kamer si odpovídali. Téměř nikdy se nám nepodaří získat ideální obrazy z kamer fyzicky. [1].....	19
Obrázek 8: Epipolární rovina je definována pozorovaným bodem $P$ a dvěma středy kamery $O_l$ , $O_r$ (bázemi). Epipóly jsou v bodě průniku přímky, spojující středy kamer, s rovinou projekce. [1].....	20
Obrázek 9: Ideální geometrie stereo zobrazování je zachycena esenciální maticí $E$ , který obsahuje veškeré informace o posunutí $T$ a rotaci $R$ . Popisuje vzájemné umístění druhé kamery k první v globálních souřadnicích. [1].....	21
Obrázek 10: Promítané strukturované světlo na plochu [4] .....	24
Obrázek 11: Zdroj světla spolu se snímačem a osvětleným bodem na zkoumaném objektu tvoří triangulační trojúhelník, pomocí něhož můžeme vypočítat hloubku [2] .....	25
Obrázek 12: Příklad strukturovaného světla [3].....	25
Obrázek 13: Příklad filtrace obrázku tak, abychom viděli pouze strukturované světlo [4] .....	26
Obrázek 14: Geometrie 3D triangulace strukturovaného světla [5] .....	27
Obrázek 15: Geometricky kalibrovaný 3D skener. Na obrazovce monitoru sestavy vidíme vyfiltrovaný obraz strukturovaného světla. [5] .....	28
Obrázek 16: Výpočet triangulace. [5] .....	28
Obrázek 17: Vzor strukturovaného světla, vysílaný Kinectem. [6] .....	29
Obrázek 18: LIDAR pro měření vzdálenosti [7].....	30
Obrázek 19: LIDAR pro Google Car. [26].....	31
Obrázek 20: Umístění LIDARu na střeše auta Google Car. [27].....	32

Obrázek 21: Ukázka, jak LIDAR na Google Car mapuje své okolí. [26] .....	32
Obrázek 22: Příklad využití propustnosti světla u lidarů, každá vrstva odrazí impuls zpět. [9] .....	33
Obrázek 23: Příklad prahování. [10] .....	35
Obrázek 24: Detekce hran. Vlevo původní obrázek, vpravo obrázek po aplikování detekce hran. [11] .....	36
Obrázek 25: Quart tree pro split and marge. [12] .....	37
Obrázek 26: Využití segmentace s quart tree. [11] .....	37
Obrázek 27: Označené body kolečkem jsou dobré body pro sledování, zatímco body ve čtvercích jsou špatnou volbou, dokonce i když jsou vybrány hranice objektu, je to špatně. [1] .....	38
Obrázek 28: Pohyb optiky a sledování. [1] .....	40
Obrázek 29: Vzhled okna v příkladovém demu na výpočet disparity. ....	42
Obrázek 30: Základní program [20] .....	43
Obrázek 31: Funkce stereo() - výpočet mapy disparity .....	44
Obrázek 32: Funkce slide_images() – 1.část .....	44
Obrázek 33: Funkce slide_images() – 2.část .....	45
Obrázek 34: Funkce winner_take_all .....	46
Obrázek 35: Funkce modefilt2(), .....	47
Obrázek 36: Okno, které zobrazuje obrázek .....	49
Obrázek 37: Okno s trackbarem .....	49
Obrázek 38: VanharaOpticalFlow.cpp – 1.část .....	50
Obrázek 39: VanharaOpticalFlow.cpp – 2.část .....	50
Obrázek 40: VanharaOpticalFlow.cpp – 3.část .....	51

## SEZNAM PŘÍLOH

Příloha 1: CD