Ústav přístrojové a řídící techniky

Senzorické systémy

Semestrální práce
Určení vzdálenosti pomocí kamer

Jan Rychtera

1. Úvod

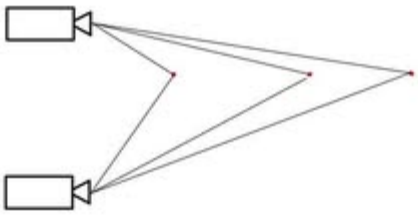
Člověk se v reálném životě orientuje pomocí syntézy snímačů různých vjemů. Od teploty získávané skrz povrch těla, detekci potenciálně nebezpečných látek čichem, hrozby pomocí sluchu, stabilitu pomocí mozečku, a pro orientaci v prostoru využívají lidé primárně zrak.

Zdraví lidé získávají zrakové informace pomocí uskupení dvou očí, nepřekvapivě umístěné na hlavě, z nichž se pomocí získaných informací v mozku skládá prostorové vnímání okolí. Samozřejmě, lidé dokáží získat tyto informace i za použití pouze jednoho zdroje obrazu, nicméně to je primárně na základě zkušeností.

Tato pozorování jsou důležitá, neboť se jimi dá inspirovat i do technického prostředí. Inspirována metodou dvou posunutých snímačů obrazů je metoda, jež se anglicky dá nazvat *Stereo vision depth analysis.*

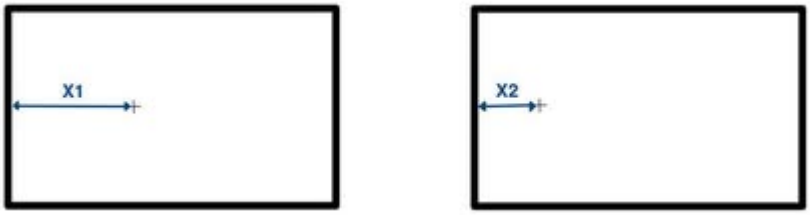
2. Princip fungování

Metoda se snaží analogicky imitovat systém dvou očí pomocí dvou posunutých kamer vedle sebe. Každá kamera poté fotí stejné prostředí, a stejně jako u lidských očí, obraz každé z kamer je mírně jiný, konkrétně posunutý viz obr.1.



Obr. 1

Tento posun se v angličtině nazývá *disparity* a v této práci bude nazývaná disparita. Tuto hodnotu lze určit z *obr.2*



Obr. 2

A vyhodnotit ji jako [1]:

$$D = x_1 - x_2$$

Získáme-li velikost tohoto posunu, je možné vyhodnotit informace o vzdálenosti, a to konkrétně podle rovnice [1]:

$$z = \frac{f}{d} \cdot \frac{T}{D}$$

Kde:

D…disparita (posun objektu mezi kamerami)
f …ohniskové vzdálenosti kamery
d…fyzická velikost pixelu na snímači kamery
T…základní vzdálenost mezi středy kamer
z…vzdálenost mezi objektem a kamerou

Pro určení disparity je potřeba vyhledat pixel z jednoho obrazu (označme jako základní obraz levý obraz, z konvence čtení zleva), v druhém obraze, pravém. K vyhledání se dá použít princip *stereo-matching* [1] [2]. Pro téměř každý pixel (výjimkou mohou být krajní prvky obrazu) v základním obrazu se porovnává oblast *nxn* matice kolem porovnávaného pixelu s oblastmi v obrazu pravém (je vhodné volit *n* jako liché číslo, neboť oblast má pak střed). Toto porovnání může být provedeno například jako:

$$k = \sum_{i,j} \left(L_{i,j} - R_{i,j}\right)^2$$

Kde:

k…kvalita shody
$L_{i,j}$…matice oblasti z levého obrazu
$R_{i,j}$…matice oblasti z pravého obrazu

Disparita se poté vyhodnotí z bodů s největší kvalitou shody (mezi nejvíce shodnými oblastmi). Pro snazší

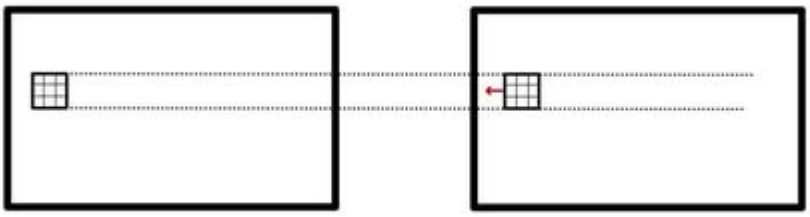
3. Zlepšení realizace

Realizace hledání disparity je výpočetně časově náročná záležitost (je-li k ní přistoupeno přímo výše zmíněnými principy). Pro zefektivnění této činnosti a snazší realizaci řešení je možno udělat několik kroků.

- Snímky pořídit typově stejnými kamerami se stejným rozlišením a stejným přiblížením. Dodržením tohoto principu je možné předejít nutné předúpravě snímků.
- Snímky pořídit ve stejné rovině (kamery co nejvíce rovnoběžné ve všech směrech), aby kamery byly pouze posunuté a obraz nebyl více deformovaný.
- Snímky pořídit ze stejné výšky.

Dodržením těchto kroků je možné vyhledávat mezi obrázky pouze ve stejné výšce (*y* souřadnici).

Předpokladem levé a pravé kamery lze nadále vyhledávací oblast omezit, vyhledáváme-li oblast zleva v pravém obrázku, a to na oblast pouze vlevo od pixelu odpovídající středu vyhledávané oblasti viz obr 3., a tuto oblast lze ještě omezit na vzdálenost omezenou odhadovanou maximální disparitou. Zde nutno podotknout, že maximální disparita se bude měnit s rozlišením obrazu.



Obr. 3

Dále pro zefektivnění výpočtu je vhodné obrázky převést do tzv. *„greyscale“*. Každý pixel obrázku nese normálně tři hodnoty barev RGB. Převedením do odstínů šedi se tyto tři hodnoty zredukují do pouze jedné hodnoty.

Pro dosažení těchto podmínek byly použité obrázky z datasetů Middlebury College ve Vermontu dostupné na internetu.

Nadále se výkon dá zlepšit například paralelním programováním problému viz [3], nicméně tomu se v této práci nebude věnováno.

4. Realizace kódu

Ručně psaný kód i s komentáři v pythonu je v příslušném GIT repositáři. V tomto dokumentu budou stručně popsány použité classy, metody a funkce, pro přesnou implementaci se obraťte přímo na kód.

Kód používá knihovnu *numpy* pro maticové výpočtu, knihovnu *cv2* pro práci s obrázky a porovnání s veřejně dostupným nástrojem, a knihovnu *matplotlib.pyplot* pro vizualizaci výsledků. Zbylé importy jsou buď pro syntax nebo základní importy pythonu.

class StereoDepth():

Classa pro použití ručně psaného kódu.

def __init__(*self*, *img_left*, *img_right*, *search_block_size*:*int*):

Kreator uvnitř StereoDepth classy, přijímá argumenty pro levý, pravý obrázek a velikost vyhledávací matice (*n*), pro lichá n.

def SSD_counter(*self*, *left_matrix*, *right_matrix*):

Hodnotící metoda, implementována:

```
match = np.sum((left_matrix - right_matrix)**2)
```

Jako suma druhých mocnin rozdílů prvků.

def solve(*self*):

Metoda která pomocí dvou for-smyček prochází levý obrázek zleva doprava a shora dolu a pro každý bod volá metodu *self.right_runner(*).* Z této metody dostane zpět x souřadnici s nejlepší shodou oblastí z pravého obrázku, vypočte disparitu a tu uloží do atributu:

```
        self.disparity_map = np.zeros_like(img_left, dtype=np.float32)
```

def right_image_runner(*self*, *x_left*, *y_left*, *left_matrix*):

Metoda procházející v pravém obrázku oblasti vlevo od aktuálně zkoumaných souřadnic v levém obrázku. Přijímá aktuální souřadnice z levého obrázku x_left, y_left, a pro porovnání přijímá i aktuálně zkoumanou oblast levého obrázku, neboť volá *self.SSD_counter(*).* Metoda navrací x souřadnici s nejlepší shodou oblastí.

def heatmap_show(*self*):

Balastová metoda pro vizualizaci výsledků pomocí matplotlib.

def plot_on_ax(*self*, *ax*, *title*="Stereo Matching Result"):

Balastová metoda pro vizualizaci výsledků pomocí matplotlib, sloužící k porovnání vlivu nastavení velikosti porovnávací oblasti v závislosti na velikosti atributu *n*.

def disparity_to_distance(*self*, *disparity_map*, *focal_length_px*, *baseline_m*):

Balastová metoda sloužící pro vizualizaci výsledků převedeny na vzdálenost. Převedení je implementací vzorce v kapitole 2.

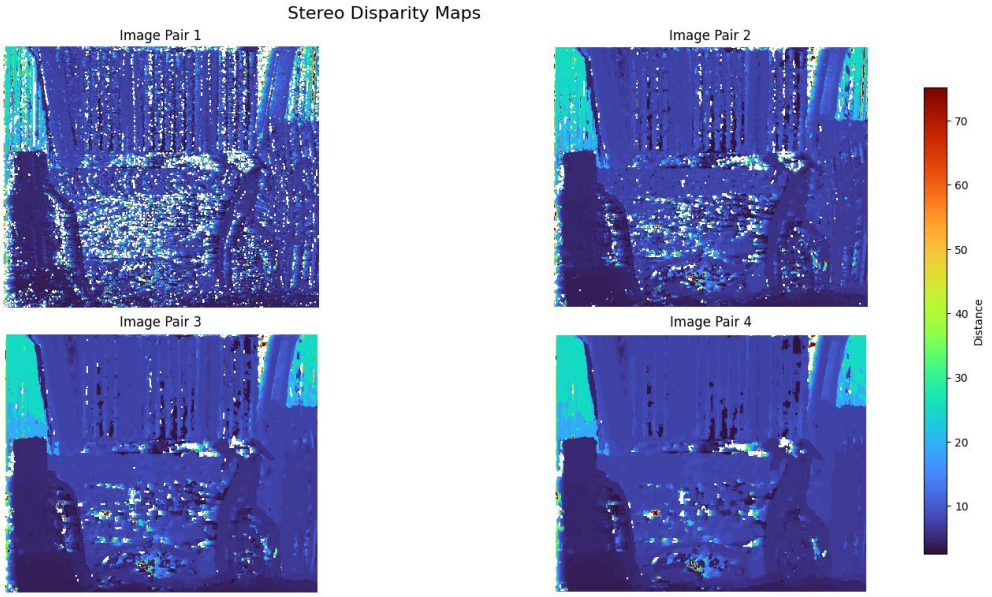
```
        disparity_copy = disparity_map.copy()
        mask = disparity_copy <= 0
        disparity_copy[mask] = 0.0001  # or mask later
        depth_map = (focal_length_px * baseline_m) / disparity_copy
        return ma.masked_array(depth_map, mask)
```

5. Výsledky

Pro originální sadu obrázků kde levý je pro referenci:

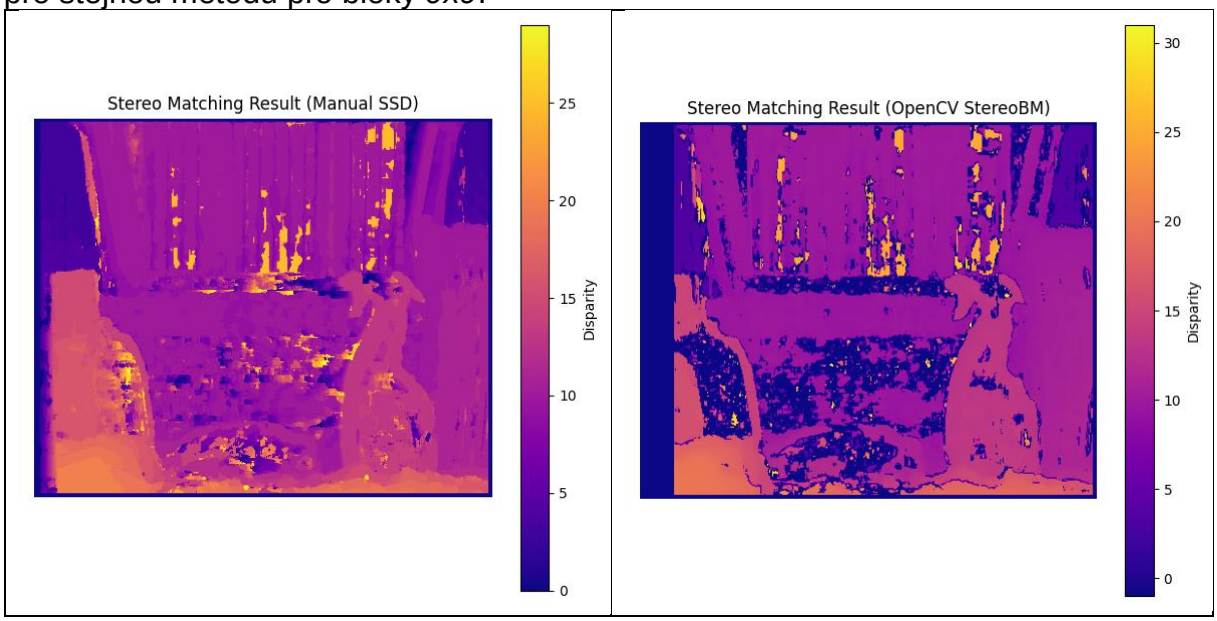
Byly nalezeny výsledky kde:

- Image Pair 1 – pro matici 3x3
- Image Pair 2 – pro matici 7x7
- Image Pair 3 – pro matici 5x5
- Image Pair 4 – pro matici 9x9



Je vidět že velikost vyhledávací oblasti ovlivňuje kvalitu výsledku, kde obstojné jsou pravděpodobně velikosti *7x7* a *9x9.* Tyto velikosti však nemusí být ideální pro jiné případy a jsou silně vázány na použité vstupy.

U map disparit je pak porovnána vlastní implementace s implementací knihovny cv2 pro stejnou metodu pro bloky *9x9*.



Je vidět že výsledkově je vlastní implementace přijatelná.

Největší rozdíl je však v rychlosti kalkulací. Z výňatku z výpisu v konzoli je vidět, že knihovna cv2 je zhruba 3000 krát rychlejší než vlastní implementace.

```
(388, 470)
Running...
Image Pair 1 plotted in 44.861 seconds
Image Pair 2 plotted in 44.484 seconds
Image Pair 3 plotted in 44.600 seconds
Image Pair 4 plotted in 45.204 seconds
Running OpenCV StereoBM...
OpenCV StereoBM computed in 0.013 seconds
```

6. Závěr

Byl vytvořen skript, který vyhledává disparitu a přepočítává ji na vzdálenost. Tento skript by se dal považovat za minimálně částečný úspěch, nicméně výrazně zaostává za dostupnými knihovnami jako např. srovnává cv2. Tato knihovna je napsána efektivněji v C nebo C++ a dosahuje větších výpočetních rychlostí než autorský pythonovský skript i pravděpodobně lepších výsledků.

Za zmínku nadále stojí i použití neuronových sítí pro odhad vzdálenosti ze stereo kamer, konkrétně knihovna RAFT-Stereo, který výsledky odhaduje z naučených dat.