



ZÁPADOČESKÁ  
UNIVERZITA  
V PLZNI

Fakulta elektrotechnická

Katedra aplikované elektroniky a telekomunikací

# BAKALÁŘSKÁ PRÁCE

Lokalizace malých těles pomocí optických metod

Autor práce: Jessica Fenclová

Vedoucí práce: Ing. Martin Juřík

Plzeň 2019

ZÁPADOČESKÁ UNIVERZITA V PLZNI  
Fakulta elektrotechnická  
Akademický rok: 2018/2019

**ZADÁNÍ BAKALÁŘSKÉ PRÁCE**  
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jessica FENCLOVÁ**  
Osobní číslo: **E16B0073P**  
Studijní program: **B2612 Elektrotechnika a informatika**  
Studijní obor: **Elektronika a telekomunikace**  
Název tématu: **Lokalizace malých těles pomocí optických metod**  
Zadávací katedra: **Katedra aplikované elektroniky a telekomunikací**

Z á s a d y p r o v y p r a c o v á n í :

1. Zpracujte rešerši zabývající se principy určování polohy objektů pomocí optických metod.
2. Navrhněte alespoň jeden ze způsobů, pomocí kterého bude možné určit polohu předem definovaného objektu v ploše.
3. Zrealizujte měření s ohledem na klíčové parametry - předpokládaná přesnost a výpočetní náročnost.
4. Zhodnoťte dosažené výsledky použitého řešení.

Rozsah grafických prací: podle doporučení vedoucího

Rozsah kvalifikační práce: 30 - 40 stran

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

1. Singh U., Singh S., Srivastava M.: "Object Detection and Localization Using SURF Supported By K-NN", International Journal of Computer Science Trends and Technology (IJCSST), Volume 3, Issue 2, Mar-Apr 2015.
2. Oztarak H., Akkaya K., Yazici, A.: "Lightweight Object Localization with a Single Camera in Wireless Multimedia Sensor Networks", GLOBECOM 2009 - 2009, IEEE Global Telecommunications Conference, 2009, kapitola IV.
3. Image Processing Toolbox - <https://www.mathworks.com/products/image.html>

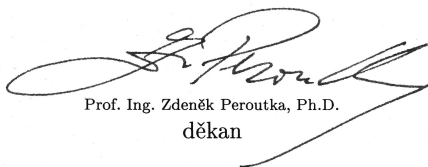
Vedoucí bakalářské práce:

**Ing. Martin Juřík**


Katedra teoretické elektrotechniky

Datum zadání bakalářské práce: 5. října 2018

Termín odevzdání bakalářské práce: 13. června 2019

  
Prof. Ing. Zdeněk Peroutka, Ph.D.  
děkan



  
Doc. Dr. Ing. Vjačeslav Georgiev  
vedoucí katedry

V Plzni dne 5. října 2018

# Abstrakt

V této bakalářské práci jsou nejprve charakterizovány metody lokalizace pomocí kamery. Je uveden rozbor použitých algoritmů, jako jsou Block matching, integrální obraz a SURF. Dále jsou popsány optoelektronické senzory. V praktické části byla využita kamera pro lokalizaci malého tělesa a v programu byly implementovány algoritmy pro zpracování snímků. Součástí práce je ukázkové zařízení navržené pro lokalizaci těles senzorem typu jednocestné závory. Toto snímací zařízení je řízeno přes mikroprocesor s jádrem HCS08. V závěru práce jsou porovnány rychlosti obou metod pro optickou lokalizaci.

## Klíčová slova

Block matching, SURF, integrální obraz, lokalizace, senzor, LED, fototranzistor

# Abstract

Fenclová, Jessica. *Localization of small objects using optical methods* [*Lokalizace malých těles pomocí optických metod*]. Pilsen, 2019. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Martin Juřík

---

In this bachelor thesis the methods for localization using a camera are described first. An analysis is done of algorithms used such as the Block matching algorithm, integral image and SURF. Furthermore optoelectronic sensors are also described. In the practical section a camera was used to detect a small object and algorithms were implemented for image processing. Part of this thesis is a prototype device designed as a through-beam mode sensor for the localization of objects. This device is controlled by a microprocessor with a HCS08 core. Finally, the speeds of the methods for optical localization are compared.

## Keywords

Block matching, SURF, integral image, localization, sensor, LED, phototransistor

## Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 13. června 2019

Jessica Fenclová

.....

Podpis

# Obsah

Seznam obrázků	vii
Seznam tabulek	viii
Seznam symbolů a zkratk	ix
<b>1 Úvod</b>	<b>1</b>
<b>2 Teoretická část</b>	<b>2</b>
2.1 Lokalizace pomocí kamery . . . . .	2
2.1.1 Algoritmus Block matching . . . . .	3
2.1.2 Integrální obraz . . . . .	5
2.1.3 Další algoritmy používané v odvětví počítačového vidění . . . . .	6
2.2 Lokalizace pomocí fotocitlivých součástek . . . . .	9
2.2.1 Využití reflexních senzorů . . . . .	11
2.2.2 Využití jednocestné závory . . . . .	13
<b>3 Praktická část</b>	<b>14</b>
3.1 Aplikace kamery a algoritmů . . . . .	15
3.1.1 Základní aplikace algoritmu Block matching . . . . .	16
3.1.2 Hledání kolem maxima rozdílu s referenčním snímkem . . . . .	16
3.1.3 Hledání v okolí zajímavých bodů . . . . .	18
3.1.4 Aplikace integrálního obrazu a detektoru rohů . . . . .	19
3.2 Využití navrženého zařízení pro lokalizaci . . . . .	20
3.2.1 Návrh a výroba DPS . . . . .	20
3.2.2 Software mikroprocesoru . . . . .	22
3.2.3 Simulace a zpracování dat . . . . .	24
3.3 Porovnání naměřených výsledků dvojice aplikovaných metod . . . . .	26
<b>4 Závěr</b>	<b>32</b>
Reference, použitá literatura	34
Přílohy	36

<b>A Schémata zapojení</b>	<b>36</b>
A.1 Deska MCU . . . . .	36
<b>B Desky plošných spojů, výkresy</b>	<b>41</b>
<b>C Použité skripty, zdrojové kódy</b>	<b>46</b>
C.1 Skript Lokalizace.m . . . . .	46



# Seznam obrázků

2.1	Block matching algoritmus s parametrem <i>SAD</i>  Převzato z [1]  . . . . .	4
2.2	Výpočet integrálního obrazu z pixelů . . . . .	6
2.3	Detektor rohu . . . . .	7
2.4	Scale-space  Převzato z [5]  . . . . .	9
2.5	Deskriptor  Převzato z [5]  . . . . .	9
2.6	Reflexní model . . . . .	11
2.7	Retroreflektivní model . . . . .	11
2.8	Model skeneru na automobilu  Převzato z [11]  . . . . .	12
2.9	Mračno bodů  Převzato z [3]  . . . . .	13
2.10	Through-beam režim . . . . .	13
3.1	Model arény s robotem-Scarabeus  Převzato z [7]  . . . . .	14
3.2	Grafický zobrazení principu BMA s hledání kolem maxima rozdílu . . . . .	17
3.3	Snímek prázdné arény a šablona robota . . . . .	18
3.4	Grafický znázornění principu algoritmu BM urychleného s užitím SURF . . . . .	19
3.5	Grafický znázornění simulace zastínění fototranzistorů při některé z možných pozic robota . . . . .	21
3.6	Zobrazení průběhu na osciloskopu s paketem bajtů posílaných do počítače . . . . .	23
3.7	Vývojový diagram algoritmu pro redukci stejných matic . . . . .	25
3.8	Graf závislosti limitu na průměrné naměřené hodnotě . . . . .	26
3.9	Grafické znázornění lokalizování zakrytého sedmého senzoru . . . . .	30
3.10	Grafické znázornění lokalizování při zapojení senzorů ob jeden . . . . .	30
A.1	Propojení MCU s USB . . . . .	36
A.2	Schéma desky s MCU . . . . .	37
A.3	Schéma desky s USB . . . . .	38
A.4	DPS MCU a USB-top . . . . .	39
A.5	DPS MCU a USB-bot . . . . .	40
B.1	Osazení součástek na DPS s mikroprocesorem . . . . .	41
B.2	Osazení součástek na horní vrstvu DPS s LED . . . . .	42
B.3	Osazení součástek na spodní vrstvu DPS s LED . . . . .	42
B.4	Osazení součástek na horní vrstvu DPS s fototranzistory . . . . .	43

B.5 Osazení součástek na spodní vrstvu DPS s fototranzistory . . . . .	43
B.6 Rozlití vodivé vrstvy na horní straně DPS s mikroprocesorem . . . . .	44
B.7 DPS-Zleva LED a zprava fototranzistorů . . . . .	45

# Seznam tabulek

3.1	Jednotlivé bajty dat, posílané přes sériovou komunikace (kde val je zkráceně slovo value (česky hodnota)). . . . .	23
3.2	Průměrné naměřené časy trvání jednotlivých algoritmů v oříznutém snímku 412x438 . . . . .	27
3.3	Průměrné naměřené časy trvání jednotlivých algoritmů v oříznutém snímku 1000x980 . . . . .	27
3.4	Porovnání výsledných pozic při aplikaci různých algoritmů pro snímky o velikosti 412x438 . . . . .	27
3.5	Porovnání výsledných pozic při aplikaci různých algoritmů pro snímky o velikosti 1000x980 . . . . .	28
3.6	Porovnání výsledných pozic algoritmů integrální obraz a BMA urychlené se SURF při užitím různě velkých šablon. V rozlišení 412x438 . . . . .	28
3.7	Porovnání výsledných pozic algoritmů integrální obraz a BMA urychlené se SURF při užitím různě velkých šablon. V rozlišení 1000x980 . . . . .	29
3.8	Porovnání pozic vypočítaných algoritmy . . . . .	29
3.9	Porovnání rychlostí metody rozsáhlé (porovnání všech matic) se zkrácenou metodou (zredukovaný počet matic) . . . . .	29
3.10	Porovnání počtu neunikátních matic pro různé simulace rozmístění součástí	30

# Seznam symbolů a zkratek

ADC .....	Analog to Digital Converter. Analogově-digitální převodník.
ASCII .....	American Standard Code for Information Interchange. Americký standardní kód pro výměnu informací.
BMA .....	Block Matching Algorithm.
DPS .....	Deska plošných spojů.
IDE .....	Integrated Development Environment. Vývojové prostředí.
LED .....	Light-Emitting Diode. Elektroluminescenční dioda.
MPEG4 .....	Kód komprimace.
RAM .....	Random Access Memory. Paměť s náhodným přístupem.
RGB .....	Red, blue, green. Barevný model.
SMD .....	Surface Mount Device. Součástka určená pro povrchovou montáž.
SIFT .....	Scale-invariant feature transform.
SURF .....	Speeded-up robust features.
THT .....	Through Hole Technology. Technologie osazování plošných spojů součástkami s drátovými vývody.
UART .....	Universal Asynchronous Receiver-Transmitter.
USB .....	Universal Serial Bus. Univerzální sériová sběrnice.

# 1

## Úvod

Cílem práce bylo vyzkoušet různé optické metody lokalizace malých těles. Vyhledávání rychlých a optimálních metod pro lokalizaci těles je všeobecně zapotřebí. Důvodem pro vznik práce byla potřeba lokalizovat malé těleso, konkrétně malého robota na platformě zvané Scarabeus, viz obr.3.1, který vznikl na Fakultě elektrotechnické Západočeské univerzity v Plzni. V práci byla zvolena optická metoda, kterou bylo možno pojmout dvojitým způsobem. První způsob, zvolený pro optickou lokalizaci, byl s pomocí kamery a algoritmů. Kamerou je možné natáčet video pohybujícího se tělesa a následně bylo v každém snímku těleso vyhledáno prostřednictvím algoritmu. Druhý způsob byl proveden s postaveným zařízením, které bylo navrženo jako fotoelektrický senzor typu jednocestné závory.

V práci jsou v první kapitole nejdříve představeny metody optické lokalizace. Postupně jsou vysvětleny algoritmy používané pro lokalizaci počínaje Block matching algoritmem, integrálním obrazem a SURF algoritmem. Následující část kapitoly se zabývá lokalizací pomocí fotoelektrických senzorů. Kromě toho je uvedeno praktické využití jednotlivých typů fotoelektrických senzorů.

Druhá kapitola obsahuje popis naší strategie lokalizace dvěma metodami. Jen mezi algoritmy pro zpracování obrazů patří několik druhů a variací algoritmů, proto bylo nutné zúžit možnosti na konkrétní algoritmy. V tomto případě na integrální obraz, SURF algoritmus a Block matching algoritmus. Bylo prozkoumáno, jak tyto algoritmy optimalizovat. V další části kapitoly je popsána metoda lokalizace s použitím fotoelektrického senzoru. Vytvoření fotoelektrického senzoru zahrnovalo několik kroků. Nejprve bylo navrženo zařízení v příslušném programu. Dalším krokem byla výroba desky z návrhu a následovalo osazování součástek na desku a jejich pájení. Posledním krokem k zhotovením zařízení bylo naprogramování mikroprocesoru. V závěrečné části kapitoly byly porovnány naměřené výsledky dvou metod lokalizace těles. Cílem bylo přijít na to, která metoda je nejrychlejší a nejspolehlivější pro optickou lokalizaci malých těles.

# 2

## Teoretická část

Lokalizace a detekce předmětů je nedílnou součástí moderních zařízení a systémů v průmyslových odvětvích jako je robotika, logistika a doprava. V robotice je důležitá lokalizace předmětů, protože pro řízení robotů je potřeba mapovat okolí a také detekovat překážky. Totéž platí při návrhu autonomních vozidel. Pro představu metod optické lokalizace je probrána její teorie a praktická aplikace v běžném světě. Nejdříve je popsána metoda lokalizace kamerou a aplikace algoritmů pro zpracování snímků. Druhá popsaná metoda je lokalizace se senzory.

### 2.1 Lokalizace pomocí kamery

Jedna z metod lokalizace je s užitím kamery, ze které se snímky dále zpracovávají vybraným algoritmem. Odvětví, které se zabývá zpracováváním snímků a jejich výpočetní analýzou, se jmenuje *Image processing*. Systém většinou zpracovávaný snímek vnímá jako dvoudimenzionální signál, snímek tedy může být definován jako funkce  $F(x, y)$  a amplituda  $F$  s dvojicí souřadnic je označena jako intenzita snímku v tomto daném bodě nebo častěji za intenzitu pixelu. Postup při zpracování snímků se dělí do několika kroků. Ještě před zahájením procesu je snímek modifikován do správné podoby, se kterou lze dále dobře pracovat. Modifikací do správné podoby může být například odstranění šumu ze snímku, maximalizování zaostření nebo změna velikosti. Cílem může být odfiltrování šumu, který představuje v oddělené přední scéně (přední scéna = scéna popředí, zadní scéna = pozadí) snímku další nechtěná miniaturní tělesa. Je možno také použít morfologický filtr, který pomůže odfiltrovat artefakty vnesené do snímku šumem. Častým procesem je také *prahování* (orig. Thresholding). Porovnáním intenzity pixelu s konstantou je postupně rozdělen celý snímek na černé a bílé pixely. Konstantou bývá hodnota průměru intenzity originálního snímku. V prvním kroku zpracování snímku je snímek rozdělen na přední a zadní scénu, tento proces se nazývá *segmentace snímku*, kde přední scénou se obvykle rozumí těleso a zadní scénou pozadí. Posledním krokem je *zaznamenání snímku* (orig. Image registration), což je proces srovnání obrázků z různých datových sad pro vizuální nebo výpočetní analýzu a to buď na základě intenzit, nebo na základě vybraných rysů.

Takto zpracované snímky je možno využít v algoritmech pro detekci.

### 2.1.1 Algoritmus Block matching

Algoritmus zvaný anglicky *Block Matching Algorithm*, dále BMA, v překladu přiřazování bloků, je jeden z mnoha aplikovaných algoritmů pro zpracování snímků. BMA je široce využíván v analýze pohybů ve dvoudimenzionálních snímkových datech. Jeho další aplikace je v kompresi videa do formátu jako je například MPEG4. BMA má několik verzí od základního *úplného vyhledávání* (orig. Full search) až po současné rychlé adaptivní algoritmy, jako *Adaptive rood pattern search*. Všeobecně lze u BMA modifikovat algoritmus a tím i jeho výsledky buď podle typu zadané velikosti bloku (pevný nebo variabilní), nebo rozlišením snímku, tedy počtem pixelů. Snímek je rozdělen do bloků pixelů, například pro kompresi MPEG do bloků o velikosti 16x16 pixelů. Každý blok a jeho posunutí je predikováno z bloku stejné velikosti v referenčním snímku. Blok je jen posunut na predikovanou pozici. Toto posunutí je označeno *vektorem pohybu* (orig. Motion vector). Úkolem je získat nejvhodnější vektor pohybu použitím vyhledávací metody oblasti pixelů a vhodnou manipulací BMA parametrů. Hlavním parametrem je *kritérium shody* (orig. Matching criteria), které udává změřenou shodu mezi bloky. Jedním z nich je suma absolutního rozdílu, zkráceně SAD, ta se vypočítá dle rov. 2.1.

$$SAD(p, q) = \sum_{i=1}^n \sum_{j=1}^n (I_c(i, j) - I_r(i + p, j + q)) \quad (2.1)$$

Kde:

- $I_r(i+p, j+q)$  – cílový snímek
- $I_c(i, j)$  – referenční snímek
- $i, j$  – pozice bloku
- $n$  – velikost bloku
- $p, q$  – o kolik se posunul snímek

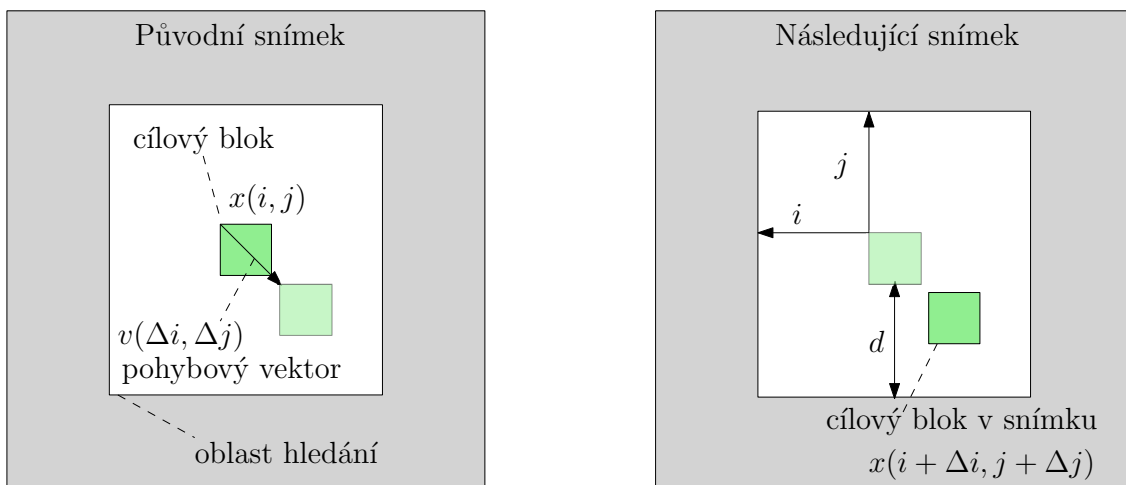
Jako nejspolehlivější kritérium shody je uvedeno v [2] MSD (orig. Mean Square Difference), které se vypočítá podle rov. 2.2. MSD generuje kvalitnější sekvence než ostatní, například MAD (orig. Mean of Absolute values of Differences).

$$MSD(p, q) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (I_c(i, j) - I_r(i + p, j + q))^2 \quad (2.2)$$

Kritéria, jako MSE (orig. Mean Squared Error) nebo MAE (orig. Mean Absolute Error), je možné používat ke kvantitativnímu prokázání porovnaných snímků. Kompenzace pohybu je proces, při kterém je z bloku v referenčním snímku predikováno posunutí

bloku, toto posunutí je odhadnutý vektor pohybu. Kompenzace pohybu je dosaženo pomocí interpolace sousedících bodů. Zlomkovou vzdáleností mezi pixely jsou interpolovány pixely do referenčního snímku. Předpověď kompenzovaného pohybu je ovlivněna velikostí bloku. U některých typů se bere v úvahu jen možnost snížení počtu prohledaných bodů. Pro každý bod se stále musí vypočítat kritérium shody pro blok o rozměru  $B \times B$ . Cílem je volit výpočetně nejméně náročný algoritmus. Pro vyhledávání nejlepší shody s daným blokem lze definovat dva typy strategie vyhledávání:

- hledání globálního minima v celém snímku,
- mít podle nastavených parametrů již odhad polohy globálního minima a vyhledávat blízko něho.



**Obr. 2.1:** Block matching algoritmus s parametrem  $SAD$  [Převzato z [1]]

U základního typu BMA úplného vyhledávání je v každém bloku snaha snížit hodnotu  $SAD$  a z něho získat ten požadovaný vektor pohybu, který se poté stane vlastností současného bloku. Jenže úplné vyhledávání je prostorově i časově výpočetně náročné.  $SAD$  evaluace je také časově náročná. Proto bylo v nedávné době navrženo několik rychlých BMA, aby byl snížen počet  $SAD$  operací. Rychlé BMA v podstatě používají jen podmnožinu oblasti pro snížení počtu hledání. Nejvíce používané varianty jsou následující:

- *3-step-search* - česky vyhledávání po třech krocích. Začíná uprostřed s velikostí kroku rovné čtyřem. V první fázi je osm sousedících kandidátů pro vyhledávání vzdálených od středu o velikost kroku  $s$ . Z pozic vyhledaných je zvolena ta s nejmenší hodnotou vypočítaného kritéria. Při každé další fázi se krok zmenšuje o polovinu současné velikosti opakovaně až do velikosti rovné jedné. Takže se algoritmus zastaví po třech krocích. Později byl navržen rychlejší nový 3-step-search a později 4-step-search.
- *New 3-step-search* - nová verze vyhledávání po třech krocích je rychlejší díky tomu, že přehlídí nepatrné pohyby a je schopno se zastavit v polovině vyhledávání, aby



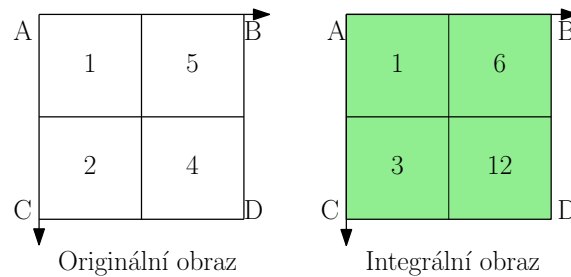
tak snížilo výpočetní náklady. V první fázi se podobá svému předchůdci, ale navíc vyhledává osm sousedních pozic i s krok o velikosti jedné. Z šestnácti pozic vybírá pozici s nejmenší hodnotou kritéria. Když je pozice s nejmenším vypočítaným kritériem v počátku, zastaví se vyhledávání a nastaví se vektor pohybu na souřadnice (0,0), jinak se nastaví pozice jako výchozí pro další vyhledávání.

- *Logarithmic search* - prostřední blok prohledávané oblasti a čtyři další od něho vzdálené o krok  $s$  jsou vybrány pro porovnání s cílovým blokem. Blok, který byl shodný, se nově určí jako střední blok a opakuje se vyhledávání čtyř bloků kolem. Velikost bloku oblasti prohledávání se zmenšuje o polovinu jen při splnění kritéria, že blok shody je na okraji okna, jinak se velikost kroku nemění.
- *Hierarchical search* - je rychlý a účinný přístup. Při vyhledávání je vytvořena pyramida, kde první úroveň obsahuje snímek o původní velikosti a každá další úroveň o čtvrtinu menší. Algoritmus se snaží zkombinovat výhody velkých bloků s výhodami malých. Velké bloky spíše dokáží sledovat opravdový pohyb oproti blokům malým. Jejich kvalita shody je ale menší než u malých bloků. Takže se nejdříve využívá schopnost sledování pohybů velkých bloků (přiřadí se velké bloky k sobě) a pak se využije jejich vektor pohybu jako počáteční bod pro vyhledávání pomocí malých bloků.

### 2.1.2 Integrální obraz

Integrální obraz je používán jako rychlý a efektivní způsob výpočtu hodnot pixelů v daném snímku. Zpracování snímku, ve snímku s vysokým rozlišením, vyžadovalo zrychlení výpočtu, a tak byl v roce 1984 vytvořen algoritmus *integrální obraz* (orig. Integral image). Problematická byla nutnost vypočítat průměr dané oblasti několikrát za sebou. Tato časově náročná fáze byla urychlena s integrálním obrazem. Využívá se především k výpočtu průměrné intenzity v určitém snímku. Díky rychlosti je součástí několika algoritmů počítačového vidění. Viola-Jones algoritmus pro rozpoznání obličejů jej znovu zpopularizoval v roce 2001, a v roce 2006 byl použit i v SURF algoritmu. Integrální obraz pozice  $x,y$  v matici pixelů je roven součtu hodnot všech pixelů nad ní a vlevo včetně té pozice  $i(x,y)$ , jak je vidět v rovnici 2.3. To je provedeno pro každý pixel, až do dolního pravého pixelu, který tím pádem je součtem všech pixelů ve snímku. Pokud máme vygenerovaný integrální obraz, pak výpočet sumy pixelů obdélníku A,B,C,D s libovolnými rozměry zdrojového snímku, lze provést pouhým sečtením pixelů A,B,C,D. Výhodou je, že výpočetní čas je nezávislý na velikosti obdélníku.

$$II(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (2.3)$$



**Obř. 2.2:** Výpočet integrálního obrazu z pixelů

### 2.1.3 Další algoritmy používané v odvětví počítačového vidění

Pro rozpoznání předmětů a popisu obsahu obrázku se používají *SIFT* (orig. Scale-Invariant Feature Transform) a jeho zrychlená verze *SURF* (orig. Speeded-up Robust Features). Tyto algoritmy jsou nejpraktičtěšími metodami pro detekci a popis klíčových bodů v snímku a to proto, že jsou invariantní vůči měřítku, rotaci, jasů a rozmazání snímku. V článku [6] se autoři zabývali porovnáním těchto metod a můžeme se v něm dočíst, že je *SURF* více invariantní vůči rotaci než *SIFT*. V případě nezávislosti na měřítku získáme lepší výsledky použitím *SIFT*. Výhodou *SURF* je třikrát větší rychlost, než u *SIFT* a to díky tomu, že využívá integrální obraz a čtvercové filtry. Proto se v průmyslu standardně používá *SURF*. *SIFT* a *SURF* mají 3 základní fáze:

1. Dohledání klíčových bodů, které se pravděpodobně dají dohledat v různých snímcích totožného objektu znovu. Tyto klíčové body by měly být invariantní. Rohy, skvrny a podobně jsou dobré body a většinou je lze dohledat v různých měřítkách.
2. Přiřazení orientace klíčovým bodům. Najít pravou "orientaci" toho bodu, tím že jsou nasbírány směrové gradienty kolem bodů. Z histogramu gradientů zjistíme, která orientace je nejvýznamnější, a připíšeme ji bodu. Později provedené operace se budou vztahovat k této orientaci. Tím je zajištěna invariance vůči rotaci.
3. Výpočet deskriptoru, který nese informaci o okolí bodu a udává například jak vypadá po změně orientace v pravém měřítku.

*SURF* se nepatrně liší v první a třetí fázi. Pro zrychlení výpočtů by měl být v prvním a třetím kroku použit integrální obraz. Dále se pro výpočetní operace aplikuje Haarova vlnka pro vytvoření vhodného deskriptoru (bude vysvětleno později).

Pro porozumění *SURF* je dobré začínat s podrobným vysvětlením *SIFT*. *SIFT* je i detektor i deskriptor klíčových bodů. Je tedy důležité tyto pojmy nejdříve definovat. Jedná se o matematické operace, kterými jsou nejdříve vyhledané klíčové neboli charakteristické body v různých oblastech obrazu a poté jsou matematicky popsány deskriptorem. Detekování charakteristických bodů se pokouší o vyhledání těch rysů, které lze později popsat deskriptorem a porovnat s body v jiném obrázku, u nichž lze potvrdit shodu. Detektor *SIFT* je v podstatě více měřítková podoba klasických rohových detektorů jako je Harrisův

detektor rohu nebo Hessián (orig. Harris/Hessian corner detector) a má schopnost automatického doladění měřítka. Rohy jsou optimální body zájmu, protože jsou opakovatelné a jedinečné. Optimální detektory jsou invariantní k velikosti a rotaci. To znamená, že detekce funguje i tehdy, když je snímek oddálen, nebo naopak přiblížen, anebo když je snímek otočen. Harrisův detektor sám o sobě není invariantní vůči změně v měřítku (když se zvětší snímek rohu, tak se místo rohu budou detekovat hrany), ale je invariantní vůči rotaci, a to díky elipse, která je grafickým znázorněním vlastních čísel matice derivací. Když dojde k rotaci, tak se sice natočí elipsa, ale její velikost (hodnoty vlastních čísel) se nezmění. Harrisův detektor může být vylepšen použitím Laplace na lokální maxima, aby byl invariantní ke změně velikosti. Naproti tomu SURF používá Hessián detektor, který je determinantem Hessovy matice druhých parciálních derivací skalární funkce viz níže uvedená matice.

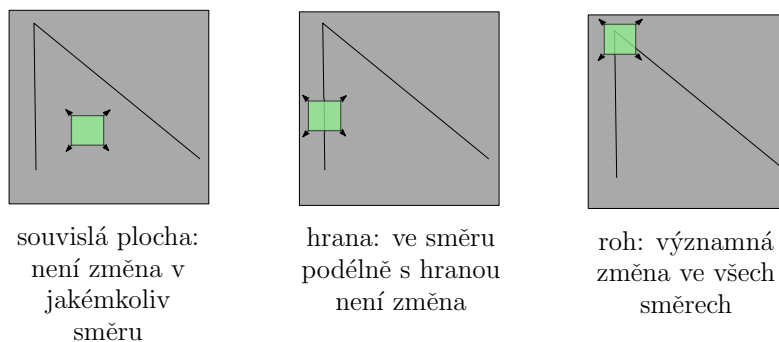
$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{yx}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

Kde:

- $L_{xx}(x, y, \sigma)$  - je konvoluce (vynásobení) snímku s vhodným Gaussovským kernelem (tj. druhou derivací Gausiánu), totéž platí pro  $L_{xy}(x, y, \sigma)$  a  $L_{yy}(x, y, \sigma)$

Pro výpočet determinantu se aplikovali čtvercové filtry  $D_{xx}, D_{yy}, D_{xy}$ , které ořízli Gaussovské kernely. Jenže tím se hodnota Hessiánu zkreslí, takže pro kompenzaci se Hessián počítá dle rovnice 2.4. Váhování je uvedeno jako  $w = 0,9$  v [5] a pro efektivitu výpočtu je zanecháno jednoduché. Takto je znatelně zvýšena výpočetní rychlost Hessiánu.

$$Det(H) = D_{xx}D_{yy} - (wD_{xy})^2 \quad (2.4)$$



**Obr. 2.3:** Detektor rohu

Podstatnou operací v SIFT a SURF je rozdíl Gausiánů *DoG* (orig. Difference of Gaussian) s automatickým výběrem měřítka, neboli velikosti Gausiánu. Nejdříve se sestaví tzv. *scale-space*, pro zavedení nezávislosti na měřítku se přidává tento třetí rozměr, kde vstupní snímek je filtrovaný při různých měřítkách. Scale-space je pyramida, kde dva po sobě jdoucí snímky jsou provázané změnou měřítka nebo rozdílem tak, že se změnila

velikost nízko propustného filtru. K měřítku, při kterém má snímek určitou velikost, je přiřazena skupina oktáv. V oktávě jsou mezi snímky velké rozdíly velikostí Gaussovského filtru. U SIFT se scale-space vytváří aplikací Gausiánu v rámci oktávy několikrát a oktáv se vytváří tolik, kolik je potřeba. Pro každou oktávu se snímky zmenšují. V rámci jedné oktávy se postupně přibližuje k další oktávě opakovanou filtrací vstupu Gausiánem o pevné šířce, dokud se nedojde měřítkem k další oktávě. DoG detekuje v scale-space charakteristické body podobajících se skvrnám a to tam, kde vychází vysoká hodnota rozdílu. DoG se vypočítá dle rov. 2.5.

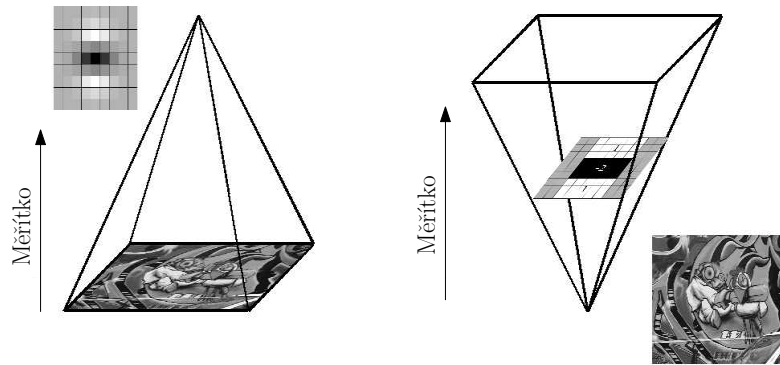
$$DoG(x, y, \sigma) = L_1(x, y, k_1\sigma) - L_2(x, y, k_2\sigma) \quad (2.5)$$

Kde:

- $\sigma$  - měřítko
- $L_1$  - první snímek
- $L_2$  - druhý snímek

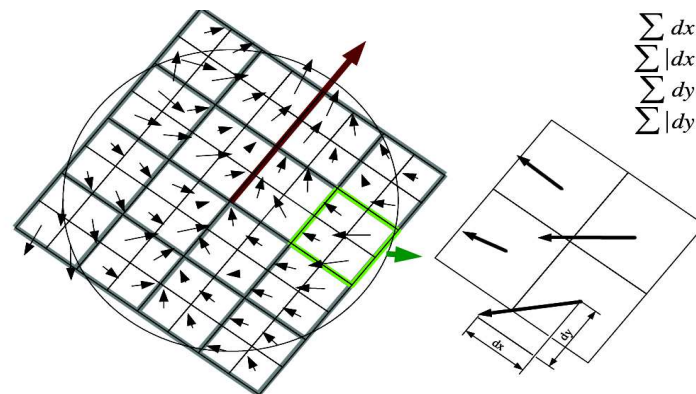
Rozdíl Gausiánu je aproximací Laplace Gausiánu, což je odečtení rozmazané verze originálního snímku  $L_1$  se snímkem méně rozmazaným  $L_2$ . Autoři zmíněného článku poznamenali, že malé Gausiány (jako ty používané v SIFT) se mohou dobře aproximovat s čtvercovou integrací čtvercovým filtrem (také známé jako čtvercové rozmazání). Výhodou čtvercových filtrů je, že je lze velmi rychle získat díky použití integrálního obrazu. Scale-space SURF je aproximací původního scale-space ve formě LoG Laplace Gausiánů. Nepotřebujete tedy jen Gausiánem rozmazané snímky, ale jejich derivace a difference. Takže se může zanechat v pozadí aproximace čtvercem Gausiánu a to proto, že se nejdříve odvodí Gausián tolikrát kolik potřebujeme. Postupně nám zůstane sada Haarovských rysů (vlnek) používaných pro výpočet deskriptoru. Oblast kolem klíčového bodu je vzorkována a sestavena do vektoru, takzvaného deskriptoru, což je provedeno pro identifikaci klíčového bodu. Když je známo umístění a velikost oblasti (odvozená z měřítka) je možné vypočítat deskriptor. SIFT je výborný v přiřazení místně příbuzných kousků snímku, ale nevýhodou je výpočetní náročnost a zdlouhavost, která je z podstatné části způsobena výpočtem histogramů gradientního směru (pro deskriptor). Pro vytvoření scale-space u SURF se proces o něco zrychlí díky užití integrálního obrazu. Počítá se přímo snímek filtrovaný při každém měřítku bez použití výsledku z předešlého měřítka. Výpočty ze SIFT jsou v SURF urychleny užitím aproximací funkcí.

Pro vytvoření deskriptoru je okolo klíčového bodu vybráno okno 16x16. Klíčový bod neleží přímo na pixelu, ale na jeho okraji. Toto okno je rozděleno na šestnáct oken 4x4. Pro každé okno 4x4 je vygenerován histogram o osmi přepážkách. Každá přepážka postupně 0 až 44, 45 až 89 atd. Do přepážek je vložen gradient orientace. To je provedeno pro všechny bloky 4x4. Nakonec je 128 (osm přepážek v každém ze šestnácti oken) získaných hodnot normalizováno. Deskriptor je potom vektor  $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ .



Obr. 2.4: Scale-space [Převzato z [5]]

Pro minimalizaci potíží se použije prahová hodnota 0,2, která je odečtena od orientace klíčového bodu. Hodnota  $\sigma=0,2$  je nejnižší hladina pro odezvu.



Obr. 2.5: Deskriptor [Převzato z [5]]

Vyhledávání charakteristických bodů slouží i k jiným účelům, jako je například spojování scén z více snímků do jednoho.

## 2.2 Lokalizace pomocí fotocitlivých součástek

Detekci a lokalizaci pohybujících se těles lze dále také realizovat pomocí senzorů. Senzorem se rozumí elektrická součástka, která reaguje na působení fyzikálního jevu elektrickým signálem. Existuje mnoho druhů senzorů, které mohou být například indukčního, kapacitního, magnetického, fotoelektrického, ultrazvukového charakteru apod. Typ senzoru vybíráme na základě dvou kritérií. Za prvé dle materiálu. Tělesa, která budou detekována jsou z určitého materiálu. Vlastnosti materiálu určují vybraný druh senzoru. Za druhé, dle vzdálenosti v jejímž rozsahu budeme chtít těleso detekovat.

Fotoelektrické senzory mají široký rozsah využití, jako například detekce přítomnosti předmětu, vzdálenost k předmětu, jeho umístění, nebo detekovat barvu předmětu. Z těchto naměřených údajů lze dále počítat předměty nebo je roztrdit podle barvy. Díky

mnoha způsobům využití je tento typ senzorů populární volbou v průmyslové automatizaci. Lokalizace musí mít spolehlivé výsledky, proto je třeba brát v potaz faktory, které mohou vnést do výsledků chyby, jako je například detekování světelných signálů nevyslaných přítomným emitorem anebo odraz světla od předmětů.

Fotoelektrický senzor reaguje na změnu intenzity vyslaného světla nebo na změnu úhlu světla, proto by se neměl používat v příliš zaprášených prostorech, jelikož by se na nich mohla usadit vrstva, která znemožní průnik paprsku. Fotoelektrické snímání vyžaduje dvě základní součástky - vysílač a přijímač. Vysílač vysílá světlo a přijímač světlo přijímá. Fotocitlivé součástky, které se používají jako přijímač světla se nazývají fotodetektory. Fotodetektozem může být jedna z následujících součástí:

- Fotodiody - jsou polovodičové součástky pracující ve dvou režimech. Mohou pracovat v režimu fotovoltaičném, tj. dopadající záření vytváří napětí na PN přechodu. Druhý režim, fotovodivostní režim, je když se mění efektivní odpor vlivem světla.
- Fototranzistory - jsou polovodičové součástky, většinou z křemíku, pracující v režimu fotovoltaičném. Dopadem fotonů na bázi fototranzistoru se vytvoří proud bází, který je úměrný intenzitě světla.
- Fotorezistory - využívají se v režimu fotovodivosti, tj. dopadající záření generuje v polovodiči volné nosiče náboje a ty zvyšují vodivost. Tento proces se také nazývá vnitřní fotoelektrický jev.
- Fotočlánky - využívají se v režimu fotovoltaičném. Převádí světlo na elektrický signál. Nejčastěji se využívá v luxmetru a jasoměru.

Podle fyzické orientace emitru vůči přijímači se rozlišují různé režimy snímání:

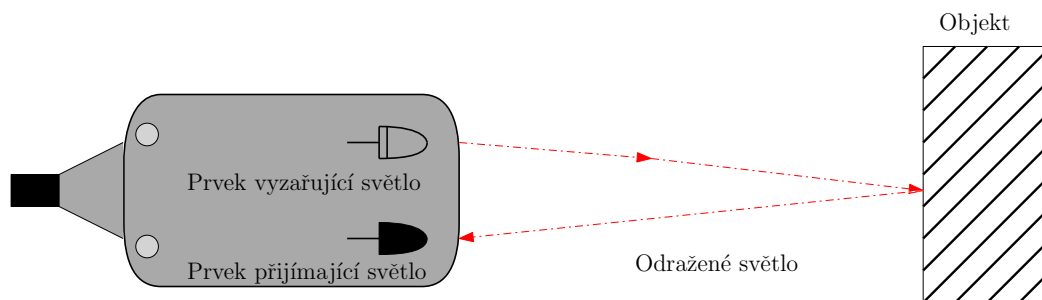
- jednocestná závora, známo pod anglickým názvem *Through-beam*, viz obr. 2.10,
- retroreflektivní senzory neboli reflexní senzory s reflektorem, viz obr. 2.7,
- reflexní senzory, viz obr. 2.6.

Ke stavbě fotoelektrických senzorů se používají fotocitlivé součástky. Optoelektronické součástky využívají vnitřního fotoelektrického jevu, tj. kvantově mechanické vlivy světla na elektrické materiály, a to hlavně na polovodiče. Základem funkce je přeměna elektrické energie na elektromagnetické záření a naopak. Většinou se jedná o záření od ultrafialového, viditelného světla až po infračervené záření. U polovodičových optoelektronických součástkách je chování závislé na materiálu. Každý materiál je jinak citlivý na různé vlnové délky. Například křemík je nejcitlivější na 400 nm až 1100 nm a germánium na 600 nm až 1600 nm. Jako emitory se nejčastěji používají LED diody, infra-LED nebo laserové diody. Fyzikální podstatou optoelektronických součástí se do větší hloubky zabývá autor v [10].

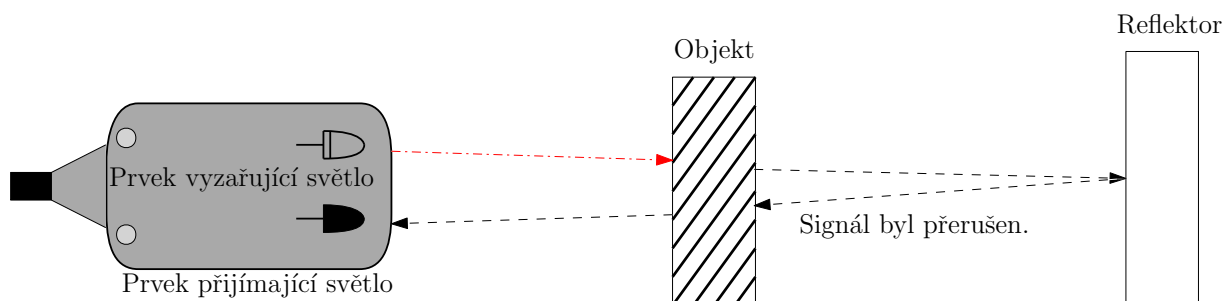
## 2.2.1 Využití reflexních senzorů

Reflexních senzorů je celá řada, z nich nejběžnější jsou reflexní senzory s reflektorem, jeho uspořádání je zobrazeno na obr. 2.8. Reflektor může vypadat různě:

- Varianta se zrcátkem. Světlo se odráží od zrcátka směrem zpět k senzoru.
- Varianta s polarizačními filtry je praktická k odstranění problémů se zrcadlením. Cílem je odrážet polarizované světlo. Před vysílač je umístěn lineární filtr a před přijímačem je vodorovný filtr a odrazka světlo odráží po polarizaci, takže původně vertikálně polarizovaná složka se po odrazu otočí o  $90^\circ$  a vrací se opět vertikálně polarizovaná složka k přijímači. Kdyby se světlo odrazilo o objekt, tak se neotočí o  $90^\circ$  a neprojde filtrem k přijímači.
- Existují také reflexní senzory, které dokáží detekovat i průhledný objekt. Detekují také intenzitu lesknutí objektu, protože světlo se odráží různě od různě lesklých povrchů. Z reflexních senzorů je často používáno provedení nazývané difuzní senzor. Odvětví využívající difuzní senzory se zabývají mapováním okolí, tvoření virtuální reality a protéz.



Obr. 2.6: Reflexní model



Obr. 2.7: Retroreflexivní model

Senzory v reflexním složení z laserového vysílače patří mezi nejmodernější zařízení pro sbírání geo-dat, této moderní metodě se říká laserové skenování. Laserové skenování je

také jedna z důležitých částí robotiky. Pro řízení robotů je často potřeba mapovat okolí nebo detekovat překážky. Pro laserové skenování je používán laser jako emitor. Laser je známá zkratka pro anglický název *Light Amplification by Stimulated Emission of Radiation*. Lasery bývají nejčastěji červené, ale také ultrafialové anebo infračervené. Obojí dále mohou být typy laseru s plynem nebo laseru s pevnou látkou. Toto poslední rozdělení je podle skupenství aktivního prostředí laseru. V laseru dochází stimulovanou emisí ke vzniku koherentního záření. Emise je přechod elektronů z vodivostního pásu do valenčního pásu, když dopadají stimulující fotony, které musí mít stejnou energii, fázi a polarizaci, jako foton vyzářený, jak bylo popsáno v [9]. Laser je lepší variantou oproti jiným emitorům, protože vyzařují koherentní monochromatické světlo, které má jednu specifickou vlnovou délku ve stejné fázi, a proto je světlo vyzářeno dál a silněji než od jiných druhů emitorů. Pro měření vzdáleností jsou to určitě důležité aspekty. Jednou z aplikací laseru je například LIDAR. *LIDAR* nebo taky zvané *LADAR* (orig. Light/Laser Detection And Ranging), je jednou z metod 3D laserového skenování. Používá se na měření vzdálenosti k cíli tak, že je cíl nasvícen pulsním laserovým světlem a ze senzorem naměřených odražených pulzů lze dále provádět výpočty. LIDAR senzory jsou umísťovány na automobily za účelem detekovat překážky nebo pro mapování okolí.



Obr. 2.8: Model skeneru na automobilu [Převzato z [11]]

Laserové skenování se dnes často používá k rekonstrukci budov do 3D modelů. Laserovým skenováním se vytvoří digitální model terénu buď s velkou hustotou naměřených bodů nebo malou, které tvoří *mračno bodů* (orig. point cloud), které lze převést do CAD modelu ve 2D či 3D. Ke každému bodu z mračna lze přiřadit informaci o barvě a intenzitě odraženého světla od materiálu. V geodézii se podle modelu indikují povrchové nerovnosti vozovek, vizualizace složitých staveb a objektů a pro dokumentaci architektonicky významných budov. Dalším případem, kdy byl LIDAR použit, bylo pro mapování povrchu Měsíce při pilotovaném letu programu Apollo, pro více informací viz [4].

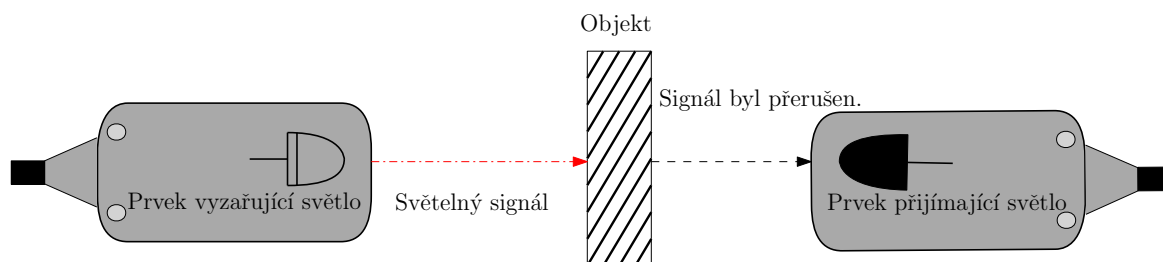




Obr. 2.9: Mračno bodů |Převzato z [3]|

## 2.2.2 Využití jednocestné závory

V uspořádání jednocestné závory je přijímač umístěn v ose s emitorem. V tomto režimu je objekt detekovaný, když je paprsek přerušen. Jednocestné závory jsou optimální typ senzoru, protože bez ohledu na materiál, barvu nebo povrchovou úpravu tělesa, je těleso spolehlivě detekováno. Při větších vzdálenostech je nejlepší volbou právě tento senzor. Někteří výrobci udávají, že lze snímat až na vzdálenost 70 metrů. Jednocestnou závorou nelze detekovat transparentní tělesa, ale zrcadlící tělesa ano. Provedení senzoru je cenově nenáročné. V průmyslu se senzor většinou využívá pro kontrolu přítomnosti předmětů nebo počtu předmětů. Podle obrázku 2.10 vidíme, že se jedná o dvě součástky a to emitor a přijímač. Jakmile paprsek vyslaný emitorem není snímán přijímačem, je aktivován výstup přijímače. Na základě aktivního výstupu lze dalším zařízením signalizovat, že bylo detekované těleso nebo je tím spuštěna další funkce, jako otevření automatických vchodových dveří do budovy. Senzory se často používají pro detekci překážky při zavírání garážových vrat, při detekci překážky se zastaví zavírání a tak zamezí vzniku škod. Podmínkou pro správnou funkci je správné seřízení senzoru. Senzor považujeme za seřízený, když na přijímač dopadne maximální množství světla, takže jsou s emitorem v optické ose. Vyzářený paprsek může být příliš veliký na to, aby byla detekována malá tělesa nebo tělesa s nízkým profilem. To lze vyřešit nasazením clony před čočku a nastavením optimálně velkého paprsku v otvorech v cloně, nebo použitím laseru jako emitoru. Některé moderní jednocestné závory umí vyzařovat viditelné i infračervené světlo a umí měřit plochu předmětu.

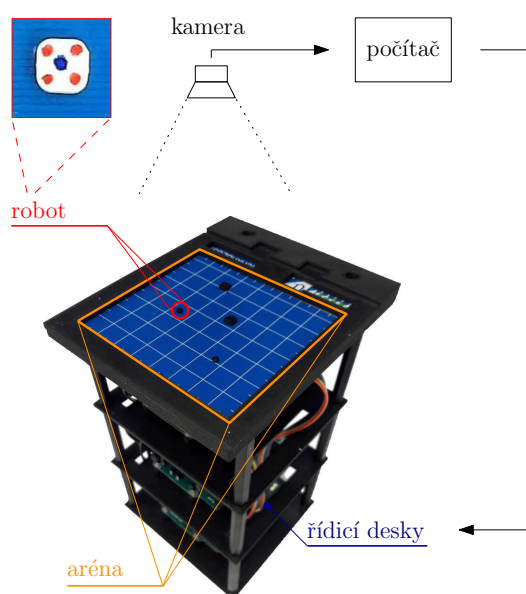


Obr. 2.10: Through-beam režim

# 3

## Praktická část

Pro realizaci optické lokalizace byly vyzkoušeny dvě odlišné metody. Ze všech možných metod jsme se rozhodli použít lokalizaci kamerou a lokalizaci pomocí senzorů, abychom porovnali, která z metod je přesnější a časově méně náročná. Pro první metodu byl jako vyhledávaný objekt použit malý robot o velikosti 3x3 mm, který se pohyboval po aréně 100x100 mm. Pro lepší představu uspořádání platformy zvaného Scarabeus, ve které se měřilo, je fotografie na obr. 3.1. Pro detailnější popis fungování robota Scarabeus je vhodná práce [8]. Nad arénou byla umístěna webkamera s rozlišením až 4K (3840x2160). Přes jednodeskový počítač Arduino UNO se dala předem nastavit či v reálném čase řídit trajektorie pohybu robota. Pro vyzkoušení lokalizace bylo natočeno video robota v pohybu, které bylo dále zpracováno čtyřmi odlišnými algoritmy v programu Matlab verze R2018b. Druhou otestovanou metodou bylo navrženo zařízení ve formě senzoru jednocestné závory pro lokalizaci malých těles.



Obr. 3.1: Model arény s robotem-Scarabeus |Převzato z [7]|

### 3.1 Aplikace kamery a algoritmů

Pro zpracování snímků a vyhledání robota pomocí algoritmů byl napsaný program v Matlabu. Součástí novějších verzí Matlabu je *Image Processing Toolbox* s funkcemi pro zpracování snímků. Toolbox podporuje například geometrické operace, sousední a blokové operace, lineární filtrace a konstrukce filtru, transformace, analýzu a vylepšení obrazu, binární operace obrazu a výpočty regionu zájmu. Při práci s obrazem je tento toolbox velmi užitečný.

Postup lokalizace kamerou lze shrnout do tří základních kroků. Nejdříve bylo pořízeno video, které bylo nahráno do Matlabu a v něm byl na každý snímek z videa aplikován algoritmus pro vyhledávání. Ještě mezi těmito kroky byly vykonané určité úpravy nebo procesy, které se pro každý aplikovaný algoritmus mohou již trochu lišit. Prvotní úprava snímků z videa v Matlabu je jejich převedení na černobílý obraz, se kterým je snazší pracovat v tomto prostředí. Převodem z barevné verze, která je v Matlabu trojdimenzionální maticí, do černobílé byla vytvořena dvoudimenzionální matice snímku. Pro operaci byla použita funkce v Matlabu nazvaná **rgb2gray**. Výstupní snímek je stejného datového typu, jako byl snímek RGB. Z výstupního snímku získáváme matici obsahující hodnoty stejného datového typu nesoucí informaci o intenzitě jednotlivých pixelů. Hodnota může být od 0 do 1 například pro datový typ *double*. Snímek, ve kterém byl robot vyhledáván, byl pro danou polohu kamery oříznut ve dvou použitých rozlišení na oblast o šířce 412 pixelů a výšce 438 pixelů v rozlišení 640x480 a oblast o šířce 1000 pixelů a výšce 980 pixelů v rozlišení 1920x1080. Pro oříznutí byla použita funkce **imcrop**. Oříznutí bylo provedeno, aby množství artefaktů bylo redukováno. Při měření času trvání jednotlivých algoritmů je ohraničena samotná funkce algoritmu do funkcí **tic** a **toc**. **Tic** spouští časové stopky a **toc** vypíše uběhlý čas od použití **tic**. Ostatní operace ve skriptu, jako převod snímku na černobílou, byly vynechány z měření času. Naměřené časy, pro všechny snímky ve videu, byly uloženy do vytvořeného vektoru **time**. Určující informací pro porovnání rychlostí algoritmů je průměr z vektoru **time**.

Tři základní kroky byly shrnuty do jednoho celku, a tím je hlavní skript v Matlabu. Skript byl napsán uživatelsky přívětivě tj. je jednoduché pochopit jeho fungování. Ve skriptu, který byl nazván **Lokalizace**, je možné zavolat jednu ze čtyř funkcí. Funkce byly pojmenovány podle aplikovaného algoritmu. Funkce a jednotlivé algoritmy jsou popsány v následujících podkapitolách. Cílem bylo změřit doby trvání lokalizací každým algoritmem a porovnat, který byl nejrychlejší. Tyto naměřené doby se ale mohou lišit pro různé rozlišení videí. To vedlo k použití videí ve dvou odlišných rozlišení, jedno ve vysokém rozlišení 1920x1080 pixelů a druhé v nízkém 640x480 pixelů. Rozlišení snímků použitých v algoritmech, jako například s robotem a bez robota, se musí shodovat. Rychlost videa byla nastavena na 60 snímků za vteřinu.

Při práci s algoritmy byla potřeba z videa vzít a uložit první snímek, na kterém je jen aréna bez robota. Dále bylo potřeba z jakéhokoliv snímku vyříznout šablonu robota. Šablona je referenčním blokem, se kterým budou postupně porovnávány bloky v snímku.

Důležité je, aby byla šablona ve shodném rozlišení jako používané video. Ve skriptu je cyklus procházející video snímek po snímku, dokud zbývají snímky. V cyklu jsou volány funkce konkrétních algoritmů. Výstupem po proběhnutí skriptu jsou souřadnice pozice robota  $x$  a  $y$ . Pro vizuální ověření správné lokalizace bylo v Matlabu nahráno video, ve kterém je robot v každém snímku označen a je zakreslena jeho trajektorie červenou souvislou křivkou.

### 3.1.1 Základní aplikace algoritmu Block matching

BMA, jak už bylo zmíněno v teoretické části, je jednoduchý a efektivní algoritmus pro lokalizaci. Jeho úplně základní stavba úplného vyhledávání je sice přesná, ale není příliš rychlá. Je to výpočetně náročné, kvůli několikrát opakovanému kroku porovnávání bloku pixelů z originálního snímku s blokem pixelů v novém snímku. Pro snadnou aplikaci algoritmu v Matlabu byl napsán jako funkce zvaná **BMA**, která je volána ve skriptu **Lokalizace**. Funkce měla tři vstupní parametry a jeden výstupní. Vstupními parametry byly matice plná jedniček stejně velká jako matice snímku, matice snímku a matice šablony, což je referenční snímek obsahující jen robota. Pro určení míst, kde má být provedeno vyhledávání, je použita námi vytvořená matice vyplněna jedničkami. Protože jsou jedničky všude, je vyhledávání provedeno na všech místech. Jak již bylo popsáno v předchozí části, barevný snímek z videa je převeden do dvoudimenzionální matice snímku. Operace vykonaná uvnitř funkce je odečítání matice šablony robota od matice snímku z videa. Výsledkem výpočtu jsou pozice s minimálními rozdíly uložené do matice, ze které průměr je dále považován za pozici robota. Při měření času trvání tohoto algoritmu je zapouzdřena samotná funkce BMA do funkcí **tic** a **toc**. Měření času trvání výpočtu a princip funkcí **tic** a **toc** byly popsány v předchozí části.

Průměr časů lokalizace s použitím úplného vyhledávání BMA v originálních rozlišeních před ořezem 640x480 vyšlo na 1,15 vteřin a v 1920x1080 na 16,54 vteřin. Je zřetelný rozdíl časů v závislosti na rozlišení snímků. S počtem pixelů přímo úměrně narůstá doba trvání vyhledávání. Pro nás se stává úplné vyhledávání BMA nezajímavou metodou, protože je časově velmi náročná. S rostoucím rozlišením se časová náročnost více projevuje. Algoritmus ale není úplně zbytečný. Jeho základní stavba byla použita i v urychlených algoritmech.

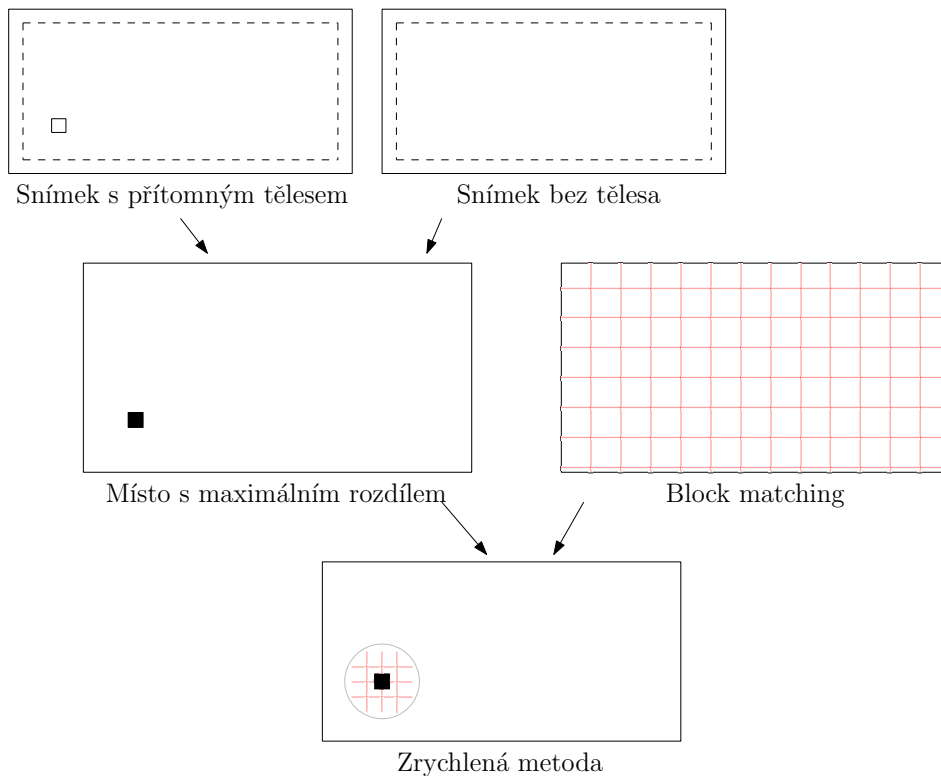
### 3.1.2 Hledání kolem maxima rozdílu s referenčním snímkem

Je žádoucí, aby proces lokalizace byl co nejvíce urychlen. Pro tento účel bylo přeskočeno porovnání každého bloku s každým blokem a vyhledávalo se až v okolí maximálního rozdílu vůči snímku bez tělesa. Byla vytvořena funkce zvaná **search\_max\_BM**, ve které probíhal jen samotný výpočet pro vyhledání robota na snímku. Vstupními parametry funkce byly matice snímku z videa, matice snímku bez robota a matice vyřiznuté šablony. Při aplikaci tohoto algoritmu je potřeba mít matice ze snímku samotné arény bez robota.

V Matlabu byly opět barevné snímky oříznuty a převedeny na černobílé (viz kap. 3.1). Oříznuté snímky arény s robotem a bez něho musely být stejně veliké.

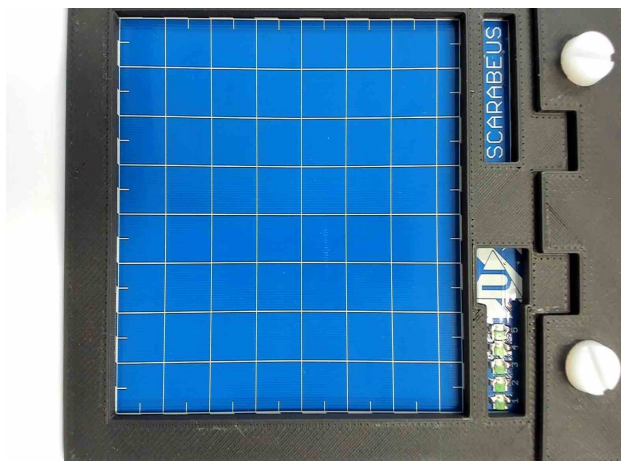
Matrice prázdné arény byla odečtena od matice snímku arény s robotem. Při aplikaci metody prahování, se zvolenou prahovou hodnotou 60, byla takto rozdělena matice na dvě části. Část kde byly hodnoty 255 byla oblast, kde rozdíl vyšel víc než 60 a oblast s 0, byla oblast kde rozdíl vyšel menší než 60. Tak byla dohledána oblast maxima rozdílu, na kterou byl potom aplikován klasický BMA úplného vyhledávání. Výstupními proměnnými jsou pozice robota v souřadnicích  $x$  a  $y$  pixelů. Tato funkce BMA byla aplikována na různá rozlišení snímků, nižším rozlišení 412x438 pixelů a vyšším 1000x980 pixelů. Velikost oříznutého snímku při nižším a vyšším rozlišení byla shodná, jako v předchozím případě při aplikaci BMA. Po přípravě snímku na lokalizaci se v tomto skriptu zavolá funkce BMA kolem maxima rozdílu matic prázdného snímku a snímku s robotem. Výstupem funkce byl výsledek označující pozici. Ve skriptu **Lokalizace** byla výsledná pozice funkce dále graficky zobrazena na snímcích ve formě čtverce kolem robota a uprostřed něho červenou hvězdou označující střed robota.

Opět byla změřena doba trvání výpočtu samotného algoritmu. **Tic** a **toc** byly umístěny kolem volání funkce **search\_max\_BM**, aby do doby nebyly zahrnuté časy trvání úprav snímku a označení lokalizovaného tělesa. Průměrná doba vyšla 0,06 vteřin pro video s nižším originálním rozlišením (640x480) a pro video s vyšším originálním rozlišením (1920x1080) na 1,67 vteřin.



**Obr. 3.2:** Grafický zobrazení principu BMA s hledání kolem maxima rozdílu

Prázdňá aréna



Šablona robota

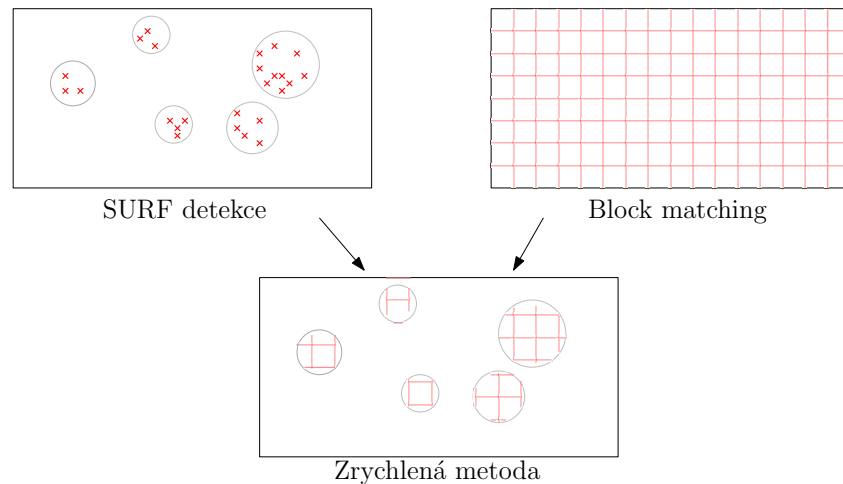


Obr. 3.3: Snímek prázdňé arény a šablona robota

### 3.1.3 Hledání v okolí zajímavých bodů

Následující algoritmus je modifikovaný BMA, tak aby se ušetřilo výpočetních kroků. Princip vychází z algoritmu BMA, jenže nejdříve byly rychlým SURF algoritmem dohledány zajímavé oblasti. V Matlabu je pro to přímo knihovní funkce **detectSURFfeatures**, která byla aplikována. Výsledkem jsou klíčové body, ze kterých je možné si zvolit počet nejsilnějších klíčových bodů. Optimální počet byl zvolen dvacet klíčových bodů, kolem kterých byl poté uplatněn BMA. Díky minimalizování rozsahu oblasti, kde byl aplikován BMA, se výpočet podstatně zrychlil. Vstupními parametry funkce byly opět vhodně upravené snímky. Snímek s arénou je stejně oříznutý, jako v předchozích funkcích a šablona robota je používána stejná. Opět jsou převedeny do černobílé, a tak je vytvořena matice snímků. Následně porovnáváme matice snímku samotného robota. Jak již bylo provedeno u hledání kolem maxima, tak se ve skriptu zavolá funkce BMA se vstupními parametry, v tomto případě jen matice snímku **imgbw** a matice šablony robota **template**. Výstup funkce je uložen do proměnné ve skriptu, jako pozice robota, kterou vykreslíme na snímku z videa.

Změřili jsme čas opět jen trvání funkce **SURF\_BM**. Výsledné průměrné časy vyhledávání malého tělesa s BMA zkombinovaného se SURF byly 0,05 vteřin při nízkém rozlišení (412x438) a 2,48 vteřin při vysokém rozlišení (1000x980). Pro určení pozice robota na aréně 100x100 milimetrů, byl určen horní levý roh jako nulová pozice a od něho se odečetla pozice robota. Výslednou hodnotou v pixelech bylo zapotřebí převést na milimetry. Pro určení počtu pixelů v jednom milimetru byla vypočítaná vzdálenost v pixelech mezi bílými čarami na aréně, protože v reálu je vzdálenost mezi nimi jeden milimetr. Díky tomu lze je použít takto jako referenční milimetr.



Obr. 3.4: Grafický znázornění principu algoritmu BM urychleného s užitím SURF

### 3.1.4 Aplikace integrálního obrazu a detektoru rohů

Jelikož se uvádí integrální obraz za výpočetní zrychlení, zkusili jsme také tuto metodu. V Matlabu byla použita knihovná funkce **integralimage** na černobílém snímku arény a na výsledek funkce byl aplikován Harris detektor rohů. Pro detektor rohů byla použita knihovná funkce **detectHarrisFeatures**. Výsledkem této funkce jsou body, které lze vykreslit do již zobrazeného snímku v Matlabu a lze zvolit počet bodů s nejsilnější odezvou, se kterými se dále počítalo. Při volbě jedné z odezev se ukázalo, že byl detekován spodní levý roh. Pro určení středu robota byla pouze upravena pozice rohu podle velikosti šablony robota. Úpravou rohu znamenalo jeho přesun na pozici danou odečtením od rohu poloviční velikosti šablony. Výpočet ve vysokém rozlišení trval 0,18 vteřin hledání a v nízkém rozlišení trval 0,04 vteřin. Použitím tohoto algoritmu bylo dosaženo nejkratší doby vyhledávání robota.

Výsledné pozice  $x$  a  $y$  byly pro každý snímek z videa uloženy do matice o velikost 420 řádků (počet snímků ve videu bylo 420) a dvou sloupců (pro  $x$  a  $y$ ), porovnáním  $x$  a  $y$  integrálního obrazu s  $x$  a  $y$  ostatních algoritmů nám říká, že je integrální obraz nepřesný. Z měření vyšel průměr o kolik se liší integrální obraz od SURF s BMA, který je zobrazen v tab. 3.4 a v tab. 3.5. Rozdíly pozice  $x$  a  $y$  nám řeknou, o kolik se každá souřadnice liší. Z trojúhelníku, tvořeném výslednou pozicí z integrálního obrazu a pozicí ze SURF s BMA, je možné určit úhlopříčku, o kterou se liší pozice. Pro každý rozdíl  $x$  a  $y$  souřadnic dvou porovnaných algoritmů byla Pythagorovou větou vypočítána úhlopříčka. Ze všech vypočítaných úhlopříček vyšla průměrná hodnota jako celková chyba integrálního obrazu. Rozdíly pozic mezi integrálním obrazem a BMA jsou v tab. 3.8.

Integrální obraz je velmi rychlý algoritmus, ale není tak přesný. Metoda, která by mohla využívat rychlost integrálního obrazu a přesnost BMA by byla optimální. Nejdříve by byly integrálním obrazem dohledány klíčové body a v okolí o určené velikosti by se kolem nich využil BMA.

## 3.2 Využití navrženého zařízení pro lokalizaci

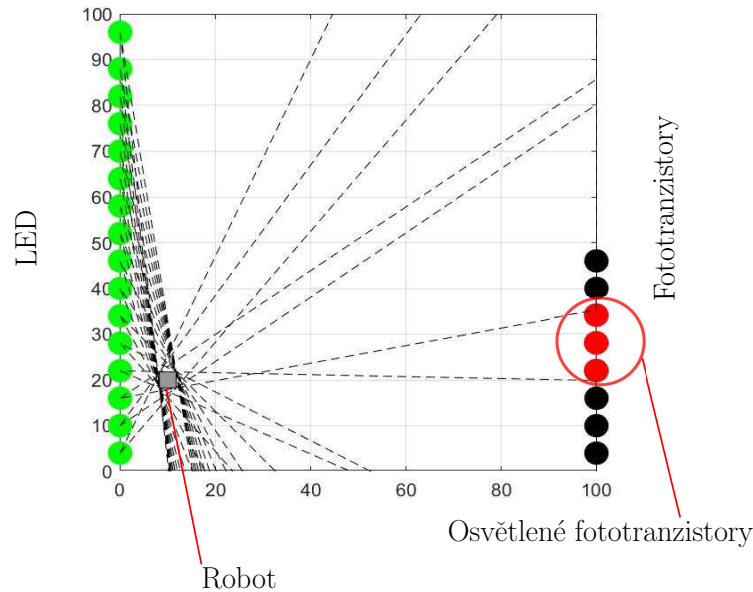
Další možností, která připadala v úvahu, byla lokalizace pomocí optoelektrického senzoru. Bylo vyrobeno zařízení podobné optoelektrickému senzoru typu jednocestné závory, což zahrnuje dvě části a to emitore a přijímač. Při výběru spektra vlnových délek vyzařovaných emitorem bylo předpokládáno, že v místnosti, kde proběhne lokalizování tělesa, mohou být zapnutá světla, anebo může svítit zvenku denní světlo. Podle možností bylo vybráno infračervené spektrum. Je jasné, že emitore i přijímač musí být navrženy tak, aby reagovali na stejné vlnové délky. Emitore v zařízení je zastoupen infračervenými LED a přijímač infračervenými fototranzistory. Jedinou nevýhodou infračerveného záření je, že pro lidské oko je neviditelné. Naštěstí v moderních mobilních telefonech je zabudovaná kamera citlivá i na jiné vlnové délky než viditelné, díky které je na telefonu vidět infračervené světlo. Jelikož se jedná o lokalizace malého tělesa pohybujícím se po malé ploše, jsou tyto součástky pro malou vzdálenost dostačující.

Z hlediska mechanických vlastností byly vybrány součástky THT a to proto, aby se snadno se součástkou případně manipulovalo i po osazení na DPS. Zařízení mělo fungovat na základě principu senzoru jednocestné závory, ale otázkou zůstalo, kam umístit LED a v jakém pořadí je rozsvěcet. Pro některou z možných pozic tělesa existuje více kombinací zapnutých LED a senzorů, které by mohly indikovat danou pozici robota. To může nastat kvůli tomu, že nebyla detekována změna stínu robota, kvůli nedostatkům součástek. Chceme unikátní kombinaci pro konkrétní pozice robota. Závisí to na tom, které LED jsou rozsvícené. Pro určení vhodného umístění LED a fototranzistorů byla vytvořena simulace všech variant viz obr. 3.5. Skript simulace vrací matice rozsvícených LED a zastíněných fototranzistorů. Matice jsou poskládané do třídímní matice, protože pro každou pozici robota je matice rozmístění LED a senzorů. Jelikož je počet možných pozic robota 10 000 (velikost arény je 100x100 milimetrů), je i takto velký počet matic v třídímní matici. Z třídímních matic bylo zjištěno kolik z nich není unikátních. Suma variant matic, co jsou si podobné, se musí co nejvíce blížit k nule. Když byla suma stejných matic příliš velká (ideálně nulová), vše bylo zopakováno pro jiné náhodně generované umístění LED. Výpočet unikátních pozic byl velmi zdlouhavý, proto byl, pro urychlení výpočtu, využit školní výpočetní server zvaný Edison. Suma pro různé umístění LED a fototranzistorů zvolené pro simulaci vycházela celkem vysoko (stovky tisíc). Zkusili jsme unikátní polohu LED a fototranzistorů vyhledat pomocí genetických algoritmů viz kap. 3.2.1.

### 3.2.1 Návrh a výroba DPS

DPS byla navržena v softwaru pro návrh desek plošných spojů-*Altium designer*. Postupně bylo vše zdokumentováno od návrhu schématu, po návrh umístění součástek na desku a až po vyexportování dokumentů potřebných k výrobě desky. Obrázky návrhu DPS jsou uvedeny v přílohách práce. Při návrhu schématu bylo nutné si uvědomit několik pravidel





**Obr. 3.5:** Grafický znázornění simulace zastínění fototranzistorů při některé z možných pozic robota

pro tvorbu DPS. Musí být řádně označené všechny součástky a jejich hodnoty, a to i zem a napájení. Vše musí být vodivě propojené. Hodnoty součástek musejí odpovídat elektrickým zásadám. Při navrhování desek byl vytvořen projekt v programu, ve kterém byly tři desky.

První a také nejobsáhlejší deska byla s mikroprocesorem. Byl použit mikroprocesor s osmibitovým jádrem HSC08 od výrobce NXP. Mikroprocesor má 16 KB FLASH programovou paměť a 1 KB paměť RAM. Při návrhu zapojení obvodů s mikroprocesorem bylo dodrženo doporučené zapojení od výrobce. Pro detailnější popis čipu je k přečtení datasheet [13]. Na desce s mikroprocesorem je potřeba připojit několik součástek, aby mikroprocesor fungoval správně. Pro časování mikroprocesoru je na desce krystal s frekvencí 8 MHz. Na pinech mikroprocesoru jsou připojená tlačítka, jedno pro přenastavení mikroprocesoru a další pro volbu režimu svícení LED nebo jiné funkce dle naprogramování. Pro umožnění sériové komunikace s počítačem je na desce rozhraní USB na UART a USB mini konektor. Na připojení přes konektory na desku s LED a na desku s fototranzistory musely být na desku s mikroprocesorem umístěny konektory. Při umísťování součástek na desku musí být dodržena vzdálenost součástek od sebe navzájem a vzdálenost od vodivých cest. Je důležité si dávat pozor na to, aby se nezpůsobil zkrat či jiná elektrická porucha na desce. Časově nejnáročnějším procesem při výrobě desek plošných spojů bylo ruční osazování součástek na desku a následné pájení. LED a senzory jsou řízeny mikroprocesorem. Mikroprocesor řídí, které LED se zapnou, naměření hodnot na fototranzistorech a převod v AD převodníku na digitální data. Také zprostředkuje sériovou komunikaci s počítačem. V počítači se uloží informace o tom, který fototranzistor se otevře, nebo ne a podle toho je potom algoritmem určeno, kde je robot. Při návrhu desek se pro usnadnění

výroby desky prohodily piny některých LED diod a fototranzistorů. To ničemu nevadilo, protože se to jednoduše ošetřilo programově.

Na PCB s LED jsou rezistory  $180\ \Omega$ , protože na LED je při otevření úbytek napětí  $1,4\ \text{V}$  a proud  $20\ \text{mA}$ . V zapojení je sice i bipolární tranzistor, ale na něm bude malý úbytek napětí, když poteče proud jen  $20\ \text{mA}$ , tak podle výpočtu  $5\ \text{V} - 1,4\ \text{V} = 3,6\ \text{V}$  a  $3,6\ \text{V}/20\ \text{mA}$ . Na PCB se senzory předpokládáme, že maximální hodnota proudu může být  $1\ \text{mA}$ , proto byl vybrán rezistor s hodnotou  $5\ \text{k}\Omega$ .

Při návrhu a výrobě DPS byla nejdříve vytvořena varianta rozmístění součástek na desce tak, jak to i bylo navrženo v programu Altium. Poté se začalo pracovat na zlepšení rozložení součástek. Počítala se unikátnost matic ze simulace, jak bylo již zmíněno. Matice LED a senzorů byly pro každou pozici  $x,y$  robota. Některá rozmístění LED a senzorů by pro více různých pozic vrhala stejný stín. V důsledku by byla nalezena pozice, která by ale nebyla jedinou možnou pozicí z  $X$  pozic. Proto se testovala unikátnost matic. Nejdříve bylo generování pozic podle napsané funkce v Matlabu náhodné, ale pro tento postup vycházel počet neunikátních pozic celkem vysoký, až stovky tisíc. Proto bylo použito genetické programování pro generování pozic. V Matlabu genetický algoritmus má následující kritéria zvané **optimoptions**. Zvolené hodnoty pro vyhledání rozmístění součástek a výpočtu unikátních matic byly použity následující:

- maximální počet generací **MaxGenerations**, nastavená na 10,
- velikost populace **populationsize**, nastavená na 60,
- použití paralelního výpočtu **useparallel**, nastavené true (pravda),
- počáteční populace **InitialPopulationMatrix**, vstupní argument byl vektor o velikosti 24, protože bylo 16 pozic LED a 8 pozic fototranzistorů.

Při aplikaci genetického algoritmu výsledné hodnoty sumy matic stále vycházely příliš velké. Vysvětlení genetického algoritmu je nad rámec této práce, proto pro další poznatky je vhodný např. [12].

### 3.2.2 Software mikroprocesoru

Pro naprogramování mikroprocesoru v programovacím jazyce C bylo použito vývojové prostředí Codewarrior, což je IDE od NXP. Nejdříve byly deklarovány a nastavené výchozí hodnoty proměnných. Proměnné, které jsou volány v přerušeních, je důležité deklarovat jako **volatile** (nestálý). Primárně bylo nutné, aby řídicí jednotka odesílala do nadřazeného systému informace o stavech na fototranzistorech. Bylo nutné zprostředkovat funkční sériovou komunikaci mezi mikroprocesorem a uživatelským počítačem, změření analogových hodnot senzorů a jejich převod na digitální hodnoty. Ukládání dat z ADC probíhá v přerušení mikroprocesoru a byla uložena do pole o velikosti osmi hodnot (šestnáct bajtů) z

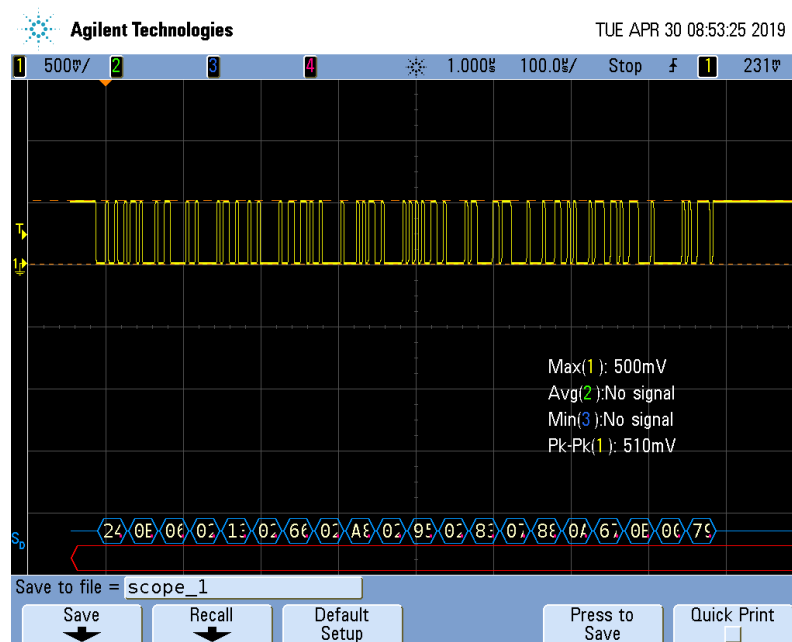
osmi fototranzistorů. Data z ADC přicházela po dvanácti bitech, tzn. že nejvyšší dvanáctibitová hodnota může být  $2^{12}$ , což je dekadicky 4096. Naměřené hodnoty byly přiřazeny do dalšího pole typu **uint16** (bezeznamková celočíselná šestnáctibitová hodnota) a byly připraveny k odesílání přes UART do počítače. Proměnné, do kterých se ukládala data z registrů, byly pole typu **uint8**. Takže se přes UART posílaly naměřené hodnoty po bajtech.

Data byla odeslána v paketech, které měly strukturu dle tab. 3.1. Pro označení začátku příchozích dat byl nastaven start bajt, jako ASCII znak dolaru, což je hexadecimálně 24. Poslední bajt v paketu (end bajt) byl nastaven jako kontrolní součet. Přenosová rychlost je 250000 bitů za sekundu. Paket posílaný do počítače je o velikosti dvaceti bajtů. Druhý bajt po start bajtu je číslo LED, která svítí. Na obr. 3.6 je vidět bajt po bajtu a jejich hexadecimální hodnoty. V paketu je také připravený bajt na informaci o režimu měření. Režim je buď automatický nebo manuální. V automatickém režimu svítí LED po LED a při manuálním se nastavuje která LED svítí.

LED jsou nastavené, aby svítily každá po dobu deseti milisekund. To lze udělat podle vnitřních hodin jádra, kdy v přerušení vyvolaném přetečením časovače každou milisekundu bude inkrementovaná proměnná až napočítá do deseti. Poté bude zavolána funkce pro svícení LED. Pro přepínání režimu svícení LED bylo naprogramované jedno z tlačítek.

start	LED	val	val	val	val	val	val	val	val	režim	check sum
1B	1B	2B	2B	2B	2B	2B	2B	2B	2B	1B	1B

**Tab. 3.1:** Jednotlivé bajty dat, posílané přes sériovou komunikace (kde val je zkráceně slovo value (česky hodnota)).



**Obr. 3.6:** Zobrazení průběhu na osciloskopu s paketem bajtů posílaných do počítače

### 3.2.3 Simulace a zpracování dat

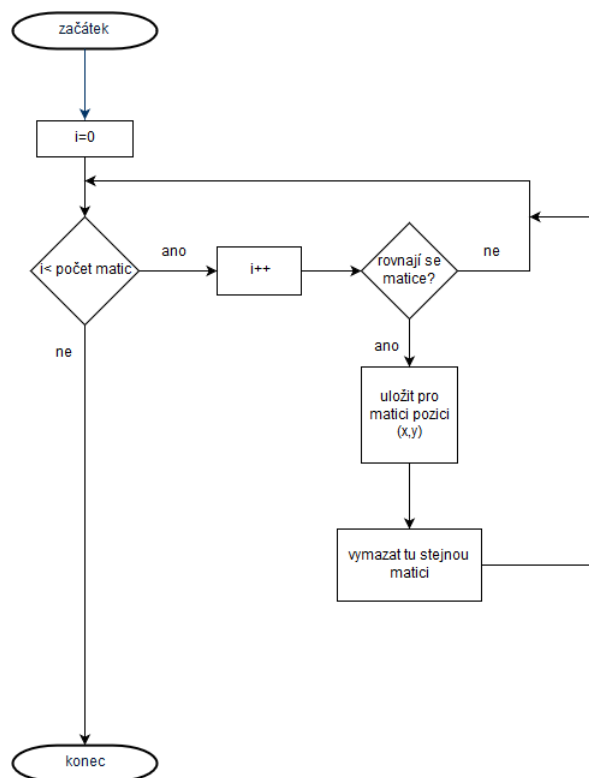
Pro práci se zařízením je třeba komunikovat přes sériový port. V Matlabu je pro práci se seriovým portem několik knihovních funkcí. Pro zprovoznění komunikace je nutné definovat na jakém COM portu je zařízení připojeno a nastavit přenosovou rychlost. Pro otevření komunikace je funkce **fopen()** a pro přečtení přijatých dat funkce **fread()**.

Zařízení pro lokalizaci bylo postaveno podle návrhu DPS, kde šestnáct LED je umístěných rovnoměrně podél hrany o 100 mm a naproti nim je stejně umístěných osm fototranzistorů. Pro zjištění, které zapojení je vhodné, bylo simulováno svícení LED určených podle zapsání vybrané pozice do vstupní matice LED, jak již bylo zmíněno v kap. 3.2. Bylo graficky znázorněno umístění LED, tranzistorů, pozice tělesa a stín, který vrhá. Pro každou samostatně rozsvícenou LED bylo vidět, které fototranzistory budou zastíněné. Ze simulace byly vytvořila 3D matice plná desetitisíc matic. Pro každý vektor pozice robota  $x$  a  $y$ , je vždy jedna matice LED a jedna fototranzistorů. Matice pro LED obsahuje pozice  $x$  a  $y$  každé ze šestnácti LED, takže má šestnáct řádků a dva sloupce. Matice fototranzistorů obsahuje, které jsou zastíněné a které ne, a to pro každou LED, takže má rozměry šestnáct sloupců a osm řádků.

Na vyrobeném zařízení LED se rozsvítí jedna po druhé. Pro každý fototranzistor se při rozsvícení jedné LED naměří hodnoty a pošlou přes ADC a poté po sériové komunikaci do počítače, kde se mohou dále zpracovávat tato data. Nejdříve se nasimuluje rozsvícení LED a zastínění fototranzistorů, z této simulace se ukládají údaje do matic. Do matice pro LED se uloží, které LED svítily a do matice pro senzory se uloží jedničky pro fototranzistory, které byly nasvícené a nuly pro ty zastíněné. Potom se v Matlabu přes sériovou komunikaci ve vytvořené funkci **serial read** zpracovávají data a uloží se hodnoty naměřené na fototranzistorech do matice. Tyto hodnoty se porovnají se stanoveným limitem, který je hranicí mezi nasvícenými fototranzistory a těmi zastíněnými. To proto, že se na otevřeném fototranzistor naměří hodnoty vyšší než 500 a u zastíněného fototranzistoru jsou podstatně menší než 500. Do matice **positive** se uložily na pozice, kde hodnoty byly pod limitem jedničky a kde byly nad nuly. Porovnáním matic LED a matic fototranzistorů ze simulace s reálně naměřenými hodnotami, lze určit pravděpodobnou pozici malého tělesa. Narazilo se na problém s lokalizací při různých světelných podmínkách. Hodnota limitu, nemůže být stejná při všech světelných podmínkách, protože se naměřené hodnoty výrazně liší. Při denním světle je hodnota 200 limitu příliš vysoká, takže neuloží do matice, že jsou fototranzistory zavřené. Vytvořila se proto funkce, kde se výpočet optimální hodnoty limitu odvíjí od její závislosti na průměrné hodnotě naměřených hodnot. Průběh funkce je znázorněn graficky v grafu. Díky tomu, že hodnota limitu bude změněna podle situace osvětlení v místnosti, kde proběhne lokalizace se zařízením, by vyhledání mohlo být o něco přesnější.

Celý proces výpočtů a zpracování dat je celkem zdlouhavý, proto je třeba vynechat zbytečné výpočty. Zbytečným výpočtem je myšleno porovnání matic fototranzistorů ze simulace s maticemi fototranzistorů naměřených hodnot tam, kde není matice unikátní. To

znamená, že pro kombinaci rozsvícené LED a pozice robota, se může uložit stejná matice fototranzistorů vícekrát. Doba se může zkrátit, tím že se vynechají takové matice, ale uloží se pro tyto matice pozice tělesa. Algoritmus pro zredukování těchto stejných matic fototranzistorů je vidět na obr. 3.7. Naměřené časy trvání každé verze výpočtu pozice tělesa se zprůměrovaly. Výsledné průměrné časy jsou pro nezrychlenou verzi 0,0315 vteřin a pro zrychlenou verzi 0,0108 vteřin. Díky zredukování počtu matic fototranzistorů se výpočet pozice zrychlil třikrát.

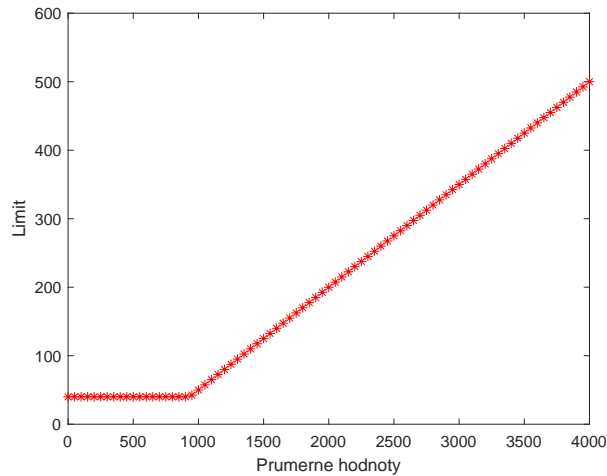


**Obr. 3.7:** Vývojový diagram algoritmu pro redukci stejných matic

Fototranzistory byly označeny jako otevřené na základě hodnot. Hodnoty, které byly vyšší než limitní hodnota byly nastaveny v matici **positive** jako jedničky zbytek hodnot byly nastaveny jako nuly. Limit byl nejdříve určen na základě naměřených hodnot na čtyřicet, ale později bylo zjištěno, že tato hodnota není fixní, ale závislá na okolním světle. Pro určení hodnoty limitu z naměřených hodnot byla vytvořena funkce **recalculateLimit**. Vycházelo se z faktu, že při nejnižším okolním světle byl průměr naměřených hodnot kolem 1000 a vhodný limit roven padesáti a jelikož při větším okolním světle nejvyšší průměr naměřených hodnot byl kolem 2000 a vhodný limit roven 200, bylo možné určit rovnici přímky závislosti limitu na průměrné naměřené hodnotě dle rov. 3.1. Konstanty  $k$  a  $q$  byly jednoduše vypočteny ze soustavy rovnic s dosazenými známými hodnotami limitů a průměrů. Funkce **recalculateLimit** byla zavolána při každém měření, aby hodnota limitu odpovídala daným okolnostem. V grafu v obr. 3.8 je vidět že při nižším průměru hodnot jak 1000 je limit konstantní hodnota.

$$y = kx + q \quad (3.1)$$

Kde: y-limit; x-průměr; k,q-konstanty



**Obr. 3.8:** Graf závislosti limitu na průměrné naměřené hodnotě

Jenže i při užití limitu daného okolními světelnými podmínky se nezdařilo eliminovat potíží s naměřenými hodnotami.

Potíž s vlivem okolního denního světla či jiného světla, nás donutil vylepšit měření se zařízením. Pro vytěsnění hodnot, naměřených jen působením okolního světla, se tyto hodnoty odečtou od hodnot naměřených při zapnutí emitorů senzoru. Mikroprocesor tak nejdříve naměří hodnoty na fototranzistorech zatímco jsou ještě vypnuté LED diody. Po naměření se může postupovat dál tak jako předtím. Hodnoty okolního světla byly uloženy do proměnné **Offset** a odečteny v Matlabu od hodnot naměřených při lokalizaci. Měřením bylo zjištěno, že naměření hodnot Offsetu nám při lokalizaci nepomohlo. Předpokládalo se, že při odečtení hodnot okolního světla od hodnot světla při zapnutých LED by hodnoty odpovídaly světla z LED, ale po odečtení hodnoty tomu neodpovídají.

Jelikož, ale na desce může být jen polovina fototranzistorů připojena přes headery, znamená to že při zakrytí LED se neprojeví že tomu tak je, podle naměřených hodnot na fototranzistorech. Proto se připojily ob fototranzistor. Pro připojení každého druhého fototranzistoru se musela vytvořit nová předloha pro porovnání se simulací, taky první pozice 4 mm od kraje a ob 12 mm až do poslední pozice 88 mm od konce.

### 3.3 Porovnání naměřených výsledků dvojice aplikovaných metod

Cílem práce bylo zjistit, který způsob lokalizace je nejrychlejší a nejpřesnější. U metody pomocí kamery a algoritmů je rychlost dána hlavně počtem a náročností operací. Jestliže je výpočet a zpracování obrázků náročný, například jako BMA Full search, kdy je mnoho

kroků, tak je pomalejší algoritmus. Při porovnávání algoritmů nejdříve v rámci metody pomocí kamery a algoritmů mezi sebou bylo zjištěno, že rychlejší lokalizace je u videa s nižším rozlišením. To proto, že je méně pixelů ve snímku, a tak se méně krát opakuje porovnávání. Nejrychlejšími BMA algoritmy byly BMA urychlený s použitím SURF a BMA v okolí maxima, jak je zřejmé z první tabulky. Zjištění poukazuje na jasnou volbu pro lokalizaci a to na integrální obraz s aplikací detektoru rohu. V následujících tabulkách jsou algoritmy seřazeny od nejrychlejšího k nejpomalejšímu.

Algoritmus	čas [s]
Integrální obráz	0,04
BMA s použitím SURF	0,05
BMA v okolí max rozdílu	0,06
Full BMA	1,15

**Tab. 3.2:** Průměrné naměřené časy trvání jednotlivých algoritmů v oříznutém snímku 412x438

Algoritmus	čas [s]
Integrální obráz	0,18
BMA v okolí max rozdílu	1,67
BMA s použitím SURF	2,48
Full BMA	16,54

**Tab. 3.3:** Průměrné naměřené časy trvání jednotlivých algoritmů v oříznutém snímku 1000x980

Souřadnice pozice středu robota  $x$  a  $y$ , pro všechny algoritmy vyšla stejně, až na vyhledávání s integrálním obrazem. Pozice při stejném výpočetním postupu středu robota nevycházela na střed, ale na horní levý roh, proto se odečetla velikost šablony vydělená čtyřmi. Čtyřmi protože při vydělení dvěma stále nebyla pozice ve středu robota. Příčinou byla velikost šablony, mohla totiž být nepřesně oříznutá a kolem robota mohlo být pozadí tj. aréna. Potom by poloviční velikost šablony ve skutečnosti neodpovídala středu robota. Při vydělení čtyřmi již vycházela pozice blíže ke středu. V následujících tabulkách jsou porovnané pozice pro různé algoritmy pro jeden snímek.

Algoritmus	pozice x	pozice y
Integrální obraz	297	115,9
Full BMA	295	120,5
BMA v okolí max rozdílu	295	120,5
BMA s použitím SURF	295	120,5

**Tab. 3.4:** Porovnání výsledných pozic při aplikaci různých algoritmů pro snímky o velikosti 412x438

Algoritmus	pozice x	pozice y
Integrální obraz	448,3	224,3
Full BMA	439	219,5
BMA v okolí max rozdílu	439	219,5
BMA s použitím SURF	439	219,5

**Tab. 3.5:** Porovnání výsledných pozic při aplikaci různých algoritmů pro snímky o velikosti 1000x980

Pro lokalizaci s integrálním obrazem je nutné zmenšit oblast šablony tak, aby bylo minimální okolí kolem robota a nebyla v šabloně i aréna, po které jezdí robot. Při měření s menší šablonou integrálním obrazem již byla určená pozice středu blíž k výsledkům ostatních algoritmů. Například pozice z integrálního obrazu 294,28 113,927 a pozice ze SURF 291,5 116,5, už je malý rozdíl. Integrálnímu obrazu to trvalo 0,04 vteřin a SURF s BMA 0,1 vteřin. Pro přesnější porovnání pozic integrálního obrazu a ostatních algoritmů byla použita zmenšená šablona pro všechny. U každého algoritmu se ukládaly pozice do matice, po měření se odečetla matice pozic pro integrální obraz od matice pozic pro algoritmus BMA, potom BMA kolem maxima a jako poslední BMA urychlené se SURF. Z rozdílu se vzaly průměrné rozdíly pro souřadnici x a potom y. Pro celkovou chybu pozice se z x a y vypočítala přepona pomocí Pythagorovy věty. V Matlabu lze použít funkci **hypot** a jako x a y parametry se mohou zadat první sloupec v matici pozic a y druhý sloupec v matici.

Porovnáním výsledných pozic při použití integrálního obrazu s pozicí při použití BMA algoritmu zrychleného pomocí SURF je zřejmé, že nejpřesnější pozice vyšla při použití šablony o velikosti 21x21 pro rozlišení 412x438 a 43x43 pro rozlišení 1000x980.

Vel. šablony	celk. průměrný rozdíl pixelů	průměr rozdílů x	průměr rozdílů y
18x18	5,4484	4,0600	2,4405
19x19	4,4477	3,2869	2,0880
20x20	3,7166	2,6946	2,1827
21x21	3,3998	2,1681	2,4033
22x22	4,7852	1,9097	3,5075

**Tab. 3.6:** Porovnání výsledných pozic algoritmů integrální obraz a BMA urychlené se SURF při užitím různě velkých šablon. V rozlišení 412x438

Při lokalizaci tělesa pomocí senzoru se urychlilo hledání tím, že se zredukoval počet matic pro porovnání, jelikož se několik matic fototranzistorů rovnaly, i když byly pro různé pozice robota.

Před stavěním zařízení byla udělána simulace náhodného rozmístění LED a senzorů a vytvořených stínů z robota ve všech možných pozicích. Hledala se unikátní kombinace



Vel. šablony	celk. průměrný rozdíl pixelů	průměr rozdílů x	průměr rozdílů y
46x46	3,7034	2,8807	1,8097
45x45	3,1720	2,4066	1,5728
44x44	2,1772	1,6142	0,9594
43x43	1,9889	1,328	1,1316
42x42	2,6841	1,41	1,9499

**Tab. 3.7:** Porovnání výsledných pozic algoritmů integrální obraz a BMA urychlené se SURF při užitím různě velkých šablon. V rozlišení 1000x980

Porovnané algoritmy	rozlišení videa	celkový průměrný rozdíl pixelů
Integrální obraz vs BMA	1000x980	3,1720
Integrální obraz vs BMA	412x438	4,4492

**Tab. 3.8:** Porovnání pozic vypočítaných algoritmy

Metoda vyhledání pozice tělesa	čas [s]
porovnání všech matic fototranzistorů	0,0315
porovnání zredukovaného počtu matic	0,0108

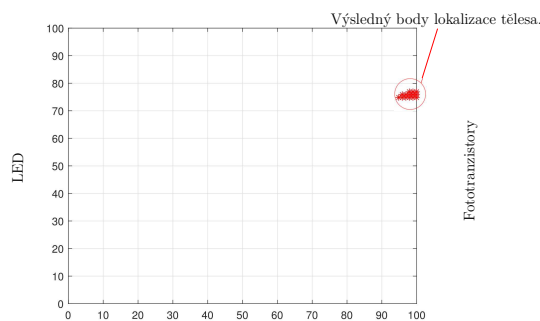
**Tab. 3.9:** Porovnání rychlostí metody rozsáhlé (porovnání všech matic) se zkrácenou metodou (zredukovaný počet matic)

rozmístění součástek, ale nebyla nalezena optimální. Součástky byly umístěny na desky a pro jejich fyzické umístění opět vypočítaná unikátnost. Unikátnosti náhodného simulovaného rozmístění a reálného fyzického rozmístění součástek byly porovnány v tab. 3.10. Sensory lze zapojit dvěma způsoby, buďto vedle sebe na půlce desky, nebo ob sensor přes celou délku desky. Unikátnost u reálného rozmístění je zastoupena jen maticí sensorů, protože LED mají jen jednu variaci a to, že svítí jedna po druhé.

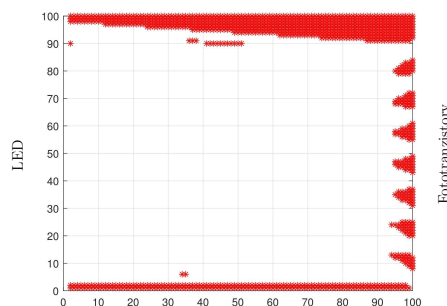
Rozmístění součástek	suma
náhodné rozmístění	48466
reálné rozmístění sensorů vedle sebe	8534908
reálné rozmístění sensorů ob jeden	1214622

**Tab. 3.10:** Porovnání počtu neunikátních matic pro různé simulace rozmístění součástek

Jediná situace, při které byla lokalizace úspěšná, bylo při přímém zakrytí fototranzistoru. Při zakrytí jakéhokoliv fototranzistoru byl vždy správně rozpoznán. V Matlabu bylo do skriptu komunikace se sériovým portem uděláno grafické vykreslování bodů, které vyšly jako možná pozice tělesa.



**Obr. 3.9:** Grafické znázornění lokalizování zakrytého sedmého senzoru



**Obr. 3.10:** Grafické znázornění lokalizování při zapojení sensorů ob jeden

Metodu lokalizace pomocí senzoru je ve výsledku těžké aplikovat na lokalizaci malých předmětů. Možným řešením by bylo použití menších LED a ve větším množství, nebo

použít součástky s menším vyzařovacím úhlem. Obecně mají THT součástky menší vyzařovací úhel oproti SMD součástek, protože se používají v dálkových ovladačích. Jiné řešení by mohlo být použití laserových diod namísto infračervených. Laserové jsou například používány v průmyslových dotykových monitorech, ale je to spíše pro velké předměty, které je jednodušší lokalizovat. Optimální metoda pro lokalizaci malých těles je jednoznačně pomocí kamery a algoritmů. Z naměřených průměrných dob trvání vyhledávání tělesa můžeme říct, že nejrychlejší metodou je pomocí kamery a algoritmu integrálního obrazu s aplikací detektoru rohů. Lokalizace pomocí zařízení oproti lokalizaci pomocí kamery a algoritmů nebyla vůbec přesná a ani příliš rychlá. Když ale porovnáme lokalizaci při aplikaci různých algoritmů dopadl nejhůř algoritmus úplného vyhledávání BMA. Z metod s použitím BMA algoritmu vyšla nejlépe metoda BMA vyhledávání kolem maxima rozdílu.

# 4

## Závěr

Analýza metod pro lokalizaci malých těles, jejich výpočetní náročnosti a možnosti praktické realizace byla provedena v kapitole 2 s využitím kamery a algoritmů či senzorů. Dále byl proveden návrh vlastního zařízení pro lokalizaci malých těles.

Z výsledků provedených měření a analýz vyplývá, že spolehlivější v lokalizaci je metoda s kamerou a algoritmy. Pomocí této metody bylo úspěšně lokalizované těleso. S užitím integrálního algoritmu bylo nejrychleji lokalizováno. Při použití ostatních algoritmů byla lokalizace také úspěšná, jen bylo těleso nalezeno po delší době. Z provedených testů bylo zjištěno, že je lokalizace rychlejší ve snímcích s nižším rozlišením. Dále z provedených měření vyplývá, že lokalizace s kamerou a algoritmy by byla optimální kombinací integrálního obrazu s BMA.

V bakalářské práci byla také demonstrována lokalizační metoda, která využívá vyrobené zařízení a jejíž cílem je lokalizovat malé těleso užitím senzoru provedeným jako jednocestná závora. Při návrhu zařízení byly brány v potaz okolní podmínky, které by měření mohly ovlivnit. Také v provedených experimentech byla snaha nastavit parametry vhodným způsobem, aby byla co nejvíce umožněna úspěšná lokalizace a zpracovat nejlépe naměřené hodnoty. Při experimentech jsme se setkali s potížemi eliminace působení okolního světla a vyzařovacím úhlem součástí. Z těchto důvodů nelze považovat vyrobené zařízení za připravené k úspěšné lokalizaci. Pro další vývoj metody lokalizace s jednocestnou závorou by byly optimální součástky s menším vyzařovacím úhlem a ve větší koncentraci, nebo místo infračervených LED použití laserů. Další možností pro zaručení úspěšné lokalizace je aplikace senzoru pro lokalizaci většího tělesa.

Výsledky řešení bakalářské práce lze stručně shrnout do následujících bodů:

- byla provedena analýza běžně používaných metod pro lokalizaci,
- byl vytvořen přehled porovnání verzí algoritmů a jejich rychlostí,
- byla provedena aplikace algoritmů v nahraném videu kamerou,
- byla prakticky zrealizována a ověřena metodika s využitím senzoru.

Při porovnání stanovených cílů bakalářské práce, které měly být dosaženy a jsou uvedeny v zadání, s výsledky bakalářské práce, je možné konstatovat, že se splnily a byly přínosem pro lepší volení metod optické lokalizace malých těles.

# Literatura

- [1] Shen, Chong and Zesen, Bai and Cao, Huiliang and Xu, Ke and Wang, Chenguang and Zhang, Huaiyu and Wang, Ding and Tang, Jun and Liu, Jun. *Optical Flow Sensor/INS/Magnetometer Integrated Navigation System for MAV in GPS-Denied Environment*. Journal of Sensors, 2016.
- [2] Manning, Colin. *Digital Video Compression*. 1996. [Cit. 28. 4. 2019]. Dostupné z: <http://www.newmediarepublic.com/dvideo/compression/adv22.html>
- [3] Grzegorz, Ciepka. *Point cloud, What is a point cloud*. 2016. [Cit. 28. 3. 2019]. Dostupné z: <https://www.3deling.com/whta-is-a-point-cloud/>
- [4] Abshire, James. *NASA's Space Lidar Measurements of Earth and Planetary Surfaces*. 2010. [Cit. 12. 4. 2019]. Dostupné z: <https://ntrs.nasa.gov/search.jsp?R=20100031189>
- [5] Bay, Herbert. *SURF: Speeded Up Robust Features*. , 2006. [cit. 18. 4. 2019]. Dostupné z: <http://www.vision.ee.ethz.ch/surf/eccv06.pdf>
- [6] Mistry, Dr and Banerjee, Asim. *Comparison of Feature Detection and Matching Approaches: SIFT and SURF*. GRD Journals- Global Research and Development Journal for Engineering, 2017.
- [7] Juřík, M., Kuthan, J., Šmídl, V., Mach, F. *Trade-off Between Resolution and Frame Rate for Visual Tracking of Mini-robots on Planar Surfaces*. MARSS, 2019.
- [8] Kuthan, Jiří. *Elektromagnetický systém pro polohování magnetických těles*. Plzeň: Univerzitní knihovna ZČU v Plzni, 2017. [cit. 21. 4. 2019]. Dostupné z: <http://www.dspace5.zcu.cz>
- [9] Scheirich, Ján. *Optoelektronické součástky*. Dostupné z: <http://www.bakal06.chytrak.cz/22—Optoelektronicke-soucastky.pdf>
- [10] Novotný, K., Martan, T., Šístek, J. *Systémy pro optické komunikace* Praha: Vydavatelství ČVUT, 2003.
- [11] 3D LiDAR: True 3D Sensing and Spinning 2D Alternatives. In: *Clearpath Robotics*[online]. 10.1.2017 [cit. 23.4.2019]. Dostupné z:

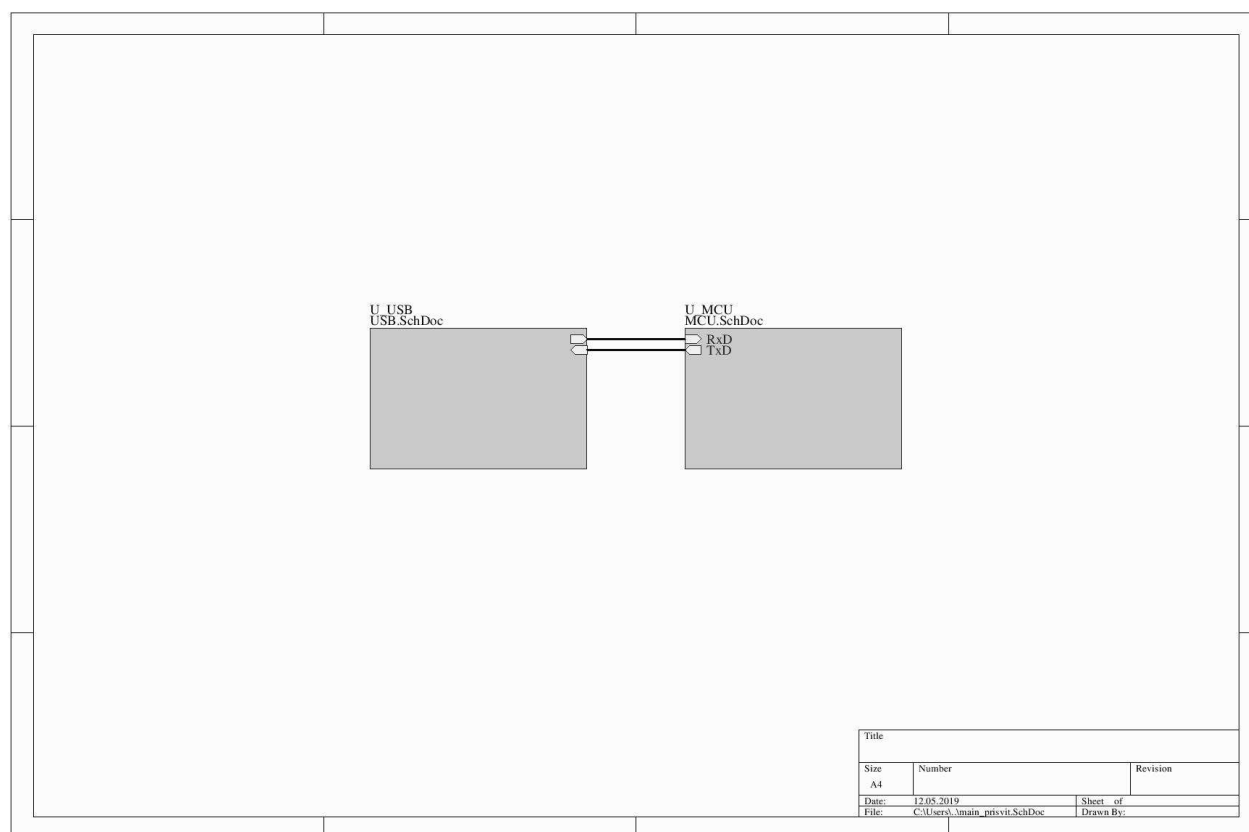
<https://www.clearpathrobotics.com/blog/2017/01/3d-lidar-true-3d-sensing-spinning-2d-alternatives/>

- [12] *GPLab: A Genetic Programming Toolbox for MATLAB*. [cit. 15.5.2019]. Dostupné z: <http://gplab.sourceforge.net/>
- [13] Freescale Semiconductor, Inc. [online katalogový list]. *MC9S08MP16RM*. 2011 [cit. 15.5.2019]. Dostupné z: <http://www.nxp.com/docs/en/data/sheet/MC9S08MP16RM.pdf>

# Příloha A

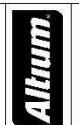
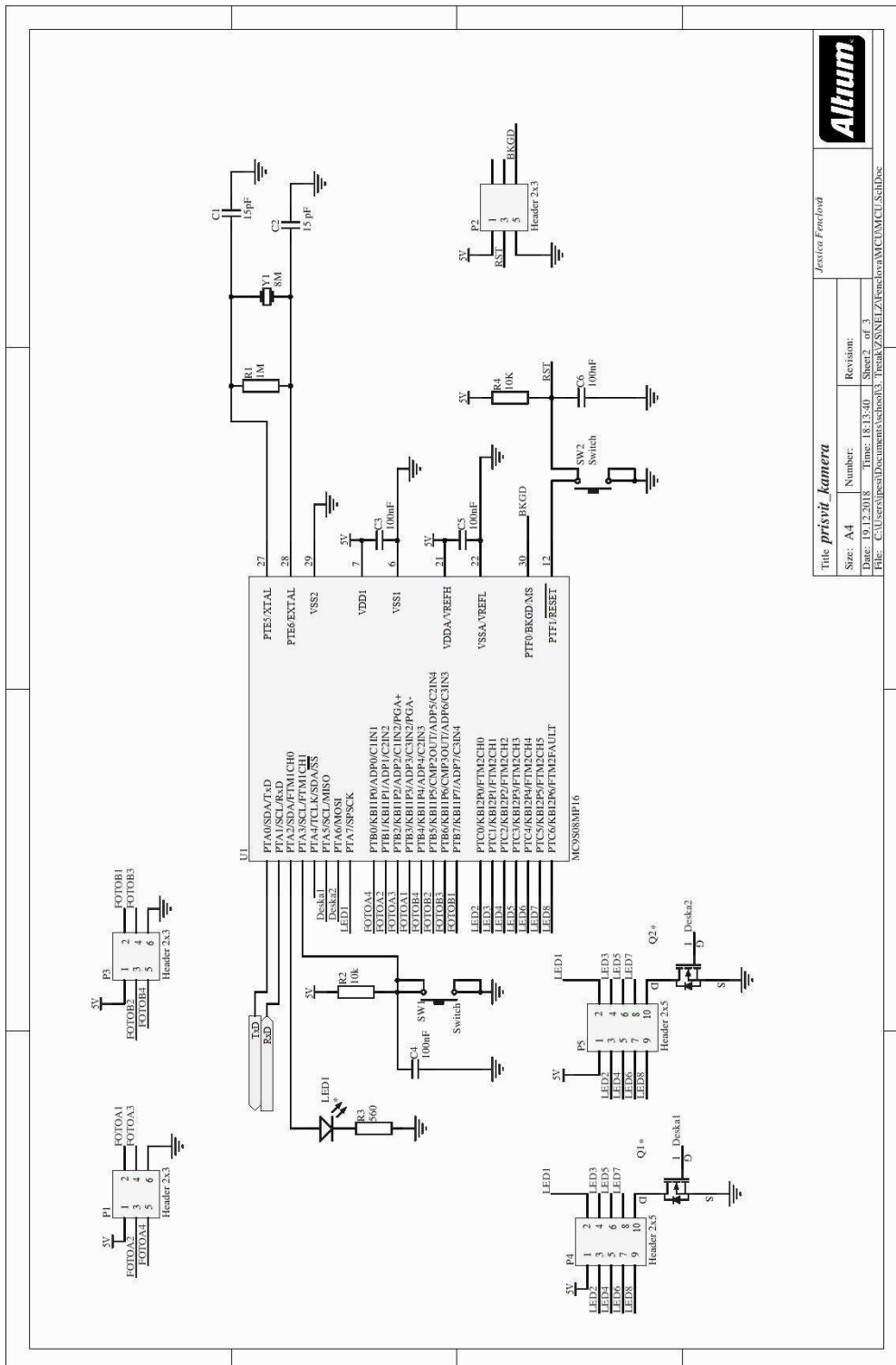
## Schémata zapojení

### A.1 Deska MCU



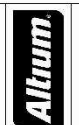
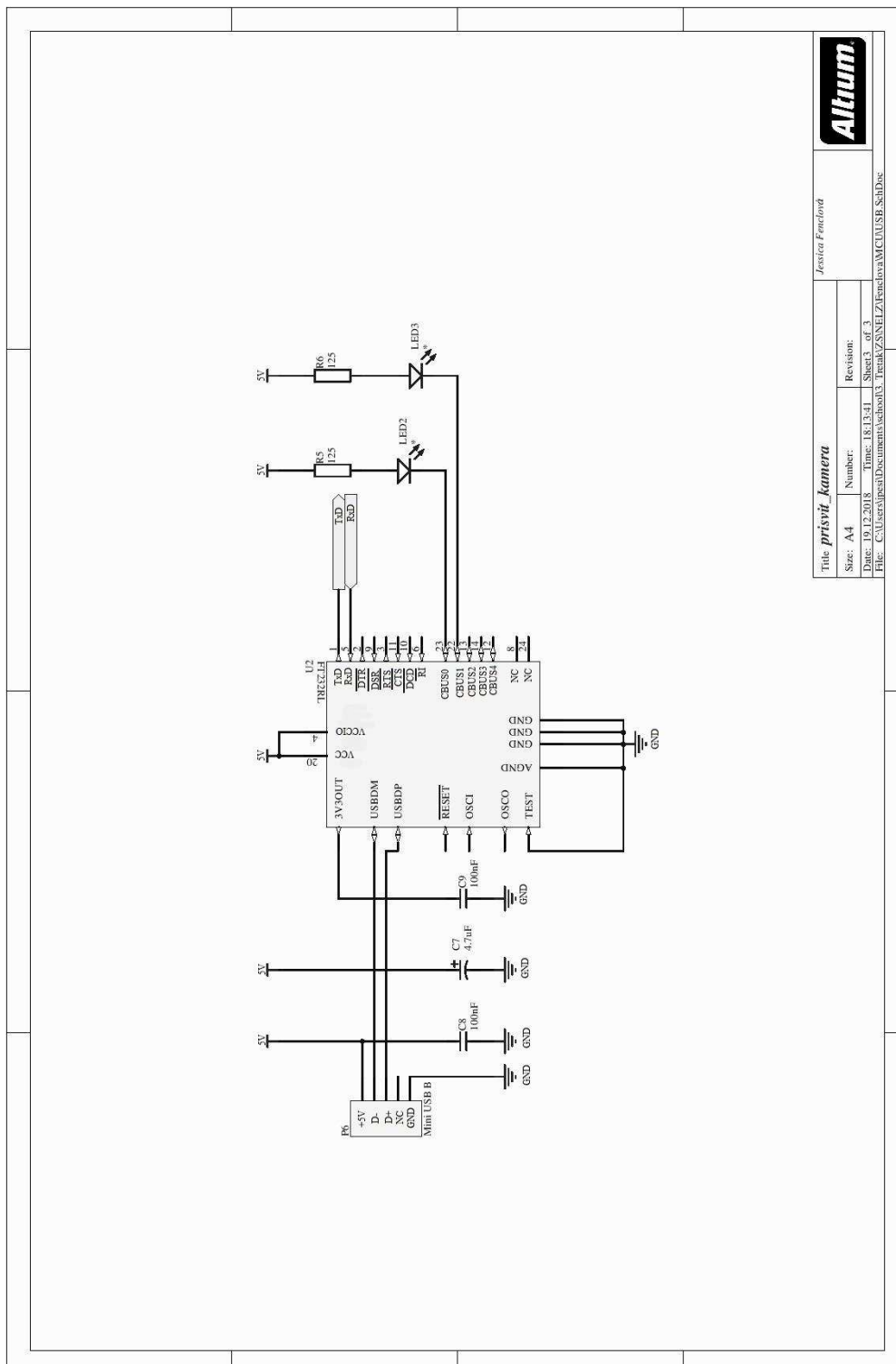
Obr. A.1: Propojení MCU s USB





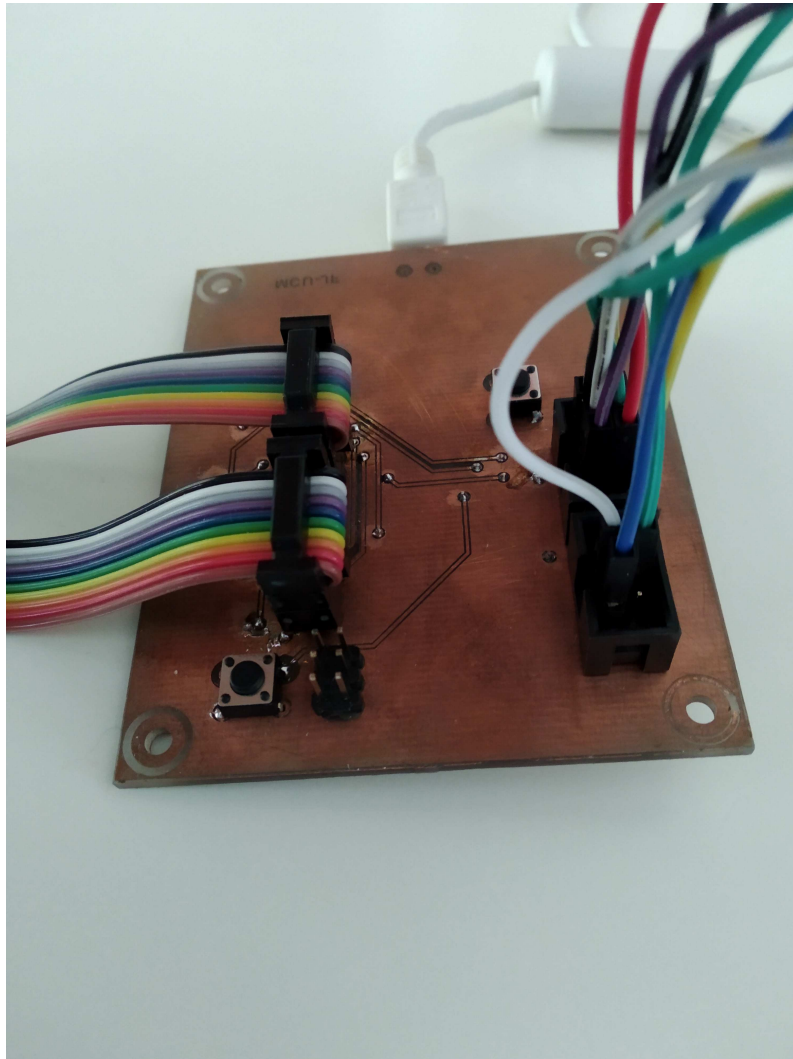
Title: <i>průsvit_kamera</i>		Revision:	
Size: A4	Number:	Date: 19.12.2018	Time: 18:13:40
File: C:\Users\jpeal\Documents\work\13_Trenk\ZS\NEI\Z\Fenclova\MCM\CU_SchDoc		Sheet 2 of 3	

Obr. A.2: Schéma desky s MCU

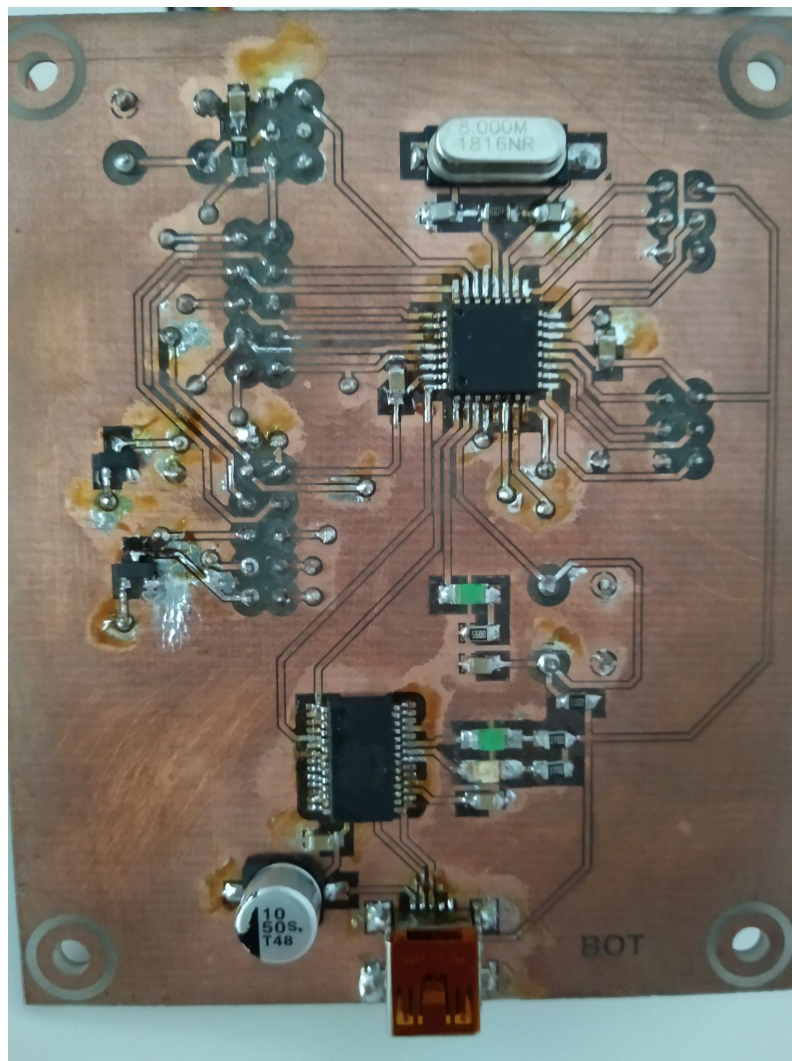


Title: *průsvit\_kamera*  
 Size: A4  
 Date: 19.12.2018  
 File: C:\Users\jpeal\Documents\schob\3\_Trena\ZSN\ELZ\Fenclova\MCU\USB\_SchDoc

Obr. A.3: Schéma desky s USB



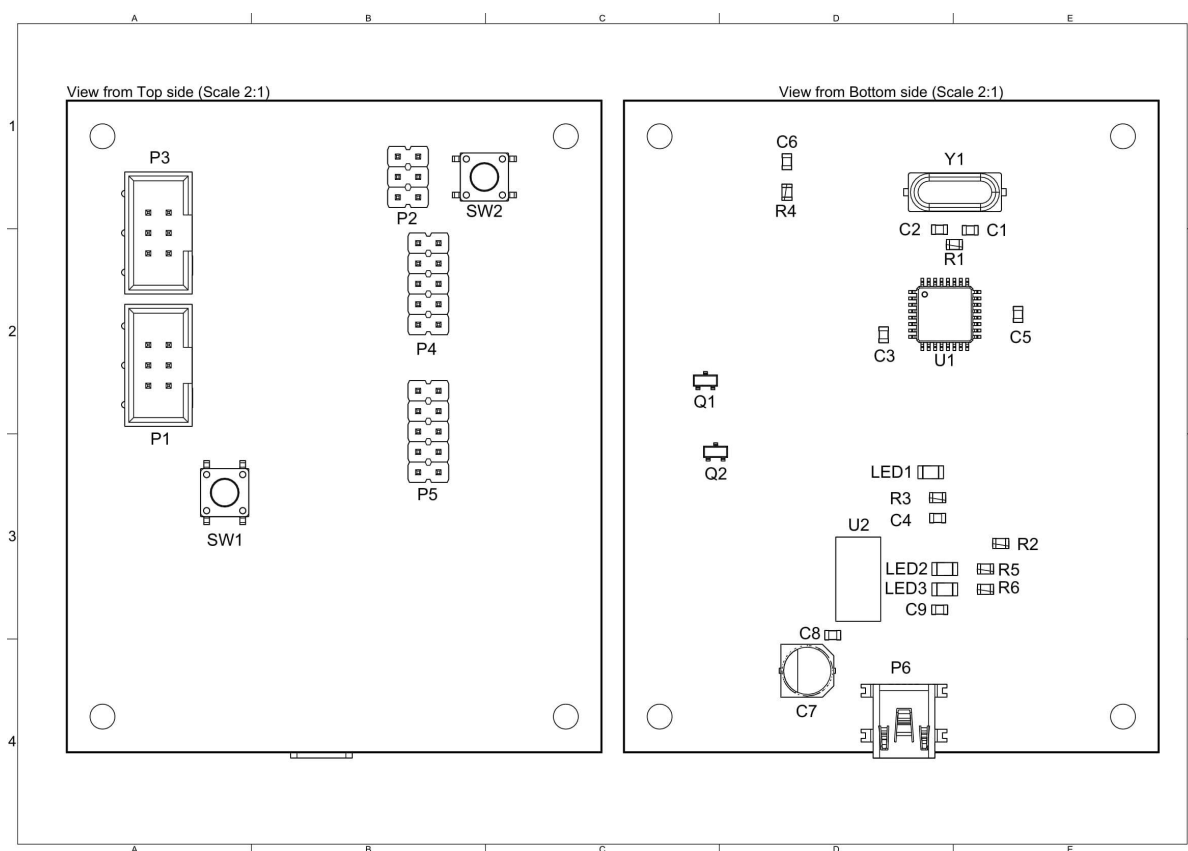
**Obr. A.4:** DPS MCU a USB-top



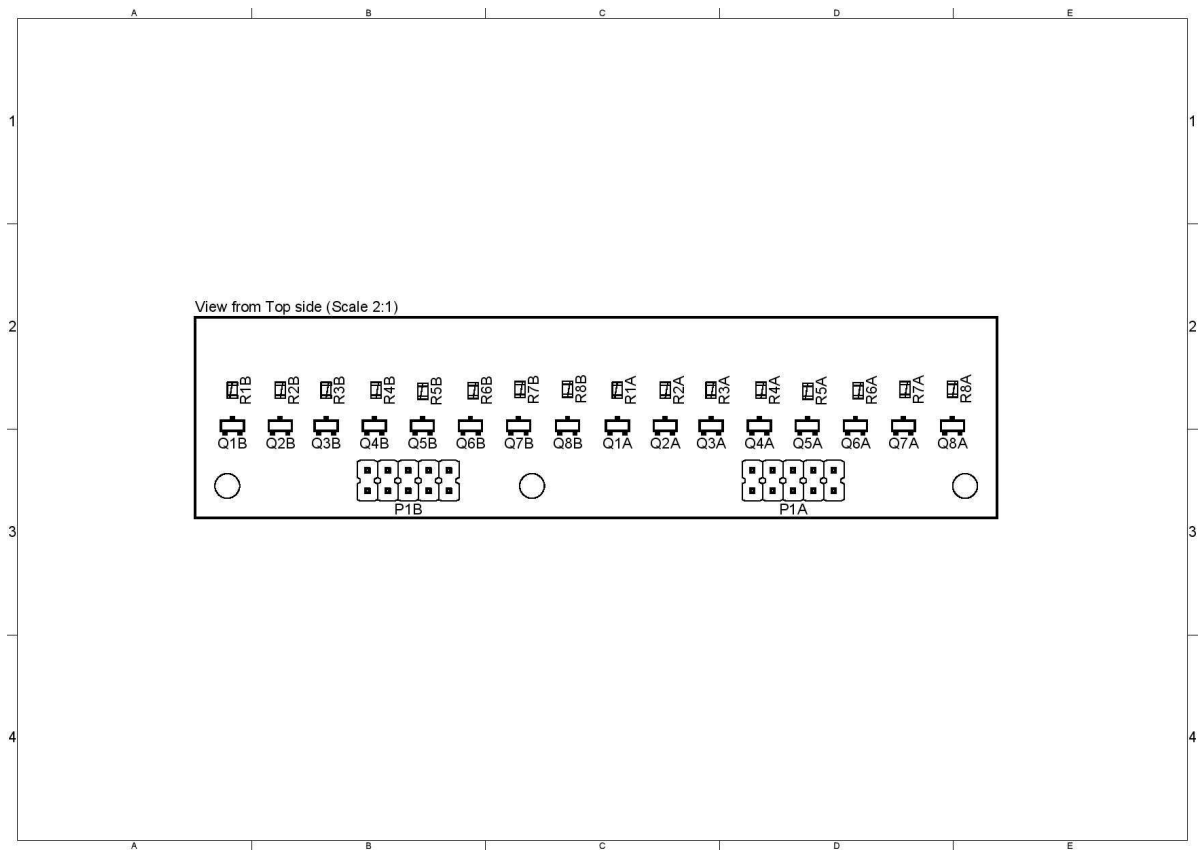
Obr. A.5: DPS MCU a USB-bot

# Příloha B

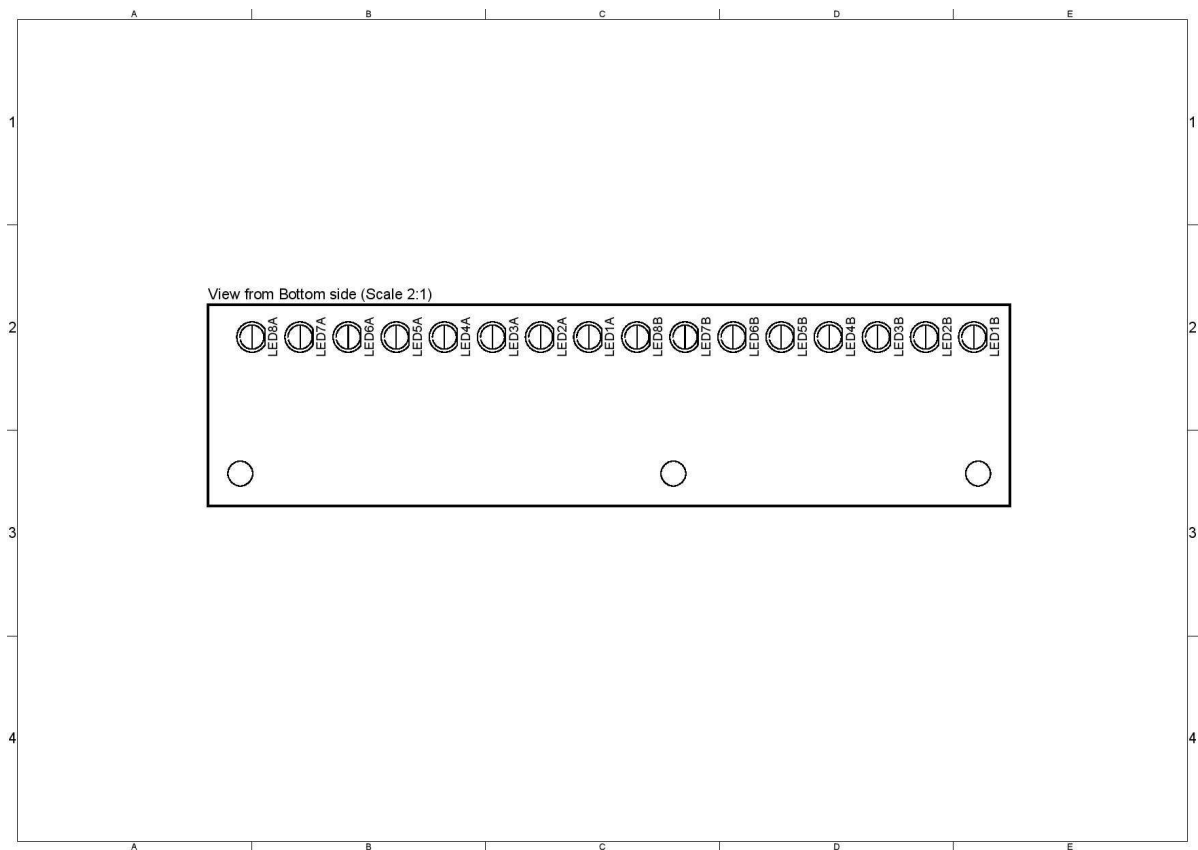
## Desky plošných spojů, výkresy



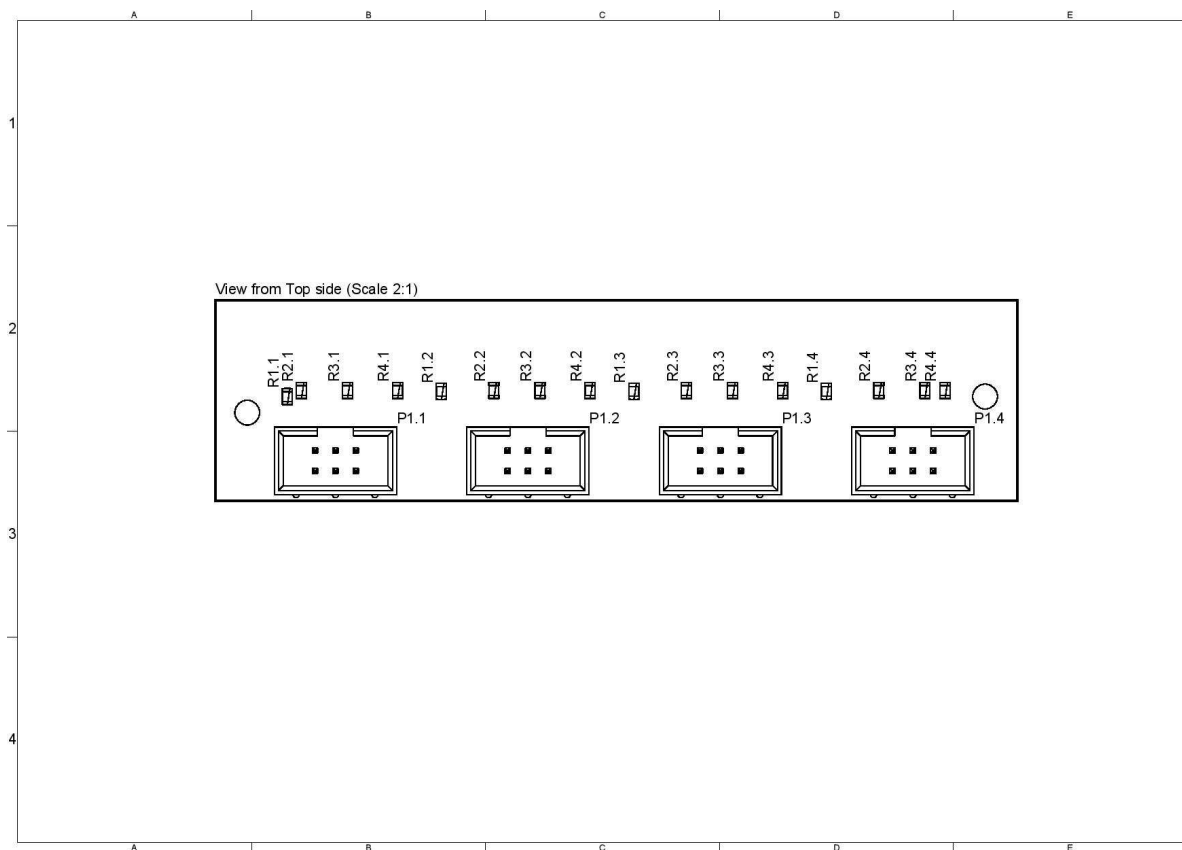
Obr. B.1: Osazení součástek na DPS s mikroprocesorem



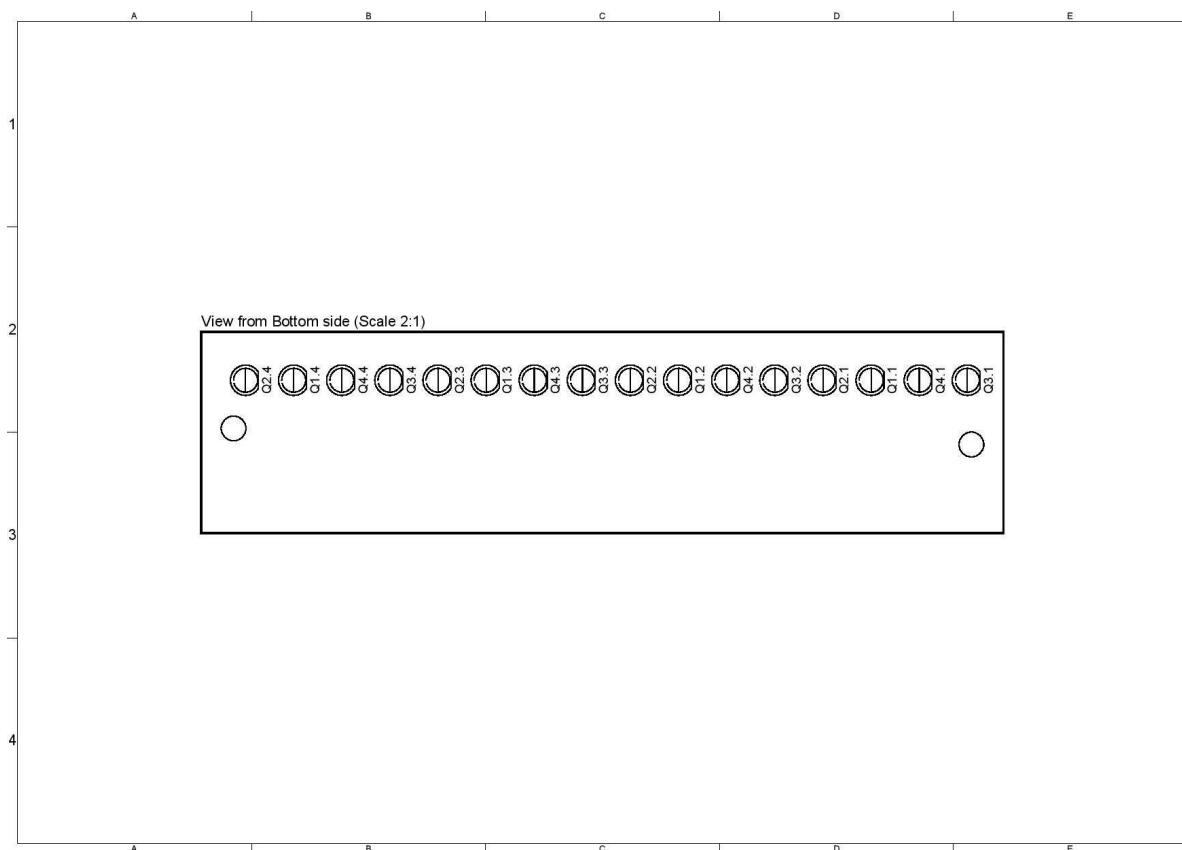
Obr. B.2: Osazení součástek na horní vrstvu DPS s LED



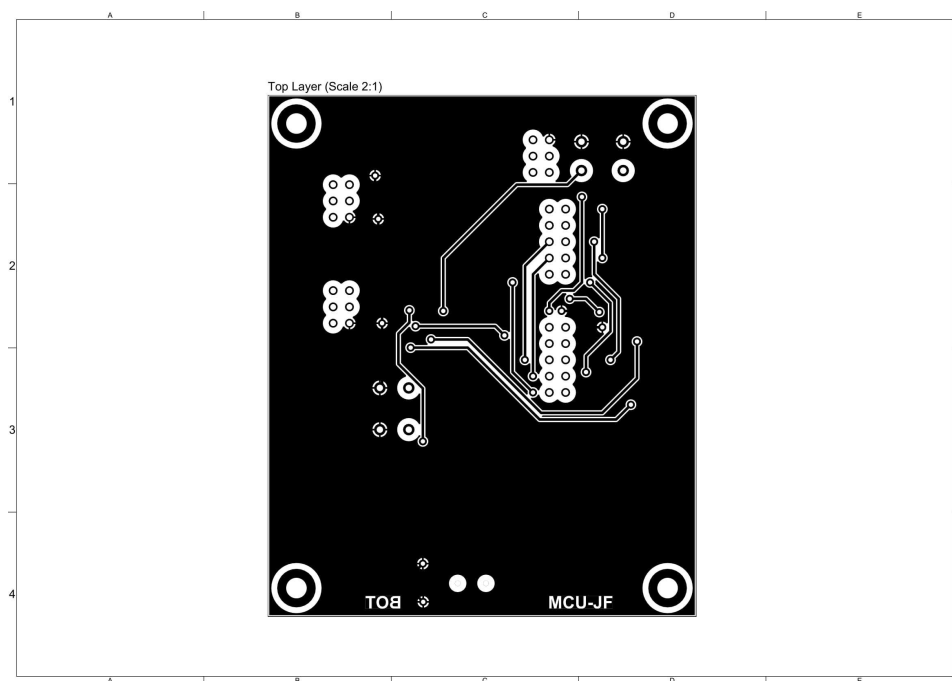
Obr. B.3: Osazení součástek na spodní vrstvu DPS s LED



Obr. B.4: Osazení součástek na horní vrstvu DPS s fototranzistory

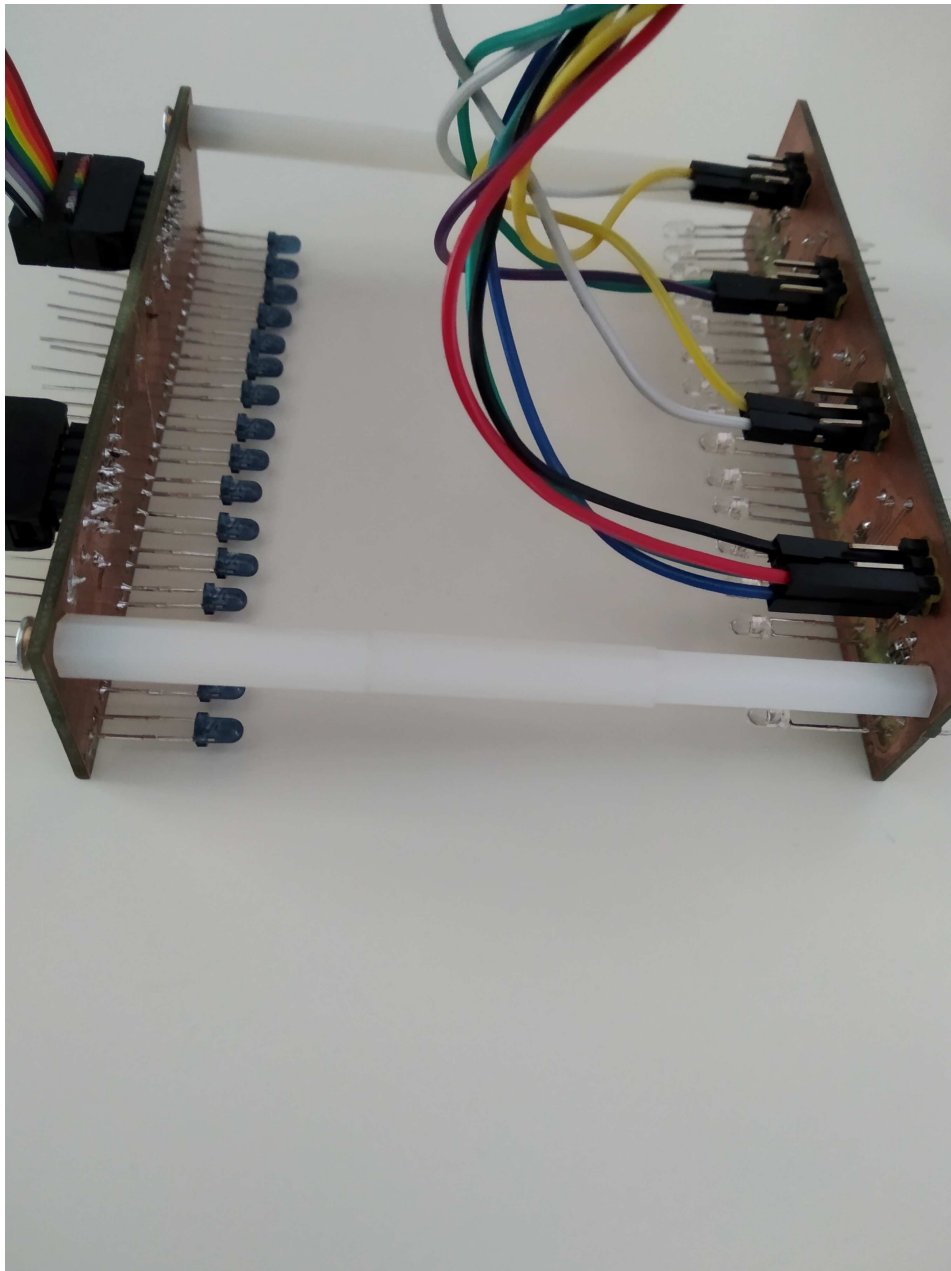


Obr. B.5: Osazení součástek na spodní vrstvu DPS s fototranzistory



**Obr. B.6:** Rozlití vodivé vrstvy na horní straně DPS s mikroprocesorem





Obr. B.7: DPS-Zleva LED a zprava fototranzistorů

# Příloha C

## Použité skripty, zdrojové kódy

### C.1 Skript Lokalizace.m

Skript lokalizace v Matlabu, ve kterém se volají funkce jednotlivých algoritmů:

```
1
2 %pro natoceni videa
3 close all
4
5 vidObj=VideoWriter('moje_videjko.avi');
6
7 vidObj.FrameRate = 60; % pocet snimku/s
8
9 vidObj.Quality = 100;
10
11 open(vidObj);
12
13
14 videoFReader = VideoReader('640x.avi','CurrentTime',1);
15
16 k=1;
17
18 pozice=[]; %vektor do kt se ukladaji pozice
19
20 time=[]; %vektor do kt se ukladaji casy
21
22 load ('template.mat'); % sablona robota
23 temp=template;
24
25 %load ('empty.mat'); % prazdna arena potrebna pro algoritmus uplne BM
26 while hasFrame(videoFReader)
27
28     vidFrame = readFrame(videoFReader);
29
30     img = vidFrame;
31     imgBW= rgb2gray(img);
32     imgbw=imcrop(imgBW,[88 18 412 438]); % 377 3 1000 980 je pro video 60FPS
33     % jinak 88 18 412 438
34
35     %emptybw=imcrop(empty1,[377 3 1364 959]); %oriznuti snimku prazdne areny
36     %emptybw=empty; %emptybw1 nebo empty
37     tic
38
39     %[x,y]=search_max_BM(imgbw,emptybw, template2); %% algoritmus BMA kolem max %%
40
41
42     % Matices=ones(size(imgbw,1),size(imgbw,2)); %% algoritmus Full BMA %%
43     % mindifpos=BM(Matices,imgbw,template);
44     % y=mindifpos(:,1);
45     % x=mindifpos(:,2);
46
```

```
47
48     [x,y]=SURF_BM(imgbw,temp); %template
49
50     %[x, y]=integral_image(imgbw, temp, emptybw); %%% algoritmus integralni obraz %%%
51
52     time(k)=toc % ukladani casu trvani lokailzace kazdeho snimku a potom zprumeruje cas
53     %%%%%%%%%%% jen pro Full BM %%%%%%%%%%%
54     %     x=(x+(size(template,2))/2);
55     %     y=(y+(size(template,1))/2);
56     %%%%%%%%%%%
57
58     formatSpec='souradnice strelu x= %.1f , y= %.1f\n';
59     fprintf(formatSpec,x,y)
60
61     k=k+1;
62     pozice(k,:)=[x,y];
63
64     figure(5);
65     imshow(imgbw);
66
67     hold on
68
69     plot(pozice(:,1),pozice(:,2), 'red', 'linestyle', '-','LineWidth',4)
70     rectangle('Position',[x-(size(template,2))/2,y-(size(template,1))/2,size(template,2),size(template,1)]);
71     plot(x,y,'r*')
72     drawnow
73
74     hold off
75
76     currFrame=getframe(gcf);
77     writeVideo(vidObj,currFrame)
78
79 end
80 save('pozice.mat','pozice');
81 close(vidObj);
82 fprintf('Prumerny cas je %.2f s\n ', mean(time))
83
```