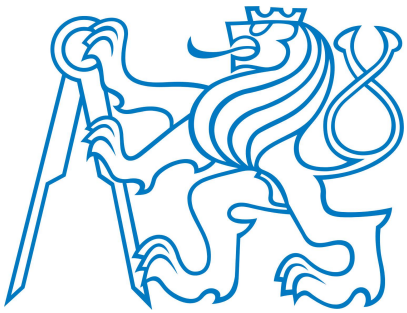
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta stavební

obor Geodézie, kartografie a geoinformatika

program Geodézie a kartografie



BAKALÁŘSKÁ PRÁCE

Vývoj zařízení pro automatickou horizontaci

Development of automatic levelling device

Autor: Štěpán Hodík

Vedoucí práce: Ing. Zdeněk Vyskočil, Ph.D.

Praha, 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Hodík Jméno: Štěpán Osobní číslo: 439259

Zadávající katedra: Katedra geomatiky

Studijní program: Geodézie a kartografie

Studijní obor: Geodézie, kartografie a geoinformatika

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce: Vývoj zařízení pro automatickou horizontaci

Název bakalářské práce anglicky: Development of automatic levelling device

Pokyny pro vypracování:
1. Vytvoření software pro digitální snímání libely a detekci plynové bubliny
2. Vyhodnocení náklonu na základě polohy plynové bubliny
3. Realizace náklonu
4. Vytvoření uživatelského rozhraní v prostředí MATLAB

Seznam doporučené literatury:

Jméno vedoucího bakalářské práce: Ing. Zdeněk Vyskočil, Ph.D.

Datum zadání bakalářské práce: 23.2.2017 Termín odevzdání bakalářské práce: 28.5.2017
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

[redacted]
Podpis vedoucího práce

[redacted]
Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v bakalářské práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce" a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací".

[redacted]
Datum převzetí zadání

[redacted]
Podpis studenta(ky)

ABSTRAKT

Tato bakalářská práce pojednává o tvorbě programu pro automatickou horizontaci přístroje ve výpočetním prostředí Matlab R2014a. Určení náklonů probíhá pomocí snímání krabicové libely digitálním mikroskopem a následné detekce kružnic na pořízených digitálních snímcích. Dále je zde popsáno technické řešení srovnávací desky, která je po zjištění náklonů urovnána.

KLÍČOVÁ SLOVA

Automatická horizontace, detekce kružnic ve snímku, Houghova transformace, počítačové vidění, MATLAB, Arduino.

ABSTRACT

This bachelor thesis deal with creation of a program for automatic leveling in the computing enviroment Matlab R2014a. Determination of tilt is done by scanning spirit level with a digital microscope and subsequent detection of circles on captured digital images. Here is also described the technical solution of the comparative board, which is straightened after finding of the leans.

KEYWORDS

Automatic leveling, circle detection in digital image, Hough transform, computer vision, MATLAB, Arduino.

Prohlášení

Já, Štěpán Hodík, prohlašuji, že jsem autor této bakalářské práce, kterou jsem napsal pod vedením Ing. Zdeňka Vyskočila, Ph.D., za použití zdrojů uvedených v seznamu.

Souhlasím s dalším využitím mé práce i programu na ČVUT Fakultě stavební, v souladu se zákonem č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Praze dne ....

Štěpán Hodík

Poděkování

Na tomto místě bych rád poděkoval své rodině, která mě nejenom při psaní této práce, ale po celou dobu mého studia velmi podporovala. Hlavní poděkování patří mému vedoucímu Ing. Zdeňku Vyskočilovi, Ph.D., jenž svým odborným vedením, ochotou a podnětnými návrhy přispěl k dokončení této práce největším dílem. Dále také panu Jiřímu Emlerovi, bez jehož zručnosti s obráběcími stroji či tavnou pistolí by byl tento úkol neproveditelný. V neposlední řadě bych chtěl poděkovat své přítelkyni za podporu a trpělivost, kterou projevila při psaní této práce.

Štěpán Hodík

Obsah

Úvod

Už od pradávna si člověk výrobou jednoduchých nástrojů snažil ulehčit a zefektivnit svou práci. S příchodem průmyslové revoluce začalo docházet k přechodu od ruční práce k tovární velkovýrobě za pomocí strojního zařízení, které výrazně zvýšilo produkci a zjednodušilo pracovní procesy. Tyto přístroje však musely být ovládány a kontrolovány lidmi. Až s příchodem počítačové techniky mohli vzniknout tzv. automatizované stroje.„*Pojmem automatizace myslíme použití samočinných řídících systémů k řízení technologických zařízení a procesů. Z pohledu industrializace jde o krok následující po mechanizaci. Zatímco mechanizace poskytuje lidem k práci zařízení, které jim usnadňuje práci, automatizace snižuje potřebu přítomnosti člověka při vykonávání určité činnosti*“ [4]. Automatizace v současnosti zastává významnou roli ve všech lidských činnostech, stroji řízenými „počítači“ jsme doslova obklopeni a život bez jejich pomoci si nedokážeme představit. Příkladem může být pračka, automat na jízdenky nebo kávovar. Nejvíce se však automatizace projevila ve výrobní oblasti.

Cílem této práce je vytvořit systém pro automatickou horizontaci přístroje (dále jen AHP), to znamená uvedení vertikální osy přístroje do svislé polohy. Určení náklonů přístroje lze provádět dvěma způsoby, a to pomocí:

a) elektronických snímačů náklonu (např. inerciální měřící jednotka) nebo

b) analogového určení - tzn. odečtením náklonu na libele.

Rozhodl jsem se pro využití druhého řešení, tedy odečítání náklonů z krabicové libely. Hlavním důvodem pro tuto volbu je finanční nákladnost elektronických snímačů dosahujících požadované přesnosti a také proto, že jsem v tomto systému chtěl využít další oblast automatizace a to „počítačové vidění“ (z angl. computer vision). Cílem počítačového vidění je alespoň částečné napodobení lidského vidění využitím technických prostředků, v tomto případě použitím digitálního USB mikroskopu. Bylo tedy nutné vytvořit počítačový program, který komunikuje s dalšími připojenými přístroji. Pomocí USB mikroskopu pořizujeme snímky krabicové libely, které jsou následně programem vyhodnoceny. Na snímku detekujeme dvě kružnice. První z detekovaných kružnic je jedna ze soustředných kroužků krabicové libely označující její střed. Druhá detekovaná kružnice charakterizuje bublinu libely. Následným porovnáním pozic středů detekovaných kružnic a ze znalosti citlivosti libely[1] určíme velikosti náklonů. Výsledné urovnání probíhá pomocí krokových motorů, které otáčí stavěcími šrouby trojnožky.

Tato práce je členěna do tří logických celků. V prvním z nich si představíme hardware (Arduino Uno, digitální mikroskop,krokové motory) a software (Matlab R2014a, Arduino IDE) potřebný k vytvoření systému AHP. Poté si alespoň částečně objasníme problematiku počítačového vidění a vysvětlíme použitou metodu detekce

[1] Citlivost libely (často chápána jako její přesnost) je definována jako změna úhlu potřebného pro pohyb bubliny v libele o určitou vzdálenost.

objektů v digitálním obraze. A nakonec si popíšeme tvorbu řídícího programu a sestrojení zařízení pro realizaci náklonů.

Část I
Hardware a software

1 MATLAB

MATLAB je velmi výkonné interaktivní programové prostředí a skriptovací jazyk čtvrté generace vyvíjený společností MathWorks, která ji dodává s celou řadou rozšíření a knihoven funkcí, v prostředí MATLABu označovaných jako toolboxy. Je nutno dodat, že MATLAB není free software a tyto knihovny (např. Robotics System Toolbox nebo Robotics System Toolbox) je nutno spolu se základní verzí programu zakoupit [14]. V této práci byla použita verze MATLAB R2014a spolu s několika toolboxy v rámci akademické licence ČVUT.

Hlavní vlastnosti programu:

- Jazyk vysoké úrovně pro vědecké a inženýrské výpočty, vizualizaci a vývoj aplikací
- Integrované grafické prostředí pro vizualizaci dat a tvorbu vlastních grafů
- Aplikace pro prokládání křivek, analýzů a zpracování signálů, řízení systémů
- Nástroje pro tvorbu vlastních aplikací s uživatelským rozhraním (GUI)
- Rozhraní pro propojení s programy a jazyky jako C/C++,Java®, Python, SQL, .NET a Microsoft Excel
- Možnost sdílení programů MATLAB koncovým uživatelům bez licenčních poplatků

1.1 Historie

MATLAB byl původně volně dostupný program vyvinutý koncem sedmdesátých let profesorem Cleverem Molerem, tou dobou působícím na Univerzitě v Novém Mexiku. Program vytvořil pro potřeby svých studentů, aby mohli používat výpočetní knihovny LINPACK a EISPACK bez nutné znalosti programovacího jazyku Fortran[2]. Jazyk se však brzy stal populárním mezi velkým počtem univerzit, konkrétně v oblasti aplikované matematiky. Průlom proběhl roku 1983, kdy se ing. Jack Little dostal do kontaktu s programem MATLAB a uvědomil si jeho skvělý komerční potenciál. Spojil se tedy s profesorem Molerem a Steveem Bangertem a začali program přepisovat do jazyku C. Již roku 1984 založili společnost MathWorks, která dodnes software MATLAB spravuje a rozšiřuje [7].

[2]Fortran (neboli **FOR**mula **TRAN**slation) je programovací jazyk vyvinutý společností IBM v padesátých letech minulého století pro vědecké výpočty a aplikace.

1.2 Představení programu

V této podkapitole si ukážeme a popíšeme grafické okno programu. Otevřené okno tohoto programu může vypadat přibližně takto (viz obr. 1.1).

Hlavní části okna:

- Command Window:

  Je okno pro zadávání příkazů MATLABu a následné zobrazení výsledků. Zadaný příkaz se potvrzuje klávesou ENTER, je však velmi nepraktické zadávat své příkazy pouze do tohoto okna, jelikož při vypnutí nebo pádu programu dojde ke ztrátě našeho postupu. Z tohoto důvodu jsou v prostředí MATLAB využívány tzv. *m-soubory*, jsou to textové soubory se zapsanou posloupností kódů. Soubory lze psát i upravovat v obyčejných textových editorech (např. v PSPadu nebo Poznámkovém bloku), s výhodou lze však využít integrovaného editoru MATLABu, ze kterého můžeme kód spustit.

- Current Folder:

  V tomto okně se zobrazují soubory obsažení v pracovní složce. Tyto soubory (ať už skripty/funkce MATLABu, obrázky atd.) můžeme spouštět pomocí *Command Window* nebo v jednotlivých m-souborech. To znamená, že námi vytvořené funkce a skripty, které chceme používat v rámci jednoho projektu, musíme mít umístěné ve stejné složce.

- Workspace:

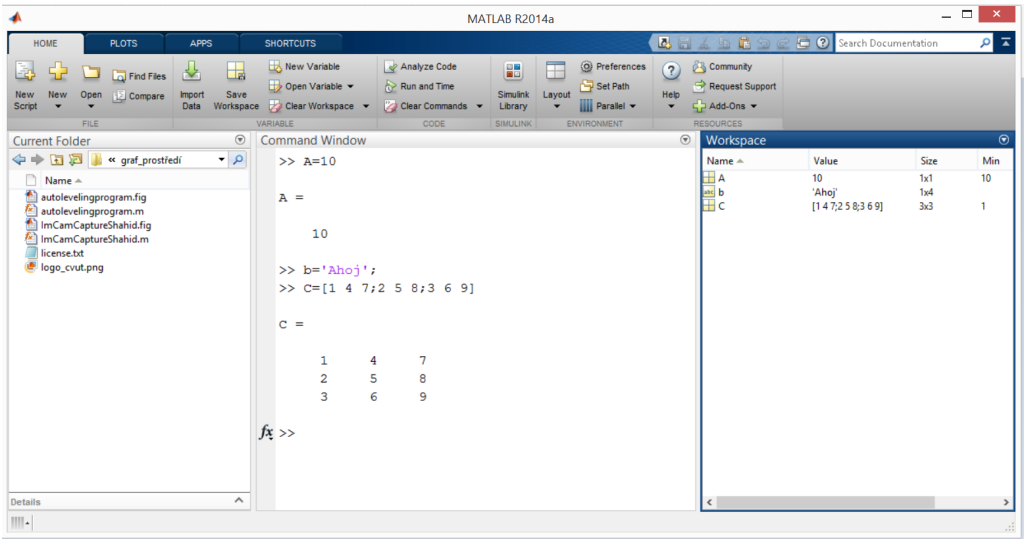
  Zde se zobrazují všechny dostupné proměnné pracovního prostředí, které byly inicializovány buď jako na obrázku 1.1 z *Command Window*, nebo v některém z m-souborů. Proměnné se zde dají vytvářet, editovat či mazat. V tomto okně si také můžeme zapnout zobrazení velikosti matic, typu uložených hodnot, minimální a maximální hodnoty v proměnné a další parametry popisující proměnnou, které mohou usnadnit hledání chyb v kódu.

- Pás nástrojů:

  Pás nástrojů byl přidán až od verze programu MARLAB R2012b, obsahuje reorganizované funkce, aplikace a nastavení, které se dříve nacházely v menu a panelech nástrojů. Pás nástrojů je organizován do čtyř záložek *HOME, PLOTS, APPS a SHORTCUTS* v rámci každé z nich dále řazeny do podsekcí [15].

1.3 Toolboxy a pomocné balíčky

Jak již bylo zmíněno výše, funkce MATLABu jsou tříděny do jednotlivých knihoven (toolboxů) podle jejich využití. Toolbox si lze představit jako adresář s „m-soubory“ obsahující funkce, které rozšiřují použití programu v příslušných vědních a technický oborech.



Obrázek 1.1: Náhled okna MATLAB R2014a

Další rozšíření je možné pomocí instalace podpůrných balíčků (support package), které poskytnou MATLABu možnost využít hardwarové a softwarové vybavení produktů třetích stran, jako například:

- USB webcam
- Arduino hardware
- iOS sensors
- Microsoft Kinect for Windows Support from Image Acquisition Toolbox

V následujících podkapitolách si pár vybraných knihoven představíme a uvedeme postup instalace podpůrných balíčků [18].

Image Processing Toolbox

Tato knihovna poskytuje komplexní sadu standardních algoritmů pro zpracování, analýzu, vizualizaci a vývoj algoritmů. S touto knihovnou lze provádět segmentaci, vylepšení obrazu, redukci šumu, geometrickou transformaci či zpracování 3D snímků [9].

Image Acquisition Toolbox

Pomocí této knihovny lze k softwaru MATLAB a Simulinku připojit průmyslové a vědecké kamery. Také obsahuje aplikaci pro detekci a konfiguraci připojeného hardwaru. Pro fungování funkcí této knihovny je nezbytné mít současně nainstalovaný i *Image Processing Toolbox* [8].

Ukázka připojení kamery:

- Vyhledání všech připojených zařízení k systému Windows.

```
>> info = imaqhwinfo('winvideo')

info =
```

```
        AdaptorDllName: 'C:\MATLAB\Sup...'
    AdaptorDllVersion: '4.7 (R2014a)'
          AdaptorName: 'winvideo'
            DeviceIDs: {[1]  [2]}
           DeviceInfo: [1x2 struct]
```

- Každému zařízení je přiděleno identifikační číslo `DeviceID`, pomocí kterého se na toto zařízení odkazuje při získání informací o něm nebo jeho připojení.

```
dev_info = imaqhwinfo('winvideo',1)

dev_info =

              DefaultFormat: 'MJPG_1280x720'
        DeviceFileSupported: 0
                 DeviceName: 'Integrated Webcam'
                   DeviceID: 1
      VideoInputConstructor: 'videoinput('winvideo', 1)'
    VideoDeviceConstructor: 'imaq.VideoDevice('winvideo', 1)'
          SupportedFormats: {1x11 cell}
```

- Získané informace o našem zařízení vložíme do funkce `videoinput`, která vytvoří objekt pro vstup videa.

```
>> vidObj=videoinput('winvideo',1,'MJPG_1280x720')

Summary of Video Input Object Using 'Integrated Webcam'.

   Acquisition Source(s):  input1 is available.

  Acquisition Parameters:  'input1' is the current selected source.
                           10 frames per trigger using the selected source.
                           'MJPG_1280x720' video data to be logged upon START.
                           Grabbing first of every 1 frame(s).
                           Log data to 'memory' on trigger.

      Trigger Parameters:  1 'immediate' trigger(s) on START.

                  Status:  Waiting for START.
                           0 frames acquired since starting.
                           0 frames available for GETDATA.
```

- Pomocí vytvořeného video objektu (`vidObj`) pak MATLAB komunikuje s daným zařízení. Chceme-li pořídit snímek, použijeme funkci `getsnapshot`. V případě, že chceme spustit živý náhled videa snímaného zařízení využijeme funkce `preview`.

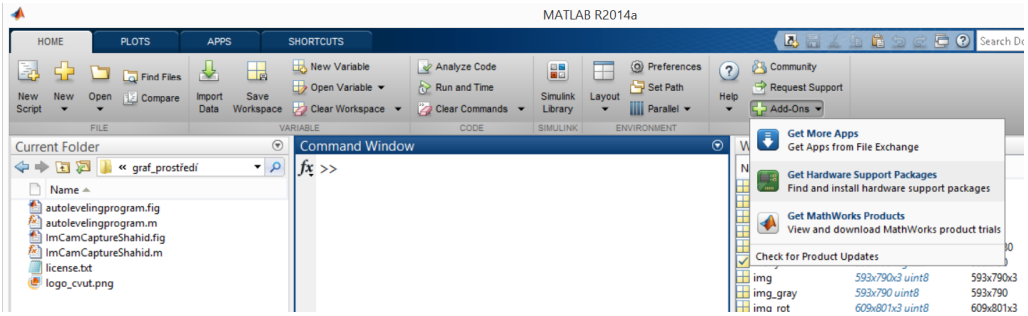
```
>> img=getsnapshot(vidObj);
>> preview(vidObj)
```

Instalace pomocných balíčků

Pro náš systém AHP budeme potřebovat balíky pro komunikaci s USB webkamerami a Arduino UNO deskou[3]. Možnost přidání balíčku najdeme v záložce *HOME* v sekci *ENVIROMENT* tlačítkem *Add-Ons* a volbou *Get Hardware Support Packages* (viz obr. 1.2) nebo prostým zapsáním příkazu `supportPackageInstaller` do *Command Window* [19].
První volbou instalátoru jsou následující čtyři možnosti:

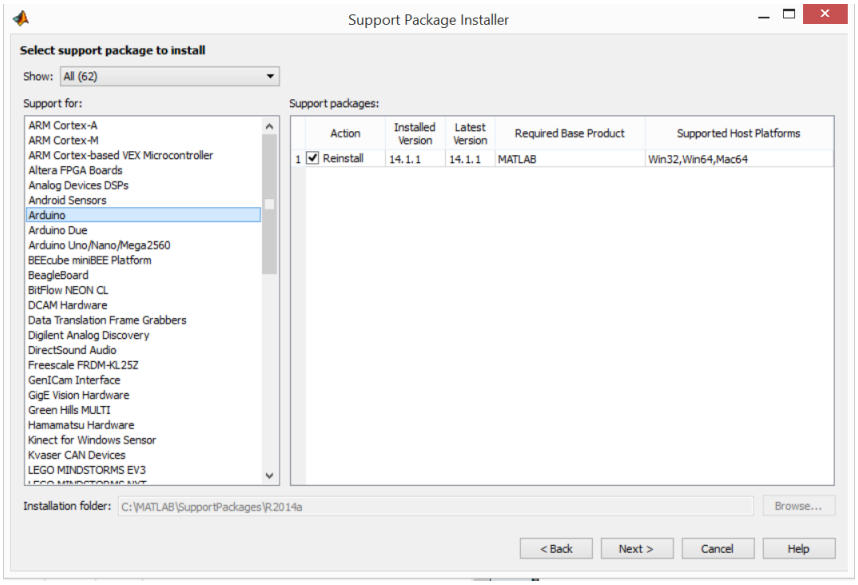
[3] Arduino je vývojová deska s mikroprocesorem, která je využívaná k tvorbě autonomních projektů. Více v kapitole **Arduino**.

- *Install from Internet:* Stažení z internetu a následná instalace podpůrného balíčku na hostitelském počítači.
- *Download from Internet:* Stažení podpůrného balíčku bez instalace. Soubory jsou ukládány do výchozí složky v mém případě `C:\MATLAB\SupportPackages\R2014a\downloads`.
- *Install from folder:* Instalace dříve stažených podpůrných balíčků.
- *Uninstall:* Odinstalace vybraných podpůrných balíčků.



Obrázek 1.2: Spuštění instalátoru podpůrných balíčků

Potvrzením první volby se nám zobrazí seznam všech podpůrných balíčku dostupných od společnosti MathWorks (viz obr. 1.3). Před instalací je ještě nutné se přihlásit k uživatelskému účtu MathWorks. Můžeme však využít i jiných zdrojů a pak bychom pro instalaci využili volby *Instal from folder*.



Obrázek 1.3: Výběr potřebných balíčků

2 Arduino

Arduino je open-source vývojová platforma z velké části založená na mikroprocesorech ATMega od firmy Atmel. Slouží k řízení a kontrole připojených elektronických součástek a senzorů, díky nimž lze vytvořit nesčetné množství projektů.

Arduino získává informace od různých snímačů a senzorů (například snímač vzdálenosti, itenzity světla nebo obyčejné tlačítko) a pomocí těchto údajů ovládá výstupy (rozsvítí LED diodu, zastaví motor nebo zobrazí hodnotu na displej) [29]. Vývojových desek Arduino existuje celá řada a každá z nich má svá specifika pro využití v jednotlivých projektech, ať už se jedná a rozměry desky, počet výstupních pinů nebo velikost paměti (přehled jednotlivých desek zde [17]).

Jelikož se jedná o otevřený projekt a schémata jednotlivých desek jsou volně přístupná, objevují se na trhu tzv. **Arduino klony** (např. FreeDuino,LABduino)[4], které více či méně dosahují kvalit originálu a jejichž hlavní předností je nižší cena. Je však mnoho klonů, které se pouze nesnaží napodobit původní desky, ale přizpůsobit je, ať už svým tvarem, nebo parametry k specifickému účelu. Příkladem mohou být desky:

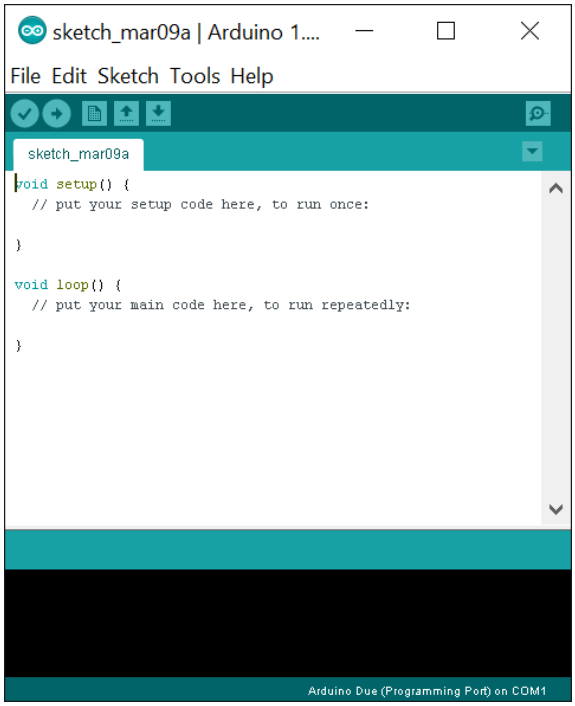
- ArduPilot – navržený pro ovládání autonomních létajících zařízení (letadla, kvadrokoptéry atd.),
- Rainbowduino – klon Arduina, který je vytvořen pro ovládání maticových RGB LED displejů,
- Bluno – jedná se o klon desky Arduino UNO , která je doplněna o modul podporující technologii bluetooth 4.0,
- A další (viz zde [3]).

Pokud nám základní Arduino deska nestačí, můžeme jí doplnit o tzv. **Arduino shield**. Shield (česky „štít“) je v podstatě kompletní řešení pro rozšíření Arduina o nějakou další nadstavbu. Jedná se o desku, která má stejné rozmístění pinů jako základní deska a jednoduchým nasazením na ní získáte rozšíření, které daný shield nabízí a je většinou jasný už z názvu shieldu (Ethernet shield, Wifi shield, Motor shield, GPS shield a další)[29]. Pokud však nechceme investovat do sériově vyráběných shieldů, můžeme si vyrobit vlastní, jak bude ukázáno v kapitole *Připojení a ovládání krokových motorů* .

Ovšem aby Arduino deska pracovala podle našich představ, musíme nejdříve vytvořit program pro Arduino mikrokontrolér (jednočipový počítač). Arduino je možné programovat v jazyce C nebo C++. Nejrozšířenějším způsobem programování je však použití knihovny C++ *Wiring*. Díky komplexnosti této knihovny je někdy označovaná jako samostatný programovací jazyk. Kód napsaný v tomto jazyce je pak následně přeložen a nahrán do mikroprocesoru z prostředí Arduino IDE (*Integrated development enviroment* obr.2.1) [29].

Tvorba projektů s Arduino deskami je velmi obsáhlá a vydala by na mnoho knih, touto kapitolou jsem se pouze snažil vysvětlit, co to Arduino je a nastínit jeho možnosti. Pro ty, kteří by se chtěli o Arduinu dozvědět víc, bych odkázal na tyto zdroje [29] [1] [2]. Nyní si představíme dvě zajímavé Arduino desky.

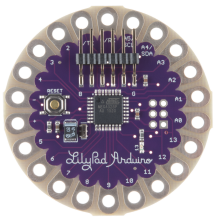
[4]Název Arduino je chráněný autorskými právy, avšak „-duino“ a podobné části jsou v názvu přípustné [29].



Obrázek 2.1: Vývojové prostředí Arduino IDE

2.1 LilyPad Arduino

Vývojová deska LilyPad Arduino se už na první pohled (viz obr. 2.2) od ostatních velmi liší. Důvod zmínky o ní není ten, že bychom jí v našem systému AHP použili (bylo využito desky Arduino Uno), nýbrž ten, že je to skvělá ukázka přizpůsobení tvaru a zpracování desky pro své využití. Je navržena pro tvorbu takzvaných wearable zařízení, tedy k tomu, že si LilyPad Arduino přišijeme k oblečení vodivou nití. To může být užitečné k tomu, že si vyrobíme cyklistickou mikinu s přišitými blinkry nebo svítící oblek pro psa [29]. Je tedy vidět, že možnosti projektů jsou omezeny pouze naší představivostí.



Obrázek 2.2: Vývojová deska LilyPad Arduino

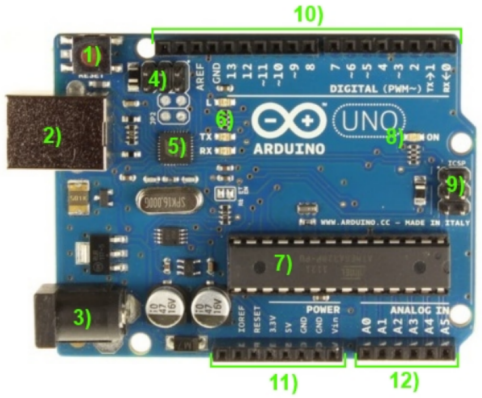
2.2 Arduino Uno

Nyní se blíže podíváme na desku Arduino Uno, která je jednou z nejvíce používaných desek. Hlavním důvodem je dostatečný počet vstupních a výstupních pinu pro zapojení dalších periférií, velikost paměti pro nahrání našeho kódu, obsažení USB konektoru a také přiměřená cena, která se pohybuje v řádu několika stokorun. To z desky dělá vhodného kandidáta pro většinu základních projektů. V

systému AHP bude využita jako řídící jednotka krokových motorů.

Popis jednotlivých částí desky (viz obr. 2.3)[29]:

1. Resetovací tlačítko – využijeme ho tehdy, chceme-li pustit kód od začátku.
2. USB konektor – slouží k připojení k PC pro nahrání kódu, vytvoření seriové komunikace nebo k napájení.
3. Napájecí konektor – pokud nechceme mít desku připojenou pomocí USB.
4. ICSP – hlavice pro externí programování USB-serial převodníku.
5. USB-serial převodník – stará se o komunikaci mezi hlavním čipem a PC.
6. Indikační LED – bliká pokud probíhá komunikace přes sériovou linku.
7. Hlavní čip desky.
8. Indikační LED – svítí pokud je deska připojená k napájení.
9. ICSP – hlavice pro externí programování hlavního čipu.
10. Digitální piny – slouží k zapojení různých el. součástek a snímačů.
11. Napájecí výstupy desky.
12. Analogové piny – slouží k připojení vodičů, na kterých budeme chtít měřit napětí.



Obrázek 2.3: Vývojová deska Arduino Uno - popis součástí - zdroj [29]

3 USB mikroskop

K pořizování snímků bylo využito ručního digitálního mikroskopu Celestron II (viz obr. 3.1). Nejedná se o mikroskop pro vědecké účely, přesto je pro náš projekt plně dostačující. Součástí balení je i stojánek, který ale nakonec použit nebyl. Bylo nutné zajistit pevnou polohu mikroskopu i při jeho vyndání a opětovném umístění.

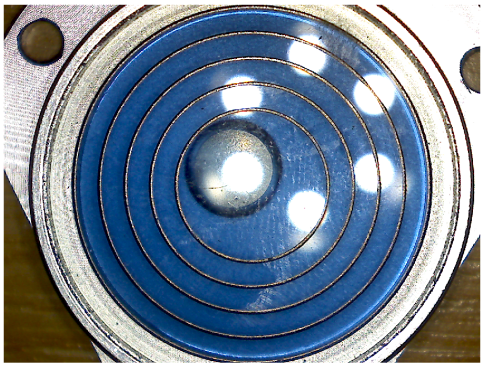
Zvětšení	10 x – 150 x
Snímač	2 Mpx (CMOS)
Rozměry	108 mm x 32 mm
Osvětlení	6 LED diody

Tabulka 3.1: Parametry digitálního mikroskopu - zdroj [6]

Obrázek 3.1: Digitální mikroskop Celestron II - zdroj [6]

Parametry mikroskopu jsou uvedeny v tabulce 3.1.

Jediná nevýhoda tohoto mikroskopu je jeho osvětlení, které je zajištěno pomocí šesti LED diod. Problém nastává při snímání odrazivých předmětů, jako je sklo, kov a dalších. Kde dochází k odrazu bodových zdrojů a tím znehodnocení snímků, jak je vidět na obrázku 3.2. Problém by se dal vyřešit vložením difuzoru (např. mléčného skla) před LED diody a tím částečně vytvořit plošný zdroj. Tento problém byl prozatím vyřešen zakrytím diod a osvětlování libely bylo zajištěno rozptýleným světlem ze zářivek v laboratoři.



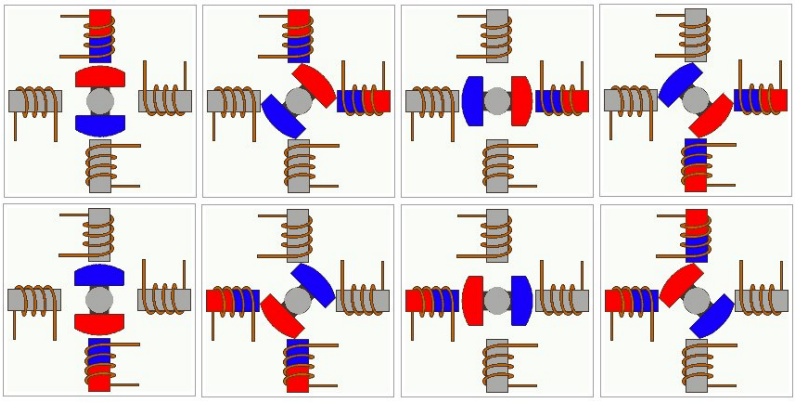
Obrázek 3.2: Snímek mikroskopu při nezakrytém osvětlení

4 Krokový motor

4.1 Popis a princip krokového motoru

Jedná se o synchronní točivý stroj, obvykle napájený impulsy stejnosměrného proudu. Na rozdíl od klasického DC motoru můžeme u krokového motoru nastavit pozici motoru a tu následně silou udržet, což má však za následek neustálý odběr proudu. Nejčastěji se s malými krokovými motory lze setkat v kancelářské technice (PC, tiskárny, faxy, atd.). Výkonnější krokové motory jsou používané v průmyslu, především v tiskařských, dřevoobráběcích, manipulačních a CNC strojích [10].

Motor se skládá ze dvou základních prvků - z *statoru* (elektromagnet) a *rotoru* (permanentní magnet). Statorové vinutí je buzeno tak, aby vytvářelo magnetické póly opačné k pólům rotoru a docházelo tak k jeho otáčení. Princip je naznačen na obrázku 4.1, který zobrazuje osm po sobě jdoucích kroků (po řádcích zleva doprava). V prvním kroku je rotor natočený k horní cívce, ke které je přivedeno napětí. V druhém kroku je využito tzv. „*půl-krokování*“, kdy k otočení rotoru o úhel 45° ve směru hodinových ručiček jsou vybuzeny (připojeny k napětí) zároveň horní i pravá cívka. Buzením jednotlivých cívek ve správném pořadí tedy dosáhneme otáčení motoru. Rotor i stator mohou mít hned několik „*mini-pólů*“, s jejichž zvyšujícím počtem se snižuje krokový úhel (úhel otočení rotoru při jednom kroku) [25]. Je jasné, že buzení jednotlivých cívek nemůžeme provádět ručně, proto je k chodu motoru nezbytné jeho připojení k řídící jednotce, která bude za nás „budit“ jednotlivé cívky ve správném pořadí a v určitých časových intervalech, díky čemuž lze definovat rychlost otáčení motoru. Krokové motory se dělí do dvou základních skupin podle řízení na :



Obrázek 4.1: Princip krokového motoru - zdroj [11]

1. **Unipolární** - U unipolárního řízení je vždy v jednom okamžiku buzena právě jedna cívka statorového vinutí. Krokový motor s tímto buzením má menší odběr, ale také menší kroutící moment.[25]

2. **Bipolární** - U bipolárního řízení krokového motoru je vždy buzeno nejméně jedno ze statorových vinutí. Krokový motor při tomto buzení poskytuje vetší kroutící moment, ovšem za cenu vyšší spotřeby a složitosti napájecích a řídících obvodů.[25]

4.2 Použitý krokový motor

Pro otáčení stavěcích šroubů jsou použity dva krokové motory s označením **M35SP-7** (viz obr.4.2), které byly odebrány z vyřazených skenerů. Součástí motorů byl i převodový systém, který na nich byl ponechán. Úhlový krok těchto motorů je 7.5°, a tedy pro otočení motoru o 360° je nutné učinit 48 kroků. Původně jsou tyto motory konstruovány jako unipolární s pěti přívodními kabely. Pro zvýšení kroutícího momentu motoru byla však společná zem cívek odebrána, díky čemuž se řízení motoru stalo bipolární.

Obrázek 4.2: Náhled použitého krokového motoru

Část II
Počítačové vidění

5 Co to je počítačové vidění?

Počítačové vidění (někdy označované jako strojové vidění nebo zjednodušeně analýza obrazu) je oblast výpočetní techniky a vývoje softwaru zabývající se získáváním informací z obrazu, ať už se jedná o jeden snímek, sekvenci snímků (video),či stereo-snímky. Software se nejenže snaží napodobit lidské vidění, ale i následné zpracování obrazu mozkem a reakci na zjištěné podměty [22]. V dnešní době se však už nemůžeme bavit jen o napodobení, ale spíše o převýšení nad lidským viděním. Nemusíme totiž snímat pouze ve viditelném spektru, jak je tomu u lidského zraku, ale můžeme snadno využít například termálních družicových snímků pro lokalizaci ložisek požáru nebo rentgenových, ultrazvukových či tomografických snímků, z kterých můžeme získat data o velikosti orgánů, toku krve nebo detekci nádorů.

Příklady dělení úloh počítačového vidění [28] :

- **Rozpoznání:**
  Klasickým problémem v oblasti počítačového vidění je zjistit, zda obrazová data obsahují konkrétní objekt, vlastnost nebo činnost. Které pak následně identifikujeme a poznatky dále zpracujeme. Problém rozpoznání můžeme dále dělit do dalších podskupin:

  - **Poznání objektu:**
    V softwaru je předem specifikovaná vlastnost objektu nebo objektů řazených do tříd (např. kolo, míč atd.) a po zpracování obrazu dojde k jeho zařazení k příslušné třídě prvků.
  - **Identifikace:**
    Identifikací rozumíme rozpoznání konkrétního objektu. Příkladem mohou být aplikace na rozpoznání obličeje, otisků prstů nebo přečtení čárového kódu.
  - **Detekce:**
    Při detekci jsou v obrazových datech hledány prvky, které splňují předem dané podmínky. Podmínkami mohou být tvar hledaného objektu, barva či velikost.

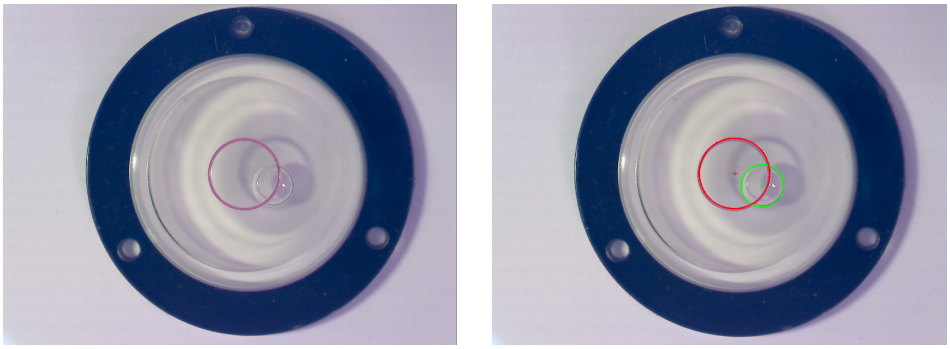
- **Analýza pohybu:**
  Jak je již z názvu jasné, tato oblast počítačového vidění se zabývá detekcí objektů a zkoumáním jejich směru pohybu, rychlosti atd.

- **Rekonstrukce scény:**
  Při snímání objektu dochází k zachycení prostorového objektu na rovinu

snímku. Rekonstrukce scény se zabývá opačným postupem, kdy s využitím více snímků dokážeme vytvořit 3D model snímaného objektu.

- **Restaurace obrazu:**
  Cílem restaurace obrazových dat je odstranění nežádoucích jevů, jako jsou šum senzoru nebo rozmazání. Snažíme se o potlačení porušení obrazu na základě znalosti charakteru poruchy nebo jejího odhadu.

V tomto textu se dále budeme zajímat pouze o detekci prvku v obraze, což bude klíčové pro vytvoření našeho systému AHP. Jak již bylo zmíněno v úvodu, určení náklonů bude probíhat tak, že na pořízeném digitálním snímku budeme detekovat obraz bubliny a střed libely, který je na libele charakterizován jako střed soustředného kroužku. Pokud chceme prvky v obraze detekovat, musíme nejprve určit podmínky, které je dostatečně popisují. Obrázek 5.1 zobrazuje vzorový snímek libely a také stejný snímek se zvýrazněnými detekovanými prvky. Pokud tedy bude osa záběru digitálního mikroskopu kolmá k rovině libely, detekovanými tvary budou kružnice o určitém poloměru. Kdyby nebyla tato podmínka splněna, kružnice by se díky perspektivnímu zkreslení zobrazily na snímku jako elipsy.



(a) **(b)**

Obrázek 5.1: Vzorový snímek krab. libely: a) originální snímek, b) detekované kružnice.

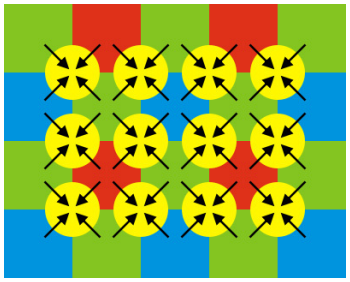
Detekce prvků v obraze se skládá z jednotlivých úkolů:

1. Pořízení snímku
2. Segmentace
3. Detekce

6 Pořízení snímku

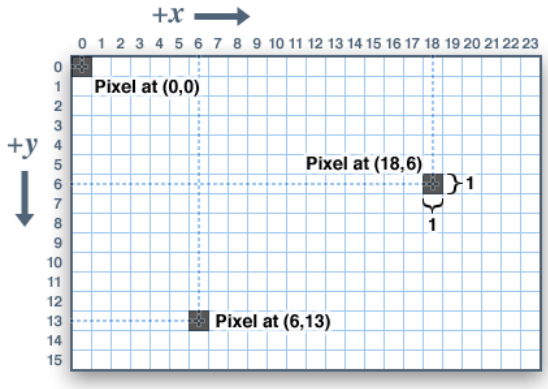
Vznik digitálního obrazu spočívá v převodu analogového signálu měřeného na snímacím čipu na signál digitální. Snímací čipy jsou součástky obsahující velké množství polovodičových světlocitlivých elementů. Každý z těchto elementů dokáže vyhodnotit pouze intenzitu dopadajícího světla (resp. úroveň jasu), nikoliv barvu. Pro získání informace o barvě je nutné před senzor umístit barevný filtr, který světlo

o stejné barvě propustí a zvýrazní a ostatní barvy filtruje. K vyjádření barvy jednoho pixelu[5] potřebujeme tedy nejméně tři snímače s předsazenými barevnými filtry, nejčastěji červeným, zeleným a modrým (viz Model RGB). Často je však využito tzv. „*Bayerovy masky*“ (obr.6.1), kdy pro výpočet barvy jednoho pixelu je využito čtyř sousedních elementů, přičemž zelená barva je v masce zastoupena dvakrát z důvodu citlivosti lidského oka na tuto barvu [21] [30].



Obrázek 6.1: Bayerova maska

Digitální obraz je obvykle reprezentován maticí hodnot, které můžeme matematicky popsat pomocí spojité skalární funkce f dvou nebo tří proměnných označované jako *obrazová funkce*. V obecném případě je pevný monochromatický obraz popsán obrazovou funkcí dvou proměnných $f(x, y)$, kde x, y jsou souřadnice daného pixelu a hodnota obrazové funkce představuje úroveň jasu. Pokud bychom pracovali s barevným (multispektrálním) obrazem, připadne každé dvojici souřadnic x, y vektor hodnot, např. jasů pro jednotlivé barevné složky. V případě popsání obrazové sekvence (videa) je nutné použít obrazovou funkci tří proměnných $f(x, y, t)$, přičemž proměnná t popisuje změnu času. Souřadnice pixelů jsou vztaženy buď klasicky ke kartézské soustavě souřadnic, nebo používají vlastní souřadnicový systém označovaný jako *image coordinate system*, v kterém je počátek umístěn do levého horního rohu digitálního obrazu, osa x směruje vlevo a osa y dolů (viz obr. 6.2). Tohoto systému je využito i v prostředí MATLAB, s tím rozdílem, že indexování začíná od 1 a ne od 0. Dalšími důležitými pojmy digitalizace obrazu jsou *vzorkování* a *kvantování*[21].

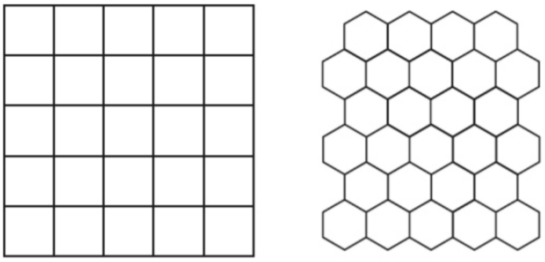


Obrázek 6.2: Souřadnicový systém obrazu

[5]**Pixel** (angl. zkr. picture element) je nejmenší bezrozměrná jednotka digitálního obrazu.

Vzorkování

Vzorkováním rozumíme zaznamenávání hodnot spojité obrazové funkce $f(x, y)$ v předem daných intervalech, díky čemuž vytvoříme matici obrazu s $M \times N$ pixely. Za prvé je nutné zvolit *vzorkovací mřížku*, což je plošné uspořádání bodů při vzorkování(viz obr. 6.3). Nejčastěji se setkáme s klasickou pravidelnou čtvercovou mřížkou, používají se však i mřížky obdélníkové nebo hexagonální. A za druhé je potřeba zvolit *interval vzorkování* neboli vzdálenost mezi nejbližšími vzorkovacími body. Vzorkovací vzdálenost by měla být volena minimálně dvakrát menší, než je velikost nejmenších detailů v obraze [21][26].



Obrázek 6.3: Vzorkovací mřížka - čtvercová (vlevo) a hexagonální (vpravo) - zdroj [21]

Kvantování

Kvantování je proces digitalizace, kdy spojitá hodnota obrazové funkce $f(x, y)$ je diskretizována do k stejných intervalů, také označovaných jako kvantovací úrovně. Pokud je tedy pro uložení jasové informace pixelu použito b bitů, je počet kvantovacích úrovní $k = 2^b$. Obvykle se používá 8-bitového kódování (256 jasových úrovní), tedy pro uložení barevného je nutné každý pixel reprezentovat 24 bity. Počet kvantovacích úrovní je nutné zvolit dostatečně velký, aby nezanikly detaily obrazu a nevznikly falešné obrysy. V některých případech však stačí využít pouze dvou úrovní, které nám definují popředí a pozadí obrazu. Takovýto obraz označujeme jako *binární* a využívá se například při skenování starých map. Na obrázku 6.4 je znázorněna změna obrazu při různých počtech kvantovacích úrovní [21][26].

6.1 Barevné modely

Důležitým parametrem obrazu je jeho barva[6], informaci o všech barevných složkách si nese uloženou každý pixel. Barvy lze vytvářet různými způsoby a právě jejich tvorbu nám popisuje barevný model [30]. Nejběžnější barevné modely digitálního obrazu jsou:

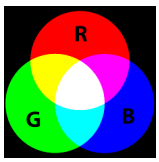
- **RGB**,
- **HSV**,
- **Gray scale**.

[6] **Barva** popisuje schopnost předmětů odrážet světlo různých vlnových délek [30].

Obrázek 6.4: Testovací obrázek Lenna - hodnota N označuje počet úrovní - zdroj (http://www.optique-ingenieur.org)

RGB

V tomto modelu vzniká vytvoření libovolné barvy složením tří základních barev - červené (R,*red*), zelené (G,*green*) a modré (B,*blue*). Každou složku lze tedy vyjádřit celočíselnou hodnotou od $0 - 255$ (při použití 8-bitového kódování). Hodnota 0 znamená, že složka není v barvě zastoupena a maximální hodnota 255 značí, že intenzita složky nabývá největší intenzity. Jedná se o tzv. *aditivní míchání*, kdy se jednotlivé intenzity složek sčítají. Pokud tedy sloučíme barvy o intezitách $(0, 0, 0)$, získáme černou barvu, naopak při plné intenzitě $(255, 255, 255)$ získáme bílou barvu (viz obr. 6.5). Tento model se nejčastěji používá v zobrazovacích zařízeních, jako jsou LCD monitory, projektory a další.[30]



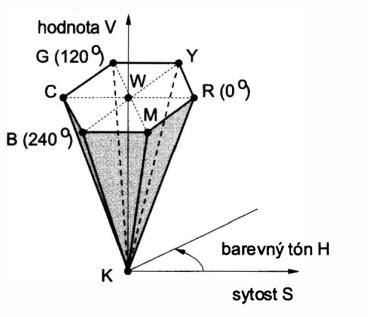
Obrázek 6.5: Barevný model RGB

HSV

Barvy v tomto modelu nevznikají skládáním základních barev, nýbrž volbou tří základních parametrů - *barevného tónu* (H, *hue*), *sytosti* (S,*saturation*) a *jasové hodnoty* (V,*value*). Barevný tón označuje převládající spektrální barvu, sytost příměs ostatních barev a jas příměs bílého světla [30]. Barevný prostor tohoto modelu je často reprezentován šestibokým jehlanem (viz obr. 6.6).

Šedá škála - Gray scale

V některých případech je informace o barvě zbytečná a stačí ukládat pouze informaci o intenzitě jasu daného pixelu. Takovýto obraz se označuje jako *šedotónový*, jehož jednotlivé pixely mohou nabývat hodnot od $0 - 255$ (při použití 8-bitového kódování). RGB obraz je vlastně složen ze tří šedotónových obrazů, kdy každý z nich obsahuje informaci o jasu dané základní barvy.



Obrázek 6.6: Barevný model HSV - zdroj [30]

Převod barevného obrazu RGB do šedotónového obrazu je možný pomocí následující rovnice:

$$I = 0.299R + 0.587G + 0.114B \tag{6.1}$$

I označuje intenzitu jasu nově vypočteného obrazu v odstínech šedi. Je vidět, že se na nové hodnotě podílejí jednotlivé základní barvy různou měrou. Tento fakt vychází z toho, že je oko je různě citlivé na jednotlivé barevné složky (nejcitlivější je na zelenožlutou)[30][22].

7 Segmentace obrazu

Jedním z nejdůležitějších a zároveň nejtěžších kroků při zpracování obrazu je *segmentace obrazu*. Jedná se o rozdělení obrazu na části, které zobrazují námi hledané prvky (oblasti nebo hrany) a jsou pro nás důležité pro další zpracování. Zbývající části obrazu jsou považovány za pozadí obrazu, které již nepřináší další informaci [27]. V následujících podkapitolách si představíme dvě z mnoha metod segmentace a demonstraci funkcí MATLABu pro jejich využití.

7.1 Prahování

Prahování je jednou z nejstarších a nejjednodušších metod segmentace obrazu. Hlavní myšlenkou této metody je, že hledané objekty a pozadí obrazu mají odlišnou intenzitu. Problém tedy spočívá v nalezení takové hodnoty (prahu) obrazové funkce, pro kterou bude platit, že pixely s intenzitou menší, než je zvolený práh, jsou určeny jako pozadí a ostatní jsou považovány za pixely hledaných objektů (viz rovnice 7.1).

$$\begin{aligned} f(i,j) = 1 \qquad & \text{je-li } g(i,j) \geq T, \\ 0 \qquad & \text{jinak.} \end{aligned} \tag{7.1}$$

Hodnota T označuje tzv. *globální práh*, jelikož je jeho hodnota stejná pro celý snímek, což může být nevýhodné u nestejnoměrně osvětlených snímků [27]. Obraz lze prahovat nejen na základě intenzity jedné složky, ale i více, díky čemuž můžeme vybírat prvky z obrazu na základě jejich barvy. V rovnici 7.3 je naznačeno prahování na základě barvy při použití barevného model HSV. Obrazové funkce h, s a v označují hodnoty základních parametrů pro daný pixel o souřadnicích i a j. Následnou volbou

dolního a horního prahu můžeme přesně definovat barvu hledaných prvků v obraze a tím je oddělit od pozadí.

$$\begin{aligned} f(i,j) = 1 \quad & \text{je-li } Th_{min} \leq h(i,j) \leq Th_{max} \\ & \wedge \quad Ts_{min} \leq s(i,j) \leq Ts_{max} \\ & \wedge \quad Tv_{min} \leq v(i,j) \leq Tv_{max} \\ 0 \quad & \text{jinak.} \end{aligned} \tag{7.2}$$

Takovéto prahování lze uskutečnit v programu MATLAB pomocí okenní aplikace `colorThresholder`. V této aplikaci si zvolíme barevný model, na kterém budeme chtít prahování aplikovat a nastavíme dolní a horní prahy pro jednotlivé parametry popisující barevný model. Na obrázku 7.1 je vidět, že by se tato metoda dala použít k nalezení středu libely, pokud by se soustředná kružnice barevně lišila od pozadí obrazu jako zde. Segmentovaný binární obraz by se následně zpracoval funkcí `regionprops`, která měří vlastnosti oblastí obrazu a tím bychom získali střed libely. Problém však nastává při detekci bubliny, která s pozadím obrazu téměř splývá a není tedy možné nalézt jednoznačný práh, proto je metoda k jejímu nalezení nevhodná. Metod prahování je však daleko více, jejich popis nalezneme v těchto článcích, kterými jsem se nechal inspirovat i při psaní této kapitoly [27][21].

(a) **(b)**

Obrázek 7.1: Vzorový snímek krab. libely: a) originální snímek , b) segmentovaná soustředná kružnice.

7.2 Detekce hran

Mnohem vhodnější je tedy v tomto případě využít segmentaci na základě detekce hran. „*Výsledky neurofyziologického a psychofyzického výzkumu ukazují, že pro vnímání člověka jsou důležitá místa v obraze, kde se náhle mění hodnota jasu*“ [21]. Tedy pixely, které tato místa reprezentují, jsou považovány za hrany. Jednou z cest nalezení hran v digitálním obraze jsou metody založené na první derivaci obrazové funkce, někdy též označované jako gradientní metody. Princip těchto metod spočívá v tom, že v místech, kde se nachází hrana, dochází k největší změně intenzity, zatímco v homogenních oblastech je změna, a tedy i první derivace je rovna nule. Gradient $\nabla f(i,j)$ je vektor vyjadřující směr největšího růstu obrazové funkce, a je

tedy kolmý k směru hrany. Velikost gradientu $|\nabla f(i,j)|$ a úhlu natočení ψ vůči ose x lze vypočítat z parciálních derivací obrazové funkce podle x a y pomocí vzorců (7.3) a (7.4) [21] [27].

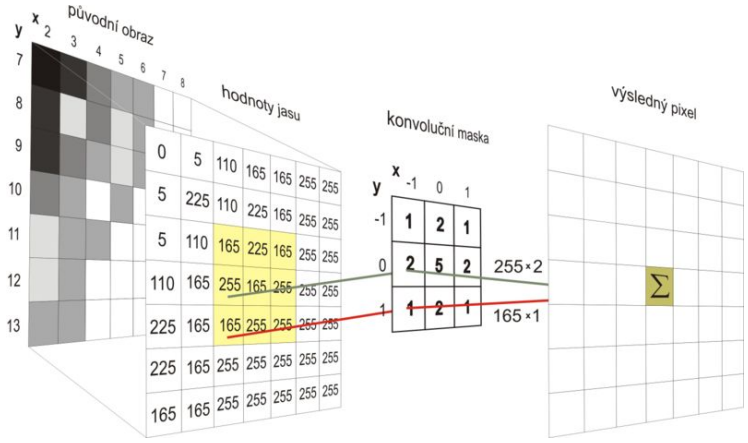
$$|\nabla f(i,j)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{7.3}$$

$$\psi = \arctan\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \tag{7.4}$$

Problém však spočívá v tom, že původně spojitá obrazová funkce je diskretizována v průběhu digitalizace obrazu (vzorkování, kvantování) a výpočet derivací pro diskrétní funkci je náročný. Proto je tedy potřeba parciální derivace aproximovat vhodným výpočtem diferencí [21]. K tomu se často využívají tzv. *hranové detektory*. Předtím, než si představíme hranové detektory, je nutné objasnit princip *konvoluce*.

Konvoluce

Konvoluce je matematický proces úpravy signálů. V případě digitálního obrazu se konvoluce využívá například k vyhlazení obrazu, zaostření obrazu nebo detekci hran. Pro výpočet konvoluce potřebujeme vstupní obraz, v kterém budeme prohledávat malé okolí každého pixelu a na základě toho přepočítávat jeho původní hodnotu. Příspěvek jednotlivých pixelů okolí je vážen hodnotami tzv. *konvoluční masky*. Princip konvoluce obrazu je přehledně ukázán na obrázku 7.2 [16].



Obrázek 7.2: Princip konvoluce - zdroj [16]

Hranové detektory

Hranové detektory se užívají k nalezení hran v digitálním obraze. Jedná se v podstatě o vhodně zvolené hodnoty v konvoluční maskách, které pak následně zvýrazní pixely, které odpovídají hraně. Hranový detektor se může skládat i z více konvolučních masek, přičemž každá vyhledává hrany v jiném směru. Příkladem může být *Prewittův* hranový detektor, který k aproximaci derivací používá dvě konvoluční masky o velikosti 3×3 , jednu pro odhad horizontálních změn G_x a druhou pro vertikální změny G_y. Definujeme-li I jako zdrojový obraz v šedé škále, pak tyto aproximace derivací získáme takto (7.5) následným dosazením do rovnic (7.3) a

(7.4) získáme následující vztahy (7.6) a (7.7)[21]. Výstupní obraz detekovaných hran je pak následně nutné segmentovat prahováním pro vyznačení pouze významných hran.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{7.5}$$

$$|\nabla f(i,j)| = \sqrt{G_x^2 + G_y^2} \tag{7.6}$$

$$\psi = \arctan\left(\frac{G_y}{G_x}\right) \tag{7.7}$$

Hranových detektorů založených na první derivaci existuje celá řada, vzájemně se liší počtem konvolučních masek a hodnotami v nich [27]. Několik příkladů:

- Prewittův detektor
- Sobelův detektor
- Kirschův
- Robinsonův

Matlab - hranová detekce

Pro vyhledání hran v obraze je v programu MATLAB připravena funkce `edge`, v které jsou naprogramovány výše zmíněné hranové detektory založené na první derivaci, ale i jiné metody jako například metody hledající průchod nulou druhé derivace popsané zde [21] [27]. Nejlepších výsledků však bylo dosaženo při použití Prewittova detektoru. Povinnými vstupními parametry této funkce jsou typ hranového detektoru a obraz v odstínech šedi, je tedy nutné původní barevný obraz nejprve do tohoto modelu převést. K tomu lze využít funkce `rgb2gray()`, která k převodu z barevného modelu RGB do odstínů šedi používá tuto rovnici (6.1). Nepovinnými parametry můžeme zvolit prahovou hodnotu nebo určit směr hledaných hran. Po zapsání příkazu `hrany=edge(Img,'prewitt')` se do matice `hrany` uloží binární obraz detekovaných hran (viz obr. 7.3).

(a) **(b)**

Obrázek 7.3: Vzorový snímek krab. libely: a) originální snímek , b) detekované hrany *prewittovým* operátorem.

8 Detekce - Houghova transformace

Poté, co jsme pomocí Prewittova hranového detektoru určili pixely, které považujeme za hrany a jsou v binárním obraze reprezentovány hodnotou 1, je nutné detekovat hrany reprezentující námi hledané kružnice. Jedna z metod řešící tento problém se nazývá *Houghova transformace* (HT). Autorem HT je Paul Van Campen Hough, který tuto metodu navrhl v roce 1959 pro detekci přímek v obraze při zkoumání drah elementárních částic na snímcích z bublinkové komory. Princip HT spočívá v tom, že body binárního obrazu (po hranové detekci) jsou transformovány do parametrického prostoru, kde hlasují o příslušnosti k dané přímce, která se v tomto prostoru zobrazuje jako bod. Tuto metodu později zdokonalili O. Duda a Peter E. Hart, kteří vytvořili *zevšeobecněnou Houghovu transformaci*, díky níž lze detekovat i ostatní parametricky popsané prvky jako kružnice, elipsy atd. [24].

8.1 Detekce přímek

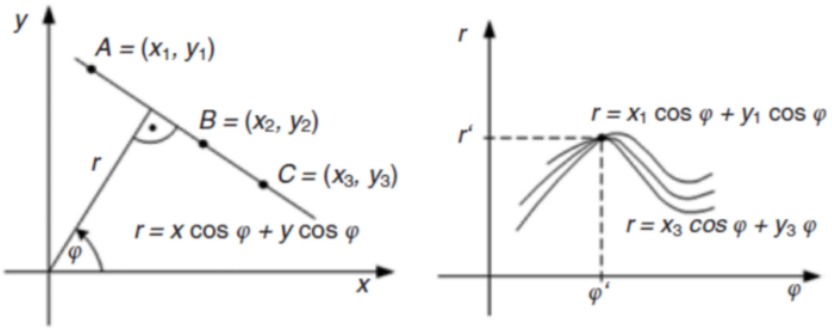
Jak je popsáno výše, hledaný prvek obrazu je nutné popsat parametrickou rovnicí. Jednou z možností parametrizace rovnice přímky je tzv. *směrnicový tvar přímky* popsaný rovnicí

$$y = kx + q, \tag{8.1}$$

kde k označuje směrnici přímky (tangenta úhlu sevřeného danou přímkou a kladným směrem osy x) a q je posun přímky po ose y. Tento tvar rovnice byl využit i při původním odvození metody. Problém však nastává při výskytu vertikálních přímek, kdy směrnice přímky nabývá nekonečné hodnoty. Tento problém vyřešili Duda a Hart použitím tzv. *normálového tvaru přímky*

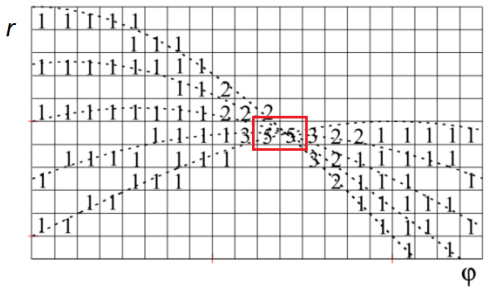
$$r = x \cos\varphi + y \cos\varphi, \tag{8.2}$$

přičemž r označuje vzdálenost bodu (x, y) od počátku souřadného systému a φ je úhel mezi normálou přímky a osou x. Definujeme-li bod $A(x_1, y_1)$, kterým prochází přímka, výsledná rovnice bude vypadat takto $r = x_1 \cos\varphi + y_1 \cos\varphi$ (viz obr. 8.1) [26][24][22].



Obrázek 8.1: Parametrický prostor - kartézský souř. prostor (vlevo), parametrický prostor (vpravo) - zdroj [26]

V parametrickém prostoru se každá přímka procházející bodem A zobrazí jako jediný bod. Svazek všech přímek procházejících bodem A bude reprezentován v parametrickém prostoru jako spojitá křivka (sinusoida) [26]. Budeme-li tedy znát alespoň dva body přímky, které promítneme do parametrického prostoru, můžeme neznámé r a φ určit jako průsečík těchto křivek. Pro detekci přímek v obraze je tedy nutné vytvořit tzv. *akumulační matici* H, která nám bude představovat dvourozměrný parametrický prostor. Poté bude pro všechny pixely vstupního binárního obrazu, které charakterizují hrany, vypočtena hodnota parametru r pomocí rovnice (8.4). Za parametr φ jsou následně dosazovány hodnoty od 0 do π s určitým krokem. Průchodem jednotlivých pixelů se inkrementují hodnoty parametrů (r, φ) v akumulátoru. Body ležící na jedné přímce, inkrementují hodnoty právě na pozicích (r, φ), odpovídající parametrickému popisu přímky a pro následnou detekci přímek v digitálním stačí nalézt maxima v akumulační matici (viz obr.8.2) [26][22].



Obrázek 8.2: Ukázka akumulační matice - zdroj [26]

8.2 Detekce kružnic

Stejně jako u detekce linií si musíme nejprve vyjádřit parametrickou rovnici kružnice. Jednoznačné určení kružnice je pomocí tří parametrů r - poloměru a (a, b), což jsou souřadnice středu kružnice. Každý bod kružnice o poloměru r a středu v bodě (a, b) lze tedy dle parametrického vyjádření popsat rovnicemi

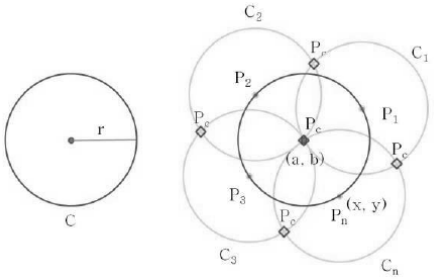
$$\begin{aligned} x &= a + r\cos\varphi, \\ y &= b + r\sin\varphi. \end{aligned} \tag{8.3}$$

Budeme-li tedy detekovat v binárním obraze kružnice o známém poloměru r můžeme polohu středu inkrementovat do dvourozměrné akumulační matice s parametry a a b, které vypočteme následovně:

$$\begin{aligned} a &= x - r\cos\varphi, \\ b &= y - r\sin\varphi. \end{aligned} \tag{8.4}$$

Parametr φ označuje interval hodnot od 0 do 2π. Kolem každého bodu binárního obrazu bude v akumulační matici zakreslena kružnice o poloměru r. Budeme-li znát přesný poloměr detekované kružnice, všechny takto zakreslené kružnice se protnou v jediném bodě, který charakterizuje střed původní kružnice (viz obr 8.3). Následným

nalezením maxim v této matici detekujeme středy kružnic o poloměru r. Problém však nastane, neznáme-li přesný poloměr detekované kružnice. V tom případě bude nutné použít třírozměrný parametrický prostor s parametry a, b a r. Důležitá je volba intervalu prohledávaných poloměrů r, zvolíme-li velké rozmezí poloměrů, výpočetní čas plnění akumulátoru a hledání maxim v něm prudce vzroste. Kdybychom naopak zvolili malé rozmezí může se stát, že kružnice nebudou detekovány, jelikož jejich správný poloměr nebude spadat do zvoleného interval[22][23].



Obrázek 8.3: Ukázka hledání středu detekované kružnice - zdroj (`https://www.researchgate.net/figure/277586131_fig2_Fig-4-Hough-circle-transform-example`)

8.3 Detekce kružnic - MATLAB

Pro detekci kružnic v programu MATLAB byly požity tyto dvě funkce `circle_hough` a `circle_houghpeaks`, které nejsou součástí knihovny Image Procesing Toolbox a byly staženy na oficiálních stránkách MathWorks [20].

Funkce circle_hough

Tato funkce složí k vytvoření 3D akumulační matice popsané výše. Vstupními parametry funkce jsou binární obraz hran, rozmezí prohledávaných poloměrů a volba metody inkrementace. Pokud zvolíme volbu *'Normalise'*, jsou hodnoty inkrementací v jednotlivých úrovních poloměru normovány, jelikož kružnice s větším poloměrem mají více bodů, které je zastupují a byly by tak zvýhodněny při hledání maxim. Volba *'Same'* ponechá původní inkrementace. Příklad zápisu funkce:

```
H = circle_hough(Img_edges,rmin:rmax,'same');
```

Funkce circle_houghpeaks

Poté, co jsme vytvořili akumulační matici, musíme následně nalézt její maxima a tím detekovat případné kružnice. A právě k tomuto účelu slouží tato funkce, jejíž výsledkem jsou detekované poloměry a souřadnice středů kružnic. Vstupem je tedy akumulační matice vypočtená funkcí `circle_hough` a nastavení těchto tří parametrů:

- *Npeaks* - je to počet hledaných maxim.
- *Nhoodxy* - minimální prostorové oddělení mezi vrcholy.

- *Nhoodr* - oddělení poloměru mezi vrcholy.

Příklad zápisu funkce:

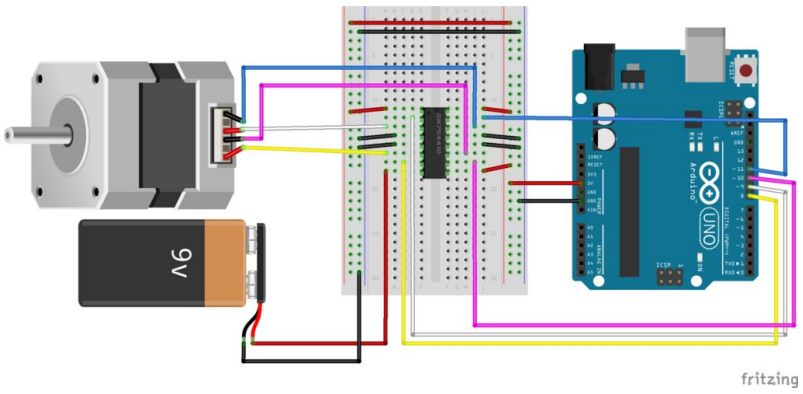
```
vrcholy = circle_houghpeaks(H,rmin:rmax, 'nhoodxy', 15, ...
'nhoodr', 21, 'npeaks', 1);
```

Část III
Program AHP

V předchozím textu jsme si představili potřebný hardware, software a metody pro detekci prvků v digitálním obraze. Díky tomu jsme získali všechny potřebné informace, pro vytvoření systému AHP. V poslední části této práce sloučíme všechny tyto poznatky a objasníme tvorbu samourovnávací desky a její řídící program v podobě okenní aplikace.

9 Připojení a ovládání krokových motorů

Jak již bylo řečeno krokové motory nemohou být použity bez řídící jednotky, která ovládá spínání jednotlivých cívek statoru ve správném pořadí a po určitých časových intervalech. Nyní si ukážeme potřebné zapojení elektronických součástek a krokových motorů k vývojové desce Arduino Uno a tím z ní vytvoříme řídící jednotku pro dva krokové motory. Příklad zapojení jednoho krokového motoru k desce Arduino pomocí nepájivého pole je vyobrazen na obrázku 9.1, který byl nalezen i s návodem na této webové stránce [5]. Druhý krokový motor byl zapojen stejným způsobem, jenom pro vstup do desky Arduino bylo využito jiné čtveřice volných digitálních pinů. Pozorný čtenář si všimne, že je k nepájivému poli připojena i 9V baterie. Je to z toho důvodu, že Arduino desky pracují s napětím o velikosti 5V a to k chodu krokových motoru nestačí, je tedy nutné připojit další zdroj energie. Ve finálním projektu je zdroj dodáván pomocí trafa, poskytujícího napětí 20V.



Obrázek 9.1: Ukázka zapojení motoru k Arduino desce

Potřebné elektronické součástky:

- deska Arduino UNO,
- 2 × krokový motor (viz kapitola *Použitý krokový motor*),
- 2 × H-můstek (L293D),
- zdroj napětí,
- deska plošného spoje.

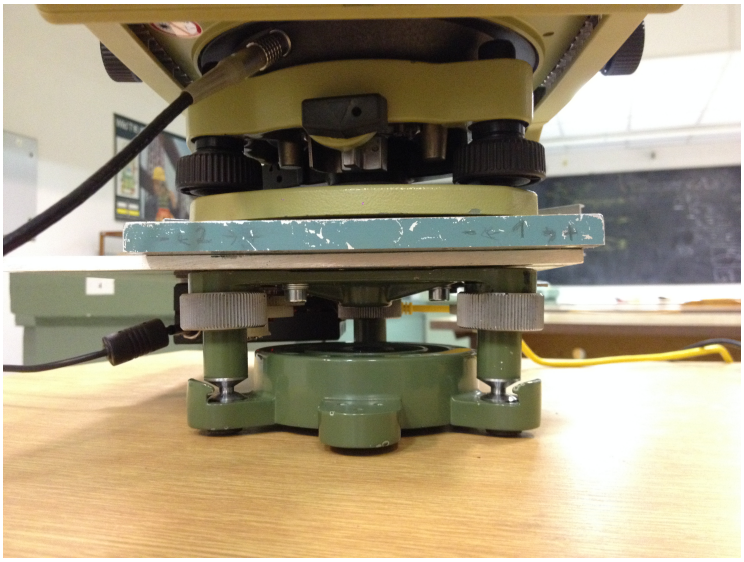
K chodu motorů je potřebné nejen správné zapojení všech částí, ale také nahrání kódu do Arduino desky. Deska je pak následně připojena k počítači USB kabelem, přes který probíhá sériová komunikace. Zde je ukázán nahraný kód a jeho jednotlivé části jsou vysvětleny v komentářích.

```
#include <Stepper.h>
#include <math.h>

const int stepsPerRevolution = 48;
// %Pocet otocek potrebnych pro otoceni o 360 stupnu :
// %nas motor ma krok chodu o velikosti 7.5 stupnu--> 360/7.5= 48
// %Vytvoreni tridy Stepper (zadani poctu kroku na otocku a ... vystupni PINy).
Stepper myStepper1(stepsPerRevolution, 10,11,12,13);
Stepper myStepper2(stepsPerRevolution, 2,3,4,5);
// %Deklarace funkce a jednotlivych promennych.
void pohyb_motoru(int prvni, int druhy);
int poc_otocek1; //
int poc_otocek2; //
int pom;

//%Prikazy v teto funkci se provedou pouze pri zapnuti desky.
void setup() {
  // %nastaveni rychlosti otaceni jednotlivych motoru - pocet ... otocek za minutu
  myStepper1.setSpeed(240);
  myStepper2.setSpeed(240);
  //% vytvoreni seriove komunikace
  Serial.begin(9600);
}

//%Prikazy v teto funkci se budou provadet v nekonecne smycce.
void loop() {
//% pokud je poslan prikaz - proved
  if(Serial.available()>0)
  {
    poc_otocek1=Serial.parseInt();
    poc_otocek2=Serial.parseInt();

    pohyb_motoru(poc_otocek1,poc_otocek2);

    // % Po ukonceni otaceni motoru je zpet poslano pismeno "O"
    // % (otoceno), diky tomu matlab zjisti, ze muze pokracovat.
    Serial.println("O");
  }

}

//% Definice funkce
//% Funkce byla vytvorena proto, aby byly motory spousteny soucasne.
void pohyb_motoru(int prvni, int druhy){

```

```
46    int posunuto1=0;
47    int posunuto2=0;
48    int pom=max(abs(prvni),abs(druhy));
49
50    int p1=1;
51    int p2=1;
52
53    if (prvni<0) {p1=-1;}
54    if (druhy<0) {p2=-1;}
55
56    for (int x=0; x<pom; x++){
57    if (posunuto1<abs(prvni)){myStepper1.step(p1);posunuto1++;}
58    if (posunuto2<abs(druhy)){myStepper2.step(p2);posunuto2++;}
59    }
60  }
```

10 Srovnávací deska

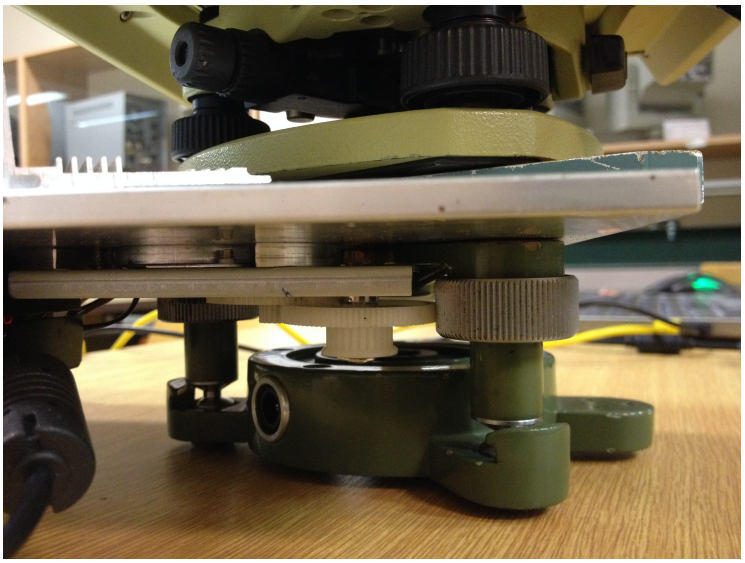
Jako podstavec této desky je použita trojnožka geodetického cílového terče MOM Budapest K202, která je překlopena vzhůru nohama. Nad trojnožku je přišroubována hliníková deska, pomocí níž je vytvořena rovina, na které je umístěna snímaná krabicová libela. Na hliníkové desce je ještě připevněn kovový prvek s upevňovacím šroubem pro přichycení přístroje (viz obr. 10.1).



Obrázek 10.1: Ukázka srovnávací desky

Ve spodní části hliníkové desky jsou připevněny dva krokové motory a jejich řídící jednotka (Arduino Uno). Krokové motory byly připojeny k stavěcím šroubům i s převodovým mechanismem, jehož ozubení zapadalo do ozubení stavěcích šroubů (viz obr. 10.2). K urovnávání se využívá pouze dvou stavěcích šroubů, a to kvůli tomu, aby nedocházelo k zdvihu, ale pouze k urovnání přístroje. Jelikož je trojnožka umístěna vzhůru nohama, nelze již aplikovat tzv. „*pravidlo levého palce*“, tedy že pohyb bubliny při urovnání bude stejný, jako směr pohybu palce levé ruky. V tomto

případě se bude jednat o „*pravidlo pravého palce*“. Je však velmi důležité, abychom rukou neotáčeli stavěcími šrouby, respektive krokovými motory zapojenými k řídící jednotce. Při otáčení by se mohl naindukovat náboj, který by řídící jednotku zničil.



Obrázek 10.2: Upevnění krokových motorů k stavěcím šroubům

10.1 Použitá krabicová libela

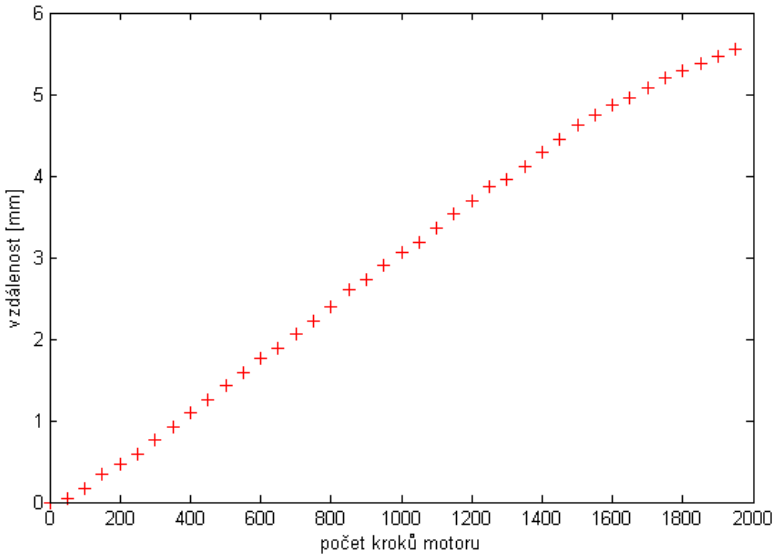
K určování náklonů byla využita krabicová libela s kovovým tělem, vyrobená firmou KINEX Measuring (viz obr. 10.3). Výrobce udává citlivost libely v rozmezí 5’-60’/2 mm, bohužel však nejsou blíže specifikovány oblasti pro jednotlivé hodnoty citlivostí [12].

Obrázek 10.3: Použitá krabicová libela - zdroj [12]

Pro získání těchto informací byl použit tento postup. Libela byla společně se srovnávací deskou zhorizontována a následně nakláněna v příčném směru. Naklánění probíhalo postupně vždy po stejném počtu kroků (konkrétně první motor o 50 kroků a druhý motor o -50 kroků). Po každé realizaci náklonu byla určena vzdálenost od středu libely. Postup byl opakován, dokud nebyl jeden ze stavěcích šroubů, vytočen do své krajní polohy. Data získaná tímto testem byla vynesena do grafu (viz obr. 10.4). Na grafu je vidět, že závislost posunu bubliny na počtu kroků má lineární charakter, což značí, že citlivost libely v této proměřované

oblasti by měla být konstantní.

Z těchto dat pak byla stanovena hodnota citlivosti libely 8'. Skutečnost, že citlivost libely byla stanovena pouze přibližně, však nebude mít na vliv na konečnou přesnost urovnání, protože citlivost libely potřebujeme znát pouze z důvodu, abychom mohli určit počet kroků motoru pro realizaci náklonu. To znamená, že při použití pouze přibližné citlivosti se nemusí urovnání podařit hned při první realizaci náklonů, ale dojde k postupnému iterování.



Obrázek 10.4: Závislost vzdálenosti bubliny od středu libely na počtu kroků motoru

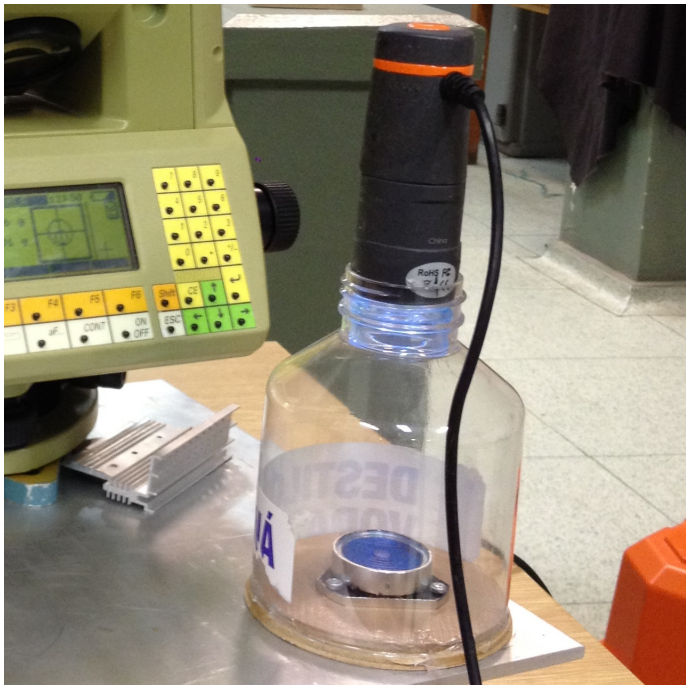
10.2 Upevnění mikroskopu

Abychom mohli detekovat kružnice na digitálním snímku, je nutné, aby osa digitálního mikroskopu byla kolmá k ose krabicové libely. K tomu byl vytvořen stojan z pevné plastikové lahve, který tuto podmínku splňuje. Digitální kamera je zasunuta do hrdla láhve, které zajišťuje dostatečnou oporu a zároveň chrání libelu před znečištěním z okolí a umožňuje její osvětlení přirozeným světlem z venku (viz obr. 10.5).

10.3 Orientace mikroskopu

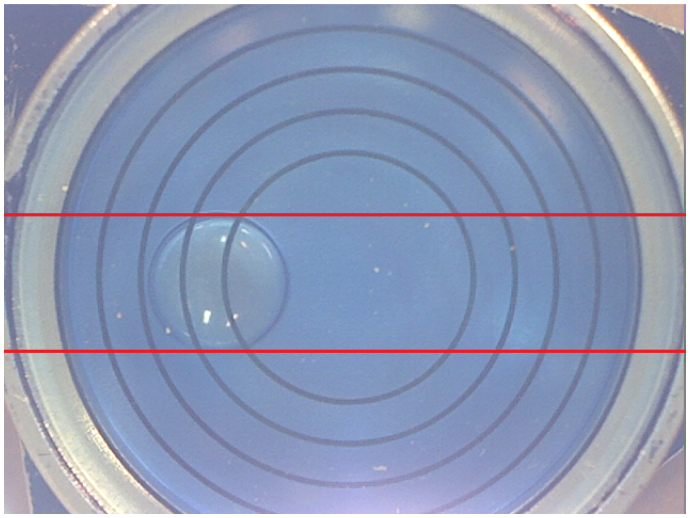
Pro správné detekování náklonů a následné urovnání je nezbytné, aby byla krabicová libela snímána ze správného směru. A to z takového, že při podélném naklánění desky bude střed detekované bubliny měnit pouze y-ovou souřadnici v digitálním obraze a naopak při příčném naklánění x-ovou souřadnici.

Nalezení tohoto směru proběhlo tak, že byla deska přibližně urovnána a následně byl do Arduina nahrán kód, který nakláněl desku pouze v příčném směru z jedné



Obrázek 10.5: Upevnění mikroskopu v láhvi

krajní polohy do druhé. V programu MATLAB bylo pak spuštěno okno, které zobrazovalo jednotlivé snímky libely pořízené mikroskopem. Snímky byly doplněny ještě o dvě horizontální přímky, které vymezovaly pozici, ve které by se měla nacházet bublina libely (viz obr. 10.6). Pokud při celém průběhu příčného náklonu bublina „nevyběhne“ mimo vykreslené přímky, znamená to, že jsme nalezli správný směr. Je jasné, že směry, které splňují tuto podmínku jsou dva, vzájemně posunuté o 180°. Musíme tedy vybrat ten, který zobrazuje snímek stejně, jako při pohledu od stavěcích šroubů. Nalezený směr byl následně označen na hrdlo láhve a digitální mikroskop.



Obrázek 10.6: Kontrola správného natočení mikroskopu

10.4 Určení zdvihu stavěcích šroubů

Poté co byly všechny části srovnávací desky zkompletovány, bylo nutné určit vztah mezi počtem otáček krokových motorů a jimi způsobeným náklonem. Náklony desky byly rozděleny do dvou složek (podélný a příčný). Budeme-li pohybovat oběma stavěcími šrouby ve stejném směru a o stejný počet otáček, realizovaný náklon bude pouze v podélném směru. Naopak při pohybu šroubů v opačných směrech budeme naklánět desku v příčném směru.

Vztah mezi počtem otáček a náklonem byl určen tak, že byla k desce připojena totální stanice Leica TCA 2003 a náklony realizované určitým počtem kroků byly odečítány pomocí integrovaného elektronického náklonoměru. Nejprve však musela být totální stanice natočena do správného směru, tedy aby podélné náklony desky detekovala totální stanice pouze jako podélné a naopak. K tomuto nastavení byl využit stejný postup, jako při nastavení orientace mikroskopu. Srovnávací deska byla současně s totální zhorizontována a následně byly odečítány náklony pro jednotlivé počty kroků (viz tabulka 10.1).

PODÉLNÝ SMĚR			PŘÍČNÝ SMĚR		
počet kroků		detekovaný úhel [gon]	počet kroků		detekovaný úhel [gon]
1. motor	2. motor		1. motor	2. motor	
10	10	0,005	-10	10	0,006
20	20	0,010	-20	20	0,016
50	50	0,024	-50	50	0,042
100	100	0,048	-100	100	0,083
200	200	0,099	-200	200	0,170
300	300	0,153	-300	300	0,259
400	400	0,201	-400	400	0,335
500	500	0,260	-500	500	0,443
-10	-10	-0,006	10	-10	-0,006
-20	-20	-0,010	20	-20	-0,014
-50	-50	-0,023	50	-50	-0,041
-100	-100	-0,045	100	-100	-0,088
-200	-200	-0,097	200	-200	-0,176
-300	-300	-0,140	300	-300	-0,256
-400	-400	-0,195	400	-400	-0,325
-500	-500	-0,253	500	-500	-0,457

Tabulka 10.1: Hodnoty detekovaných náklonů

Poté byly z těchto hodnot určeny parametry, které určují stejný počet kroků obou motorů pro náklon 1 *mgon* v podélném respektive příčném směru. K tomu byl využit tento vzorec:

$$x = \left|\frac{\text{počet kroků jednoho motoru}}{\text{detekovaný náklon [mgon]}}\right| \tag{10.1}$$

Prostým aritmetickým průměrem pak byla určena hodnota **2,022** $^{kroků}/_{mgon}$ pro podélný náklon a **1,253** $^{kroků}/_{mgon}$ pro příčný náklon. Znamená to tedy, že pokud budeme chtít naklonit desku v podélném směru o 1 $mgon$, musíme oběma krokovými motory otočit o 2,022 kroku a analogicky v příčném směru. Krokovými motory však lze otočit pouze o celočíselný počet kroků, proto výslednou hodnotu musíme vždy zaokrouhlit. Směr náklonu je pak určen směrem otáčení krokových motorů.

10.5 Určení náklonů

Detekcí kružnic na snímaném obraze získáme souřadnice jejich středů. Pokud se souřadnice neshodují, znamená to, že snímaná libela není urovnána. Vypočítáme tedy souřadnicové rozdíly dx pro příčný směr a dy a pro podélný směr. Následně je nutné určit skutečnou velikost pixelu vpx, toho docílíme tak, že vydělíme průměr soustředného kroužku změřeného na libele a průměr odečtený na obraze. Následným vynásobením souřadnicového rozdílu, velikosti pixelu a citlivosti libely C určíme výsledné hodnoty náklonů. Následující rovnice popisují tento postup.

$$\begin{aligned} dx &= x_{lib} - x_{bub} \\ dy &= y_{bub} - y_{lib} \end{aligned}$$

$$vpx = \frac{průměr\ [\mathrm{mm}]}{průměr\ [\mathrm{px}]}$$

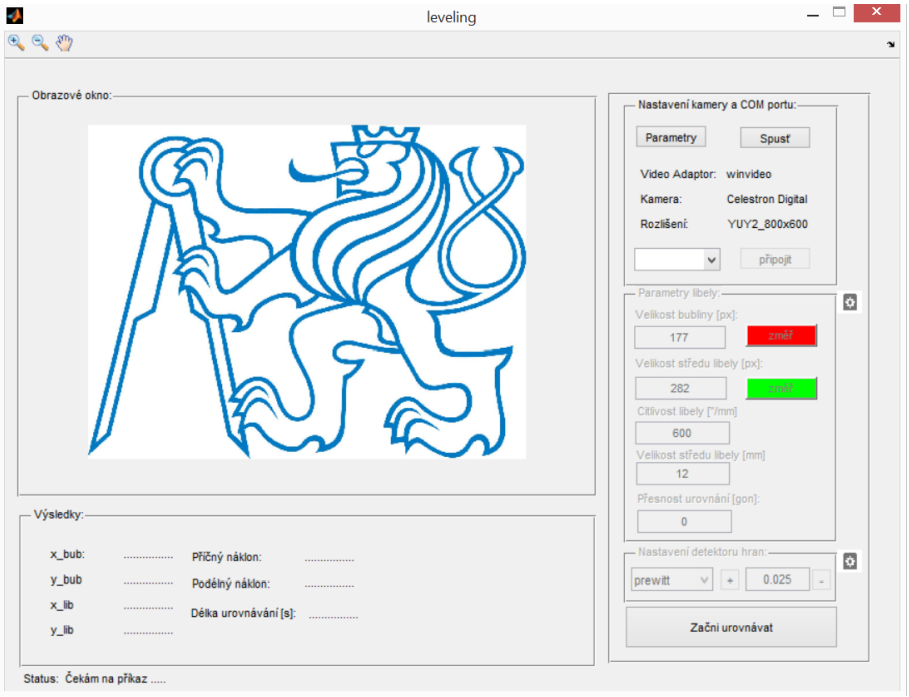
$$\begin{aligned} \varphi &= dx \cdot vpx \cdot C \\ \lambda &= dy \cdot vpx \cdot C \end{aligned}$$

11 Řídící program

Pro kontrolu celého programu byla vytvořena okenní aplikace ve vývojovém prostředí GUIDE, které poskytuje nástroje pro navržení vlastního uživatelského rozhraní (zkratkou v anglickém jazyce „GUI“). Vytvořené GUI obsahuje interaktivní tlačítka, roletová menu, boxy pro vkládání hodnot, popisy a další prvky. Díky tomu může i uživatel neznalý programování v MATLABu ovládat všechny vytvořené funkce. Při založení uživatelského rozhraní aplikace GUIDE automaticky vygeneruje dva soubory, v tomto případě ***leveling.m*** a ***leveling.fig***. Soubor *leveling.fig* obsahuje informace o umístění jednotlivých grafických prvků programu (tlačítka, texty, roletová menu atd.). Do druhého souboru *leveling.m* pak zapisujeme kód, který definuje chování programu při interakci s prvky uživatelského rozhraní [13].

Po spuštění tohoto programu a následném úspěšném urovnání se hodnoty vstupních parametrů (název kamery, průměr libely, typ hranového detektoru atd.) uloží do textového souboru „*settings.txt*“. Při následném spuštění aplikace se tyto parametry automaticky vyplní podle předchozího nastavení a můžeme rovnou přistoupit

k urovnání. Okno je pro přehlednost děleno do jednotlivých oddílů, které zlepšují orientaci v hlavním okně (viz obr. 11.1).



Obrázek 11.1: Grafické rozhraní programu

Obrazové okno

Toto okno slouží k zobrazení živého přenosu z připojené kamery, měření průměrů detekovaných kružnic a také zobrazení jejich detekce, což je důležité pro vizuální kontrolu správné detekce uživatelem. Pro manipulaci s obrazovými daty v tomto okně jsou použita tři funkční tlačítka. *Zoom In* a *Zoom Out* pro přiblížení a oddálení obrazu a tlačítko *Pan*, které umožňuje posun okna po přiblíženém obraze.

Nastavení kamery a COM portu

Stiskem tlačítka *Parametry* obsaženém v této části se zobrazí menu, v kterém se jsou uvedena veškerá dostupná zařízení včetně podporovaných obrazových formátů. Jejich potvrzením pak danou kameru připojíme. Zvolené parametry se následně zobrazí v popiscích pod tímto tlačítkem. Pro spuštění živého náhledu připojené kamery využijeme tlačítka *Spusť*.

Při spuštění této okenní aplikace vyhledá MATLAB všechny dostupné komunikační porty a jejich názvy vloží do roletového menu. V něm si pak zvolíme sériový port charakterizující komunikaci mezi počítačem a deskou Arduino Uno. Pokud by v roletovém menu bylo více dostupných COM portů, můžeme ve správci zařízení najít ten správný. Po zvolení příslušného portu spustíme komunikaci tlačítkem *Připojit*. Při úspěšném připojení se popis tlačítka změní na *Odpojit*. Před ukončením programu je nutné tímto tlačítkem sériovou komunikaci přerušit.

Parametry libely

Prvky v tomto oddílu jsou ve výchozím nastavení needitovatelné. Pro jejich zpřístupnění je nutné stisknout tlačítko nastavení, které se nachází vpravo od oddílu a je reprezentováno obrázkem ozubeného kola. Při jeho stisknutí se v obrazovém okně zobrazí snímek libely, pořízený připojenou kamerou.

Na tomto snímku je nutné následně změřit průměr hledaných kružnic, díky čemuž upravíme interval prohledávaných poloměrů při detekci kružnic s HT. K určení těchto průměrů se využívají tlačítka *Změř*, které po stisknutí vytvoří v obraze linii s popiskem její délky. Koncové body této linie se dají v obraze posouvat pomocí myši. Potvrzením tlačítka se hodnota délky linie (reprezentující průměr kružnice) uloží.

Dále je nezbytné vyplnit citlivost libely, průměr soustředného kroužku libely a přesnost urovnání. Jednotky, v kterých se hodnoty zadávají, jsou uvedeny v popiscích. Přesnost urovnání je mezní hodnota náklonu, tedy když detekovaný příčný i podélný náklon budou menší než tato hodnota, systém urovnávání ukončí. Kdybychom tedy vložili hodnotu 0, urovnávání by probíhalo, dokud by souřadnice středů kružnic nebyly totožné.

Nastavení detektoru hran

Pro zpřístupnění tohoto oddílu musíme stejně jako u nastavení parametrů libely nejdříve kliknout na tlačítko nastavení. Díky tomu se v obrazovém okně zobrazí binární obraz (po hranové detekci s parametry předchozího nastavení) snímku pořízeného kamerou. Následnou volbou hranového detektoru a prahové hodnoty můžeme ovlivnit výsledný binární obraz detekovaných hran, který pak dále zpracováváme pomocí HT. Nejlepších výsledků při detekci hran bylo dosaženo při použití **Prewittova** hranového detektoru s prahovou hodnotou **0.02**. Pokud prahovou hodnotu v tomto oddíle nevyplníme, jednotlivé hranové detektory si ji zvolí automaticky.

Výsledky

V této části hlavního okna se zobrazují průběžné výsledky urovnávání (souřadnice středů kružnic, vypočtené náklony) a po konečném urovnání i celková doba trvání.

12 Použití programu

Před spuštěním aplikace v programu MATLAB, je nutné připojit k PC digitální mikroskop a desku Arduino Uno pomocí USB kabelů. Následně připojíme napětí ke krokovým motorům, kdybychom tak neučinili, program by tuto skutečnost nezjistil a sériová komunikace s deskou by probíhala i tak, protože je napájena USB kabelem. Ve výsledku by tedy nedocházelo k otáčení krokových motorů.

Poté může být naše aplikace spuštěna. Pokud aplikaci spouštíme poprvé, musíme nastavit všechny vstupní parametry, jinak se zobrazí hodnoty z předchozího

spuštění. Následně musíme vybrat komunikační port řídící jednotky a spustit komunikaci. Nyní je již program připraven k urovnávání desky, které se zahájí tlačítkem *Začni urovnávat*.

13 Testování programu

Pro získání představy o přesnosti urovnání srovnávací desky do vodorovné polohy byla v programu nastavena přesnost urovnání 0 *gon*, to znamená, že urovnávání probíhalo do té doby ,dokud souřadnice středů kružnic nebyly totožné. Poté co byla deska takto urovnána, byly odečteny zbytkové náklony pomocí totální stanice Leica TCA 2003. Výrobce této totální stanice bohužel neuvádí přesnost elektronického náklonoměru. Přesto se dá předpokládat, že bude několikanásobně lepší, než-li přesnost krabicové libely. Z toho důvodu jsou tedy tyto hodnoty považovány za bezchybné. Zjištěné hodnoty zbytkových náklonů jsou uvedeny v tabulce 13.1 i s hodnotami původních náklonů a celkové doby urovnání.

Původní náklony		Zbytkové náklony		délka urovnání
podélný [gon]	příčný [gon]	podélný [gon]	příčný [gon]	[s]
-0,045	0,371	0,004	0,009	47
0,317	0,045	0,001	0,003	59
0,250	0,455	0,012	0,000	41
-0,236	0,421	0,012	0,002	61
0,063	-0,499	0,014	0,007	33
-0,236	0,020	0,001	0,007	34
0,374	-0,002	-0,002	-0,004	53
0,292	-0,507	0,002	-0,004	85
-0,150	-0,236	0,000	0,005	33
0,235	0,431	0,013	0,002	29

Tabulka 13.1: Testování přesnosti urovnání

Průměrná chyba urovnání tedy vychází **0,005** *gon*, což je velice uspokojivý výsledek při urovnávání s pomocí krabicové libely. Dalším důležitým parametrem je doba trvání urovnávání. Průměrná doba horizontace srovnávací desky byla při tomto testování 47,5 *s*. Hlavním důvodem takto vysoké hodnoty je to, že byla nastavená velká přesnost urovnávání. Při nastavení přesnosti urovnání 0.050 *gon*, se délka urovnávání pohybuje okolo 30 *s*.

Závěr

Cílem této bakalářské práce byl vývoj zařízení pro automatickou horizontaci. Tento úkol se skládal z několika částí. Nejdříve bylo nutné navrhnout podobu a vytvořit srovnávací desku, která reprezentuje horizontovanou rovinu. Základem této desky je upravená trojnožka z geodetického cílového terče a k ní jsou připevněny dva bipolární krokové motory, jejichž úkolem je otáčení stavěcích šroubů. K řízení těchto motorů je využito vývojové desky Arduino Uno, která komunikuje s počítačem přes sériový port.

Určení náklonů je uskutečněno snímáním krabicové libely digitálním USB mikroskopem a na pořízených snímcích jsou detekovány kružnice, které představují bublinu a soustředný kroužek libely. Pro detekci kružnic v digitálním obraze je použita Houghova transformace. Náklony desky jsou poté určeny jako součin souřadnicového rozdílu, velikosti pixelu a citlivosti libely.

Pro řízení celého systému byla v programu MATLAB vytvořena okenní aplikace, která komunikuje a řídí připojená zařízení. V této aplikaci musí uživatel vyplnit požadované vstupní parametry a stiskem jednoho tlačítka spustí proces urovnávání, které průměrně trvá 30 vteřin.

V této práci jsme se převážně zaměřili na celkovou technickou realizaci srovnávací desky a jejího řídícího programu, které dohromady tvoří systém AHP. Testování tohoto systému proběhlo z časových důvodů pouze v omezené míře. Přesto při určování zbytkových náklonů pomocí totální stanice bylo zjištěno, že průměrný zbytkový náklon vychází 0,005 gon. Kdybychom chtěli náklony získávat elektronickým náklonoměrem, cena snímače o přibližně stejné přesnosti by mohla dosáhnout i několik desítek tisíc korun. Celkově bych tedy výsledné zařízení a program hodnotil velice pozitivně, i když je samozřejmě ještě spousta možností jak ho vylepšit. Hlavním prvkem pro zlepšení by nejspíše byla doba urovnávání, která by se dala vhodnou optimalizací kódu a určitého předzpracování obrazu zrychlit. Systém bych tedy přirovnal k žáku prvního ročníku průmyslové školy, který již dokáže precizně zhorizontovat přístroj, ale k realizaci potřebuje dostatek času.

Reference

[1] *ARDUINO.CZ* [online]. [cit. 2017-05-20]. Dostupné z: https://arduino.cz

[2] *Blog o Arduino* [online]. [cit. 2017-05-20]. Dostupné z: http://blog.laskarduino.cz/

[3] *Arduino and arduino-compatible hardware* [online]. [cit. 2017-05-20]. Dostupné z: http://playground.arduino.cc/main/similarBoards

[4] *Automatizace*. In: Wikipedia: the free encyclopedia [online]. [cit. 2017-05-20]. Dostupné z: https://cs.wikipedia.org/wiki/Automatizace

[5] *Controlling a Stepper Motor With an Arduino* [online]. [cit. 2017-05-20]. Dostupné z: http://www.instructables.com/id/Controlling-a-Stepper-Motor-with-an-Arduino/

[6] *Deluxe handheld digital microscope*, Celestron, LLC. [online]. [cit. 2017-05-20]. Dostupné z: http://www.celestron.com/browse-shop/microscopes/digital-microscopes/deluxe-handheld-digital-microscope

[7] *History of matlab*, CLEVERISM. [online]. [cit. 2017-05-20]. Dostupné z: https://www.cleverism.com/skills-and-tools/matlab/

[8] *Image acquisition toolbox: User's guide*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: http://www.mathworks.com/help/pdf_doc/imaq/imaq_ug.pdf

[9] *Image processing toolbox: User's guide*,THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: http://se.mathworks.com/help/pdf_doc/images/images_tb.pdf

[10] *Krokový motor*. In: Wikipedia: the free encyclopedia. [online]. [cit. 2017-05-20]. Dostupné z: https://cs.wikipedia.org/wiki/Krokov%C3%BD_motor

[11] *Krokový motor*. Wiki SPŠ a VOŠ Písek. [online]. [cit. 2017-05-20]. Dostupné z: http://wiki.sps-pi.cz/index.php/Krokov%C3%BD_motor

[12] *Kruhová libela KINEX s upevňovacími otvory*, KINEX Measuring. [online]. [cit. 2017-05-20]. Dostupné z: https://shop.kinexmeasuring.com/cs/kruhova-libela-s-upevnovacimi-otvory-35mm-stribrny-elox-p9100003c151/?search=5022-03-035

[13] *Matlab® app building*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: http://www.mathworks.com/help/pdf_doc/matlab/buildgui.pdf

[14] *Matlab® primer*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: http://se.mathworks.com/help/pdf_doc/matlab/getstart.pdf

[15] *The matlab r2012b desktop*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: http://blogs.mathworks.com/loren/2012/09/12/the-matlab-r2012b-desktop-part-1-introduction-to-the-toolstrip/

[16] *Obrazové filtry*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18431

[17] *Seznámení s Arduinem*, ARDUINO.CZ. [online]. [cit. 2017-05-20]. Dostupné z: https://arduino.cz/seznameni-s-arduinem/

[18] *Support package installation*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: https://se.mathworks.com/help/matlab/matlab_external/support-package-installation.html

[19] *Support package installer help*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: https://se.mathworks.com/help/matlab/matlab_external/support-package-installer-help.html

[20] BONE, Pete. *Hough transform for circle detection*, THE MATHWORKS, Inc. [online]. [cit. 2017-05-20]. Dostupné z: https://se.mathworks.com/matlabcentral/fileexchange/9833-hough-transform-for-circle-detection

[21] HLAVÁČ, Václav; SEDLÁČEK, Miloš. *Zpracování signálů a obrazů*. České vysoké učení technické, 2000.

[22] HORÁK, Karel, et al. *Počítačové vidění*. Brno: Vysoké učení technické v Brně, 2008

[23] VLACH, Jaroslav. *Hledání úseček a kružnic s využitím Houghovy transformace při zpracování obrazu v LabView*. Automa, Únor, 2011, 42-44.

[24] KUBIČKOVÁ, Eliška Anna. *Houghova a Radonova transformace ve vyhledávání meteorů*. Pokroky matematiky, fyziky a astronomie, 2011, 56.2: 119-128.

[25] NEKVASIL, Vladimír. *Ovládání krokových motoru - didaktická pomůcka*. Bakalářská práce, České vysoké učení technické,2008.

[26] SOJKA, Eduard; GAURA, Jan; KRUMNIKL, Michal. *Matematické základy digitálního zpracování obrazu*. Ostrava, Plzeň: VŠB-TU Ostrava (Fakulta elektrotechniky a informatiky), Západočeská univerzita v Plzni, 2.

[27] STRAKA, Stanislav. *Segmentace obrazu*, Diplomová práce, Masarykova univerzita, 2009.

[28] SZELISKI, Richard. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[29] VODA, Z. a Tým HW Kitchen. *Průvodce světem Arduina*, 2015.

[30] ŽÁRA, Jiří, et al. Moderní počítačová grafika. Computer press, 1998.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

AHP - automatická horizontace přístroje
COM - hardwarové rozhraní
HT - houghova transformace
GUI - Graphical User Interface (Grafické uživatelské rozhraní)

Seznam použitých programů

- Matlab R2014a (MathWorks)
- Texmaker 4.3 (Pascal Brachet)
- Arduino IDE

Seznam příloh

A Elektronická příloha

CD obsahující soubory pro spuštění řídícího programu.