Jihočeská univerzita v Českých Budějovicích

přírodovědecká fakulta

Sledování včel na vstupu včelího úlu

Petr Novotný

bakalářská práce

Vedoucí práce: Ing. Miroslav Skrbek, Ph.D

2021

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Petr Novotný
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika

Katedra: Ústav aplikované informatiky

Školitel: Ing. Miroslav Skrbek, Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Sledování včel na vstupu včelího úlu

Cíle práce:

Seznamte se s inteligentním včelím úlem provozovaným Biologickým centrem AV ČR v Českých Budějovicích. Na úl nainstalujte kamery pro sledování pohybu včel na vstupu úlu. Navrhněte a implementujte programové vybavení pro získávání a zpracování dat z kamer. Programové vybavení bude detekovat včely na vstupu a sledovat jejich pohyb. Využijte obrazu dvou kamer a snažte se sledovat pohyb včel prostorově. Zhodnoťte dostatečnost instalovaných kamer pro daný úkol, případně stanovte přísnější požadavky na parametry technického vybavení a jeho umístění na úlu. Programové vybavení vyvíjejte v jazyce Python, případně v kombinaci s C/C++. Dílčí algoritmy uspořádejte do formy knihoven tak, aby byly snadno znovu použitelné. Výsledky práce řádně zdokumentujte a okomentujte. Rozsah práce upřesněte po dohodě s vedoucím práce.

Základní doporučená literatura: literaturu dodá vedoucí práce v průběhu řešení bakalářské práce.

Financování práce : UAI PřF JCU a BC AV ČR (projekt Strategie AV 21, Rozmanitost života a zachování ekosystémů)

Vedoucí práce : Ing. Miroslav Skrbek, Ph.D. .....podpis :.....

U externích vedoucích fakultní garant práce.....podpis : .....

Garant oboru bak.. studia (nepožaduje se u zaměření „příprava na mag. studium biologie)

..... podpis : .....

Vedoucí katedry ..... podpis .....

Případný souhlas vedoucího ústavu AV .....podpis : .....

V Českých Budějovicích dne .....

Převzal/a dne..... podpis : .....

Bibliografické údaje

Novotný, P. 2021: Sledování včel na vstupu včelího úlu.[Tracking bees at the entrance of a beehive, Bachelor thesis, in Czech.] - 50 p.,Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace

Hlavním cílem práce je navrhnutí a zprovoznění kamerového systému, pro inteligentní včelí úl umístěny na Experimentální a výukové včelnici Biologického centra AV ČR, v. v. i. v areálu Dendrologické zahrady na Branišovské ulici č. 31 v Č. Budějovicích a vytvoření programového vybavení pro sledování, spočtení a prostorovou identifikaci včel, na vstupu včelího úlu. Systém se bude skládat z kamer připojených k Raspberry Pi, kde dojde ke zpracování snímku pomocí metod zpracování obrazu. Získaná data budou odeslány na server a uložena do databáze. Část získaných výsledků bude zobrazena pomocí vizualizačního nástroje Grafana.

Klíčová slova

Včela, Raspberry Pi, Kamerový systém, Zpracování obrazu, Prostorová identifikace, Objektová identifikace

Annotation

The main goal of this work is to design and put into operation a camera system for an intelligent beehive located at the Experimental and Educational Apiary of the Biological Center of the ASCR, v. VI in the area of the Dendrological Garden on Branišovská Street No. 31 in Č. Budějovice and to create software for monitoring, counting and space identification of bees at the entrance of a beehive.The system

will consist of cameras connected to the Raspberry Pi where it will be processed using the image processing method. The obtained data will be processed on the server and stored in the database. Part of the obtained results will be displayed using Grafana's visualization tools.

Keywords

Bee, Raspberry Pi, Camera system, Image processing, Space identification, Object identification

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne . . . . . Podpis autora. . . . . .

Poděkování

Rád bych poděkoval panu Ing. Miroslavu Skrbkovi, Ph.D. a panu Ing. Václavu Krištůfkovi, CSc. za vedení mé bakalářské práce, cenné rady a vstřícnost při konzultacích.

Obsah

Kapitola 1

Úvod

1.1 Motivace

Projekt kamerového systému pro inteligentní včelí úl mě zaujal, protože v sobě kombinuje hardwarové i softwarové požadavky, a umožňuje mi širokou škálu možných řešení zadaného problému. Zároveň mě těší možnost pracovat na projektu, jenž kombinuje několik vědních oborů včetně informatiky, matematiky a biologie a chovu včel.

V dnešní době dochází k čím dál většímu úhynu včel a včelstev. Behaviorálním zkoumání včel může pomoci při zkoumání proč k tomuto jevu dochází. Výsledek této práce může přispět právě při behaviorálním zkoumání včel.

1.2 Cíle

Hlavním cílem práce je navrhnutí a zprovoznění kamerového systému, pro inteligentní včelí úl umístěny na Experimentální a výukové včelnici Biologického centra AV ČR, v. v. i. v areálu Dendrologické zahrady na Branišovské ulici č. 31 v Č. Budějovicích a vytvoření programového vybavení pro sledování, spočtení a prostorovou identifikaci včel, na vstupu včelího úlu. Systém se bude skládat ze dvou kamer připojených k Raspberry Pi, kde dojde ke zpracování snímku pomocí metod zpracování obrazu. Získaná data budou odeslány na server a uložena do databáze. Část získaných výsledků bude zobrazena pomocí vizualizačního nástroje Grafana.

Systém bude užitečný pro sledování aktivity včelstva během roku, která se řídí ročním obdobím, počasím (teplota, osvit, srážky, atm. tlak), střídáním dne a

noci, fenologií v přírodě, biologickými zákonitostmi rozvoje včelstva během roku a konečně zootechnickými zásahy včelaře.;

V teoretické části je nutné analyzovat současná řešení pro sledování včel a jejich funkcionality. Také je nutné nastudovat teorii počítačového vidění a možnosti získání 3D souřadnic pomocí kamer. Dále by bylo vhodné prozkoumat nejlepší hardwarové a softwarové řešení pro získání nejlepších možných výsledků.

V praktické části bude zprovozněn navržený kamerový systém a získané snímky budou zpracovány počítačovou vizí pro identifikaci, spočtení a získání prostorové polohy včel. Výsledky budou uloženy do Influx databáze a zobrazeny pomocí vizualizačního nástroje Grafana.

Kroky k naplnění hlavního cíle:

1. Rešerše problematiky sledovaní včel a tvoření kamerových systému.
2. Návrh a hardwarová i softwarová realizace kamerového systému.
3. Vytvoření softwarů pro identifikaci včel a jejich umístění v prostoru.
4. Zpracování získaných výsledků.

Práce byla finančně podpořena projektem Strategie AV 21 udělené Biologickému centru AV ČR, v. v. i. v Č. Budějovicích.

Kapitola 2

Teoretická část

2.1 Zpracování obrazu

Zpracování obrazu je metoda, která umožňuje provádět operace na snímku, za účelem zlepšení kvality snímku, nebo získání nějaké užitečné informace ze snímku. Jedná se o druh zpracovaní signálu jehož vstupem je snímek a výstupem je upravený snímek nebo nějaká informace spojená se snímkem. Metoda zpracování obrazu dnes patří mezi rapidně rostoucí technologie. Využívá se hlavně v informatice ale také v dalších vědních oborech.[1]
Zpracování obrazu se v podstatě skládá ze tří kroků:

- Importování snímku přes vhodný nástroj.
- Analýza a manipulace snímku.
- Výstup v podobě snímku nebo informace.

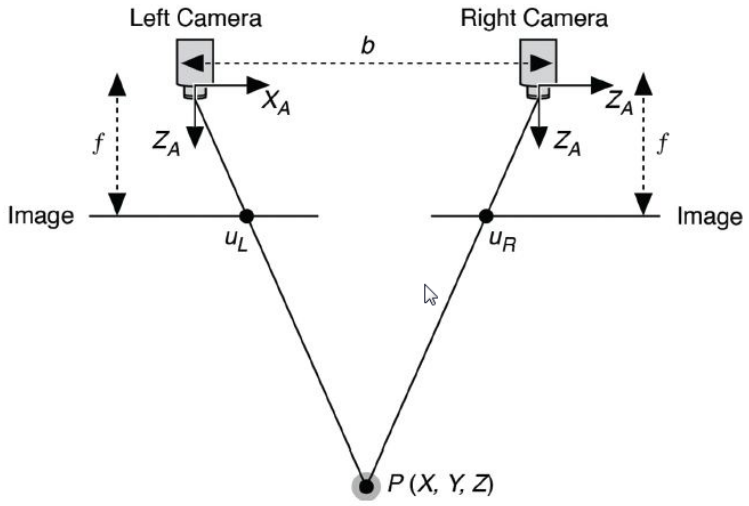
V této práci se bude používat zejména technologie OpenCV.

2.2 Metody pro získání hloubky

2.2.1 Stereo vision

3D rekonstrukce scény, vyžaduje alespoň 2 snímky získané dvěma kamerami ve stejný okamžik. Tomuto principu se říká stereo rekonstrukce nebo stereo vize. Pomocí této metody můžeme odhadnout vzdálenost objektu od kamer. Stereo vize se hojně využívá v robotice, sledovacích zařízeních nebo pro inspekci objektu. Princip

fungování by se dal popsat jako: Snímky ze dvou kalibrovaných kamer poskytují informace o disparitě (vzdálenosti), mezi dvěma odpovídajícími body ve dvou obrazech. Výsledná mapa disparity se používá k určení relativní hloubky ve scéně.[2] Zjednodušený model stereo vize lze popsat následujícím modelem:



Obrázek 2.1: model zjedodušeného systému stereo vize [2]

Vzorec pro zíkání vzdálenosti objektu (P) v tomto modelu je:

$$hloubka = f * b/d \tag{2.1}$$

kde:

f = velikost fokálu
b = vzdálenost mezi kamerami (beseline)
d = vzdálenost mezi odpovídajícími body (desparity)

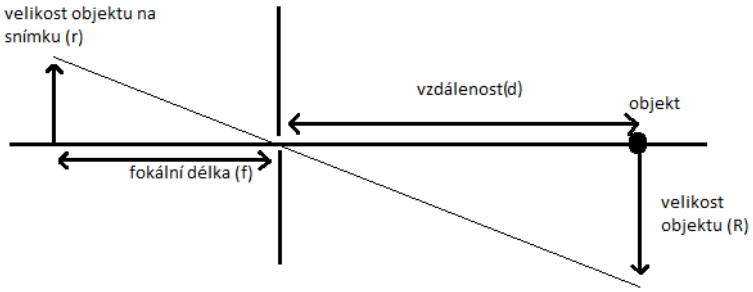
disparitu můžeme získat vzorcem:

$$d = u_l - u_r \tag{2.2}$$

Kde body u1 a u2 jsou vzdálenosti mezi projektovanými body na plátně snímku. [2]

2.2.2 Princip měření vzdálenosti objektu od kamery při známé velikosti objektu

Kalibrovaná kamera vytváří vztah mezi snímáním objektem a jeho obrazem, pracující na principu podobnosti trojúhelníku. Pomocí tohoto principu můžeme odvodit vztah mezi známými parametry, kterými jsou: fokální délka, velikost reálného objektu a velikosti objektu na snímku, a neznámého parametru vzdálenosti .



Obrázek 2.2: princip

Použitím principu podobnosti trojúhelníku můžeme odvodit následující vzorec:

$$f/d = r/R \tag{2.3}$$

po úpravě:

$$d = f * R/r$$

kde:

d = vzdálenost objektu od kamery
f = fokální délka
R = reálná velikost objektu
r = velikost objektu na snímku

Toto je pouze základní princip v reálných podmínkách je nutné provést více výpočtů které budou popsány v praktické části. K získání přesné fokální délky je

nutné provést kalibraci kamery.[3] [4]

Po testování obou metod a posouzení jejich výhod a nevýhod, byla v projektu použita metoda měření vzdálenosti objektu od kamery při známe velikosti objektu.

2.3 Kalibrace kamery

Kalibrace kamery je proces při kterém odhadujeme parametry čoček a obrazových senzorů kamery. Tyto parametry můžeme použít pro opravení zkreslení čoček, měření velikosti objektu v reálných hodnotách (centimetry, metry atd.) nebo získaní polohy kamery ze scény. Parametry kamery zahrnují intersekční, externí a zkreslující koeficienty. Pro odhadnutí těchto parametrů potřebujeme 3D body z reálné scény a odpovídající 2D body ze snímku scény. Tyto odpovídající body získáme pomocí více snímků kalibračního vzoru jako například šachovnice. Tyto snímky je nutné brát z různých úhlů. Vzor by měl být umístěn na pevné a rovné podložce. Výsledkem kalibrace je matice kamery:

$$k = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Zajímají nás zejména parametry α_x a α_y, které se rovnají fokální vzdálenosti krát měřítko pixelů v x a y ose, budeme je potřebovat pro výpočty v praktické části. Pomocí získáních parametrů a zdánlivé fokalní vzdálenosti, kterou udává výrobce, můžeme najít reálnou velikost fokalní vzdálenosti, která se často liší od té udávané výrobcem kvůli nepřesnosti ve výrobě čoček.[5] [6]

2.4 HSV

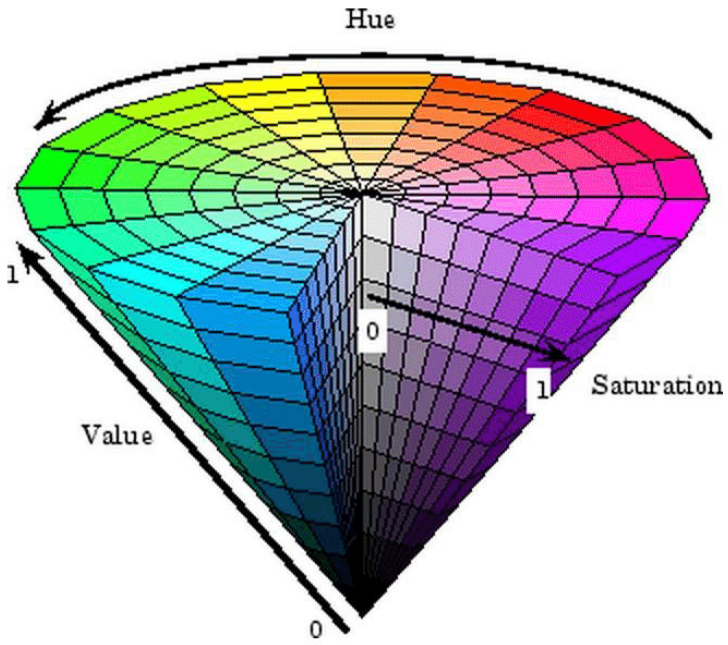
Je cylindrický barevný model který přemapuje RGB barvy do dimenzí, které jsou jednoduší pro lidské pochopení. Tyto dimenze jsou odstín (hue), saturace a hodnota (value).

• Odstín – Jedná se o úhel na RGB barevném kole. 0° úhel je roven červené barvě,

120° úhel se rovná zelené barvě a 240° se rovná modré barvě.

• Saturace – Kontroluje množství použité barvy. 100% saturace bude nejčistější možná barva, naopak 0% saturace bude černobílý obraz.

• Hodnota – Kontroluje jas. 100% jas nemá přimíchanou žádnou černou barvu, naopak 0% jas bude čistě černá barva.[7]



Obrázek 2.3: hsv model

[8]

2.5 Filtrování barev

Filtrovaní barev nebo také segmentace na základě barvy je proces, při kterém rozpoznáváme objekty nebo regiony, které mají specifickou barvu. Pro identifikaci objektu specifické barvy musíme vytvořit práh barev a vytvořit masku k separaci rozdílných barev. Pro vytvoření prahu musíme určit spodní a horní hranici rozsahu barev v nějakém systému barev. Pro filtrování barev se nejvíce hodí HSV systém.[9]

2.6 Kontury

Kontury mohou být popsány jako křivky, spojující body hranice objektu jenž má stejnou barvu, nebo intenzitu. Kontury jsou užitečné pro analyzování tvaru, detekci nebo rozpoznání objektu.[10]

2.7 Odstranění šumu

Odstranění šumu může pomoci při rozpoznávaní objektu. Šum může způsobovat necelistvost hledaných objektů a nalezených kontur. Na druhou stranu metody pro odstranění šumu můžou způsobit snížení ostrosti snímku. Mezi nejpoužívanější metody pro odstranění šumu patří Gaussův filtr. [11]

2.8 Parametry kamery

Kamery často umožňují softwarové nastavení různých parametrů. V Linuxu můžeme tyto parametry nastavit například pomocí nástroje V4L2-CTL. Úprava těchto parametrů může mít velký vliv na kvalitu získaných snímků. Pokud tyto parametry nebudou upraveny jejich hodnoty budou nastaveny na základní hodnotu. Mezi nejdůležitější parametry patří:

- jas – Měrná veličina svítivosti.
- Kontrast – Je rozsah mezi rozdílnými tóny obrazu. Tento rozsah tvoří textury, stíny, barvy a čistotu obrazu.
- Saturace – Kontroluje množství použité barvy.
- Gain – Řídí nastavení zesílení signálu ze snímače kamery. Signál je zesílen jako celek včetně šumu.
- Ostrost – Velikost čistoty detailu a ostrosti hran.
- doba vystavení – Je doba, kdy je senzor kamery vystaven světlu pro získání snímku.

2.9 Srovnání JPG a PNG

2.9.1 JPG

JPG je ztrátový, komprimovaný formát. Tyto vlastnosti ho tvoří jako dobrá volba pro ukládaní fotografií u kterých, preferujeme malou velikost souboru před jeho kvalitou.

2.9.2 PNG

PNG je bezeztrátový, komprimovaný formát. Tyto vlatnosti ho tvoří jako dobrá volba pro ukládaní fotografií u kterých preferujeme kvalitu navzdory velké velikosti souboru.

2.9.3 Srovnání

JPG oproti PNG nabízí nižší velikost souboru, nižší kvalitu, rychlé načítání, nepodporuje transparentnost a umožňuje nastavit úroveň komprese.
PNG oproti JPG nabízí větší velikost souboru, vyšší kvalitu, pomalejší načítaní, podporu transparentnosti a neposkytuje nastavení úrovně komprese.[12]

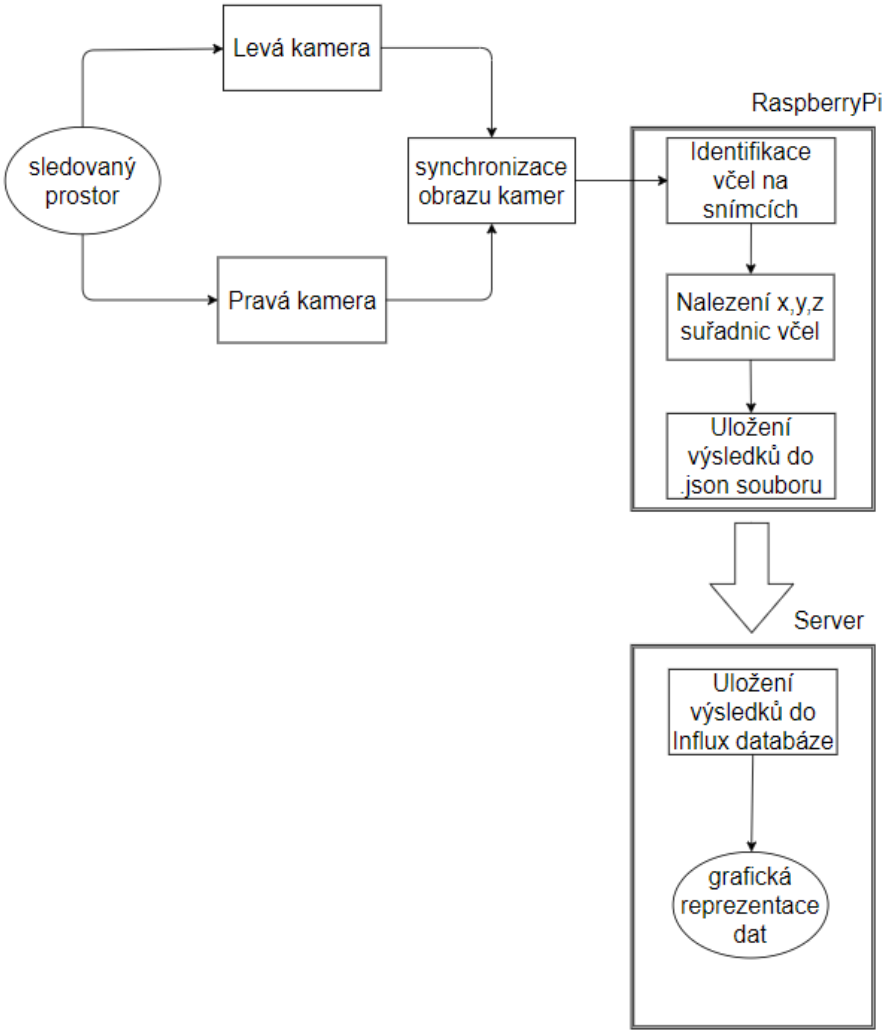
Pro tento projekt bude použit formát JPG, protože snímky nepotřebujeme ve vysoké kvalitě a záleží nám na co nejrychlejší manipulaci se snímky.

Kapitola 3

Návrh řešení

3.0.1 Kamerový systém

Systém se bude skládat ze dvou kamer, umístěných nad vstupem včelího úlu, sledující prostor před ním. Kamery budou připojeny k zařízení Synchronized Stereo Camera HAT, které synchronizuje jejich obraz na úrovni nanovteřin. Pořízení snímků, nastavení kamer a rozdělení snímků bude probíhat na Rasberry Pi. Celý systém bude napájen pomocí POE. Snímky budou následně odeslány na Linuxový server, kde bude probíhat detekce, spočtení a prostorová identifikace včel na snímcích. Následně budou získané výsledky zapsané do Influx databáze. Níže si můžete prohlédnout blokové schéma systému. 3.1



Obrázek 3.1: blokové schéma systému

3.0.2 Programové vybavení

Programové vybavení bylo tvořeno v jazyce Python. Software nacházející se na Raspberry Pi slouží k odebíraní snímků, jejich rozdělení, identifikaci včel na snímcích, získání jejich x,y,z souřadnic a uložení získaných do .json souboru.

Software nacházející se na serveru slouží pro stahování .json souboru z Raspberry Pi a zapsání získaných dat do Influx databáze. Data jsou poté čtena z databáze vizualizačním nástrojem Grafana.

Poté byl vytvořen software pro kalibraci kamer a testování použitého řešení.

3.0.3 Proč nebyla využita stereo vize?

Ačkoliv bylo vytvořeno částečně funkční řešení pro získaní hloubky včel pomocí stereo vize, nebylo možné jej jednoduše implementovat z několika důvodů:

1. Vytvořené řešení mělo příliš velký "šum"vytvořené hloubkové mapy a nebylo možné spolehlivě hloubku získat. Tento šum se dá regulovat pomocí nastavování určitých parametrů vytvořené hloubkové mapy, bohužel bez problémové nastavení se nalézt nepodařilo.

2. Již z principu může stereo vize pracovat pouze na prostoru snímaném oběma kamerami, tudíž prostor sledovaný pouze jednou kamerou nemůže být použit. Tato skutečnost by zapříčinila snížení velikosti sledovaného prostoru o přibližně 40%.

3. Projevilo se jako obtížné získat hledanou vzdálenost Z v reálných rozměrech jako například milimetry.

Z těchto důvodu bylo použito alternativní řešení jenž se projevilo jako spolehlivější.

Kapitola 4

Výběr hardwaru

4.1 Raspberry Pi 3 model B

Raspberry Pi 3 model B, je mikropočítač třetí řady vydán v roce 2016 společností Raspberry Pi Foundation. K mikropočítači je možné připojit mnoho různých zařízení, pomocí 40 pinového GPIO portu a, nebo pomocí dalších portů. Raspberry Pi má také plnou podporu LAN a Wi-Fi připojení. Kompaktní rozměry umožňují použití v místech, kde by bylo použití klasického desktopového počítače nemožné nebo nepraktické. Raspberry Pi 3 nabízí, díky svým rozměrům, poměrně nízký výkon.[13]

4.2 Synchronized Stereo Camera HAT

Synchronized Stereo Camera HAT je rozšíření pro Raspberry Pi, umožnující připojení dvou 5MP OV5647 kamer nebo 8MP IMX219 kamer. Rozšíření umožňuje snímat video, nebo obraz z připojený kamer v synchronizovaném režimu. Raspberry Pi identifikuje připojené kamery jako jednu kameru. Rozšíření je zejména užitečné pro stereo vidění. Kvalita synchronizace se pohybuje v úrovni nanovteřin. Rozšíření se připojuje k Raspberry Pi pomocí GPIO a CSI portu.[14]

4.3 Arducam 8Mpx IMX219 stereo camera board

Jedná se o plošný spoj obsahující dvě 8MP kamery. Kamery jsou od sebe vzdálené 8cm. Vlastnosti kamer:

• Senzor: 1/4"Sony 8MP IMX219

• Frekvence snímků: 30 fps @ 8 Mpx, 60 fps @ 1080p, 180 fps @ 720p

- Formát dat: RAW8 / RAW10
- EFL: 2,8 mm
- F.NO: 2,8
- Zorné pole (FOV): 75 ° (horizontálně)
- Rozhraní: MIPI CSI-2 2linka
- IR citlivost: Viditelné světlo, integrovaný IR filtr[15]

4.4 Napájení systému

Splitter TP-LINK TL-PoE10R plně vyhovuje standardu IEEE 802.3af a podporuje všechny adaptéry PSE Supplier nebo PoE kompatibilní se standardem IEEE 802.3af PoE. Dodává volitelné stejnosměrné napětí 12 V, 9 V nebo 5 V všude tam, kde není elektrické vedení či zásuvka pro připojení zařízení a jeho dosah je až 100 metrů.[16]

POE určeno pro napájení systému ze sítě. Systém využívá nastavení napětí na 5V, při manipulaci se systémem je nezbytné neměnit toto nastavení z důvodů možnosti poškození systému přepětím.

4.5 Krabice

Kamerový systém je umístěn v plastové elektroinstalační krabici o rozměrech 150x110x70mm s certifikací odolnosti IP56, která však byla porušena vyvrtáváním děr pro čočky kamer.

4.6 Linuxový server

Zpracování dat z kamerového systému probíhá na Linuxovém serveru provozovaným Přírodovědeckou fakultou.

Kapitola 5

Výběr softwaru

5.1 Programovací jazyk

Python je vysokoúrovňový, skriptovací,programovací jazyk. Byl vytvořen v roce 1991. Je zpravován a vyvíjen společností Python Software Foundation. Python není zpětně kompatibilní se svými předchozími hlavními verzemi. Poslední verze je 3.8.5. Python byl zvolen pro svou jednoduchost, podporu OpenCv a Rasberry Pi.[17]

5.2 Použité knihovny

5.2.1 OpenCV

OpenCv je open source knihovna, vyvíjena společností Intel. Knihovna je určena zejména k počítačovému vidění a zpracování obrazu. Knihovna je psaná v jazyce C++, ale je možné ji používat v dalších jazycích, například Python nebo Java. OpenCv bylo zvoleno z důvodu, že nabízí algoritmy a funkce, které můžeme použít při identifikaci včel a získat jejich polohu v prostoru. Knihovna také nabízí funkce pro získávání a upravování snímku z kamer. Pro Rasberry Pi OS je nutné provést vlastní překlad a instalaci knihovny.[18]

5.2.2 NumPy

NumPy je open source knihovna určena pro programovací jazyk Python. Knihovna slouží zejména k vyvaření a zpracovávaní polí a matic v jazyce Python. Knihovna také obsahuje kolekci pokročilých matematických funkcí. Knihovna byla zvolena z

důvodu, protože je často používaná spolu s OpenCv a knihovnu v mnoha ohledech doplňuje.[19]

5.2.3 Time

Knihovna Time slouží k různým časovým úkonům. Knihovna bude použita v ovládacím skriptu pro periodické spouštění softwaru v námi zvoleném časovém období (za světla) a pro časová razítka.[20]

5.2.4 Subprocess

Knihovna umožňuje vytvářet nové procesy, připojit jejich vstupy, výstupy a získat jejich návratové kódy. Tento modul může nahradit moduly os.system a os.spawn*. Knihovna je použita pro skripty pracující s Linuxovou operační řádkou.[21]

5.2.5 Glob

Modul glob najde všechny názvy cest odpovídající zadanému vzoru podle pravidel používaných unixovým shellem, i když výsledky jsou vráceny v libovolném pořadí. [22]

5.2.6 Astral

Astral je balíček pro výpočet pohybů slunce a měsíce vůči zemi. V práci bude použit pro každodenní určení východu a západu slunce.[23]

5.3 Raspberry Pi OS

Raspberry Pi OS je oficiálně podporovaný software pro Rasberry Pi. Systém je postavený na základě operačního systému Debian. Systém je dostupný jak ve 32 bitové verzi, tak i v 64 bitové verzi. Raspberry Pi OS má předinstalovanou řadu nástrojů pro práci, programovaní a vzdělávaní s Rasberry Pi. Systém byl zvolen z důvodu své oficiální podpory ze strany Rasberry Pi fundation.[24]

5.4 Video4Linux

Video4linux poskytuje rozsáhlé kolekce ovladačů a programovací prostředí pro televize, zachycující karty, usb kamery, radia, teletext dekodéry mnoho dalších zařízení. Video4linux je v práci použito pro získání ovladačů kamer a nastavení parametrů kamer.[25]

5.5 Linux screen

Pomocí příkazu Linux screen můžete posunout spuštěné terminálové aplikace na pozadí a vrátit je zpět, když je chcete vidět.Podporuje také zobrazení na rozdělené obrazovce a funguje přes připojení SSH. V práci je použit pro běh systému na pozadí. [26]

5.6 InfluxDB

InfluxDB je open source TS databáze, vyvíjena společností InfluxData. Je psaná v jazyce Go a je optimalizovaná pro rychlé ukládaní a získání časově označených dat. Influxdb je určená především pro práci s obsáhlými databázemi. Umožňuje provádět analýzy v reálném čase. Vyznačuje se vysokou propustností, kompresí a dotazováním v reálném čase. Umožňuje zápis dotazu prostřednictvím příkazového řádku, protokolu http, API, klientských knihoven a pluginů pro všechny běžné datové formáty. InfluxDB dokáže zpracovávat miliony datových bodů za sekundu – je podpořena automatickou komprimací dat. Kontinuální dotazy (CQ) a retenční zásady (RP) pomáhají automatizovat datové procesy, včetně vypršení platnosti starých dat. [27]

5.7 Grafana

Grafana je open source vizualizační nástroj, pro zpracování velkého množství dat do přehledných a snadno upravitelných dashboardů. Grafana umožňuje připojení k téměř všem moderním databázovým systémům např. Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL atd. [28]

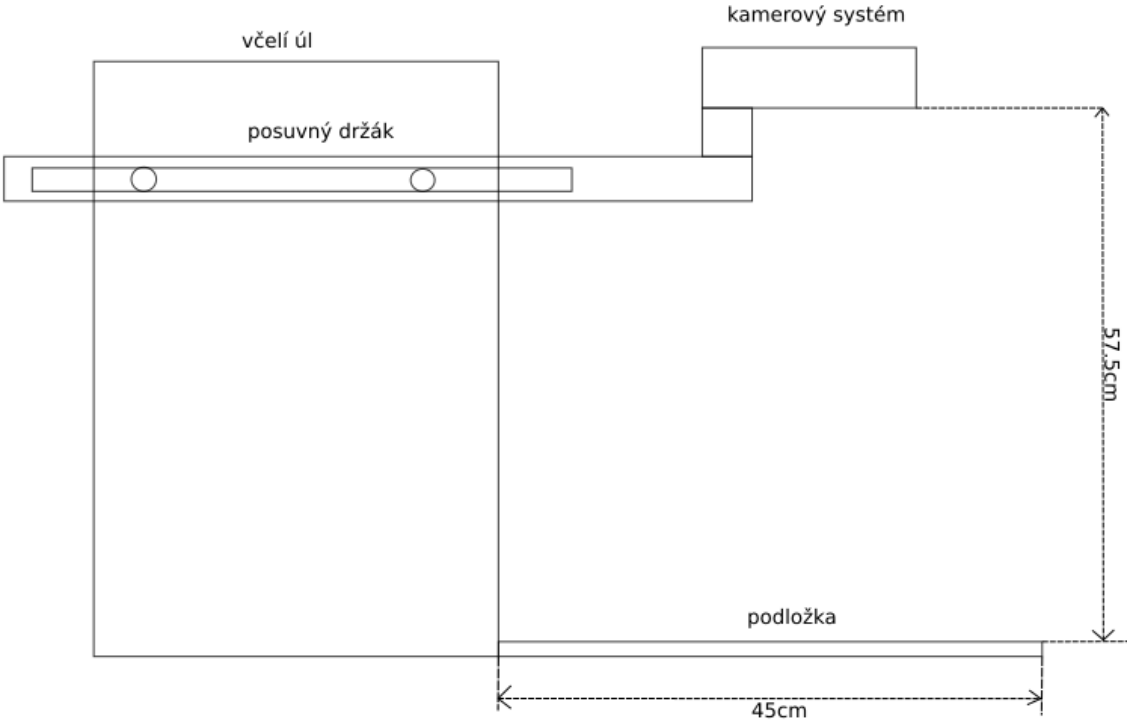
Kapitola 6

Praktická část

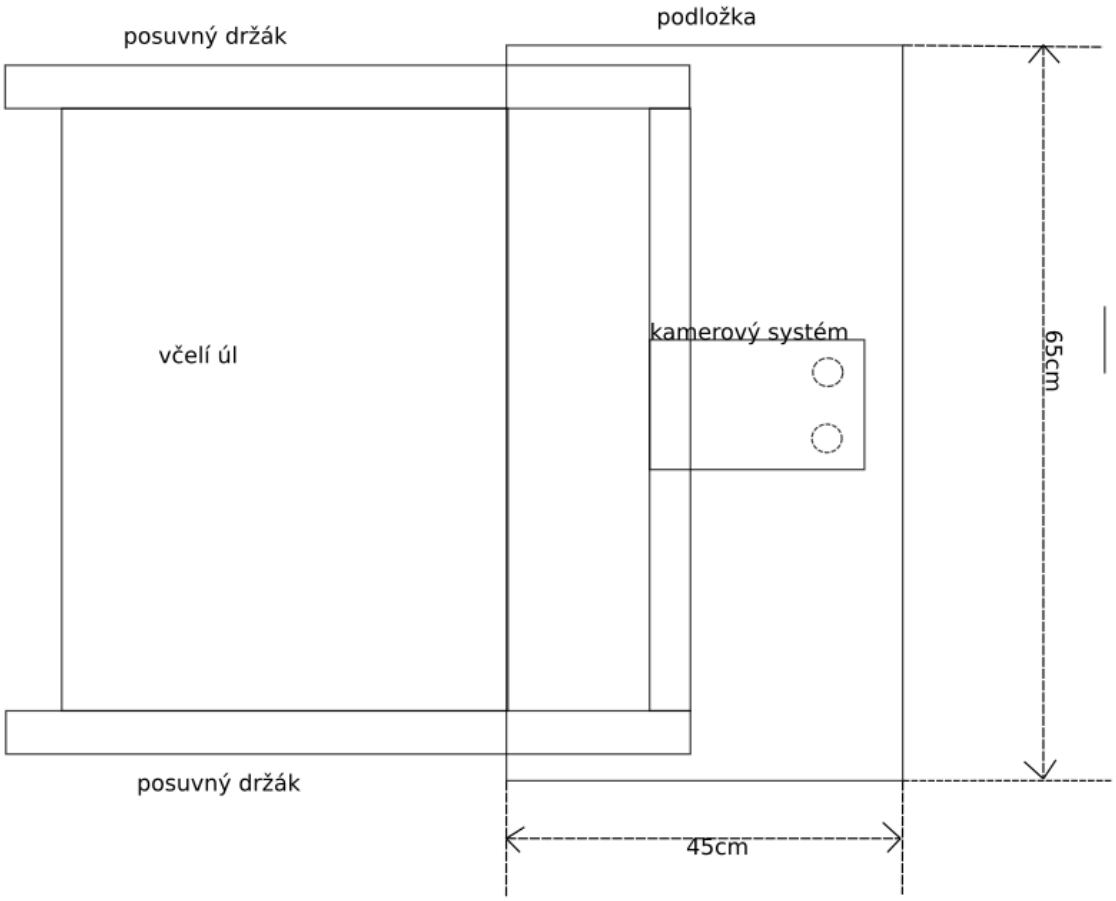
6.1 Hardwarová implementace

6.1.1 Montáž kamerového systému

Kamerový systém je umístěn na posuvném držáku, který je připevněn k úlu umožnující posunout kamerový systém blíže nebo dále od úlu. Kamerový systém je umístěn ve výšce 57.5 cm od sledované podložky. Podložka má bílou barvu, pro co největší kontrast mezi včelami a pozadím, pro lehčí identifikaci. Rozměry podložky jsou 65x45cm, sledovaný prostor je ale o něco menší přibližně 50x40cm.



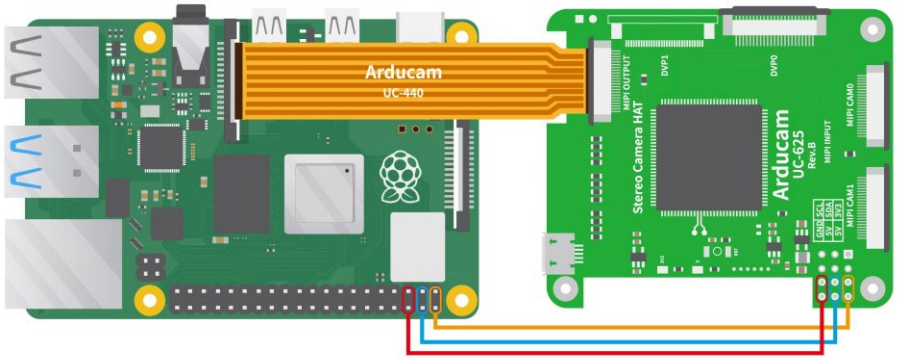
Obrázek 6.1: model systému z boku



Obrázek 6.2: model systému ze shora

6.1.2 Připojení Synchronized Stereo Camera HAT k RaspberryPi

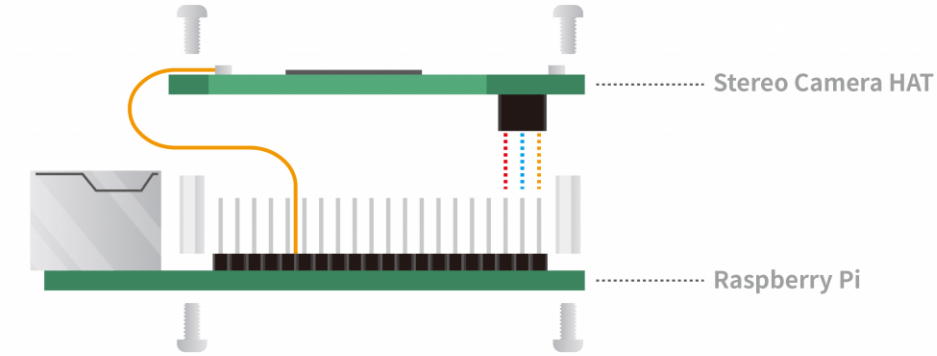
Synchronized Stereo Camera HAT byl připojen k rasberry Pi pomocí 22/15 pinovým ribbon kabelem. 22 pinová strana byla připojena k Raspberry Pi přes CSI port. 15 pinová strana byla připojena k HATu mipi output portu.6.3



Obrázek 6.3: Schéma zapojení Arducam

[29]

2. HAT byl poté připojen k Rasberry Pi pomocí GPIO portu a sešroubován dohromady pomocí přiložených šroubů.6.4



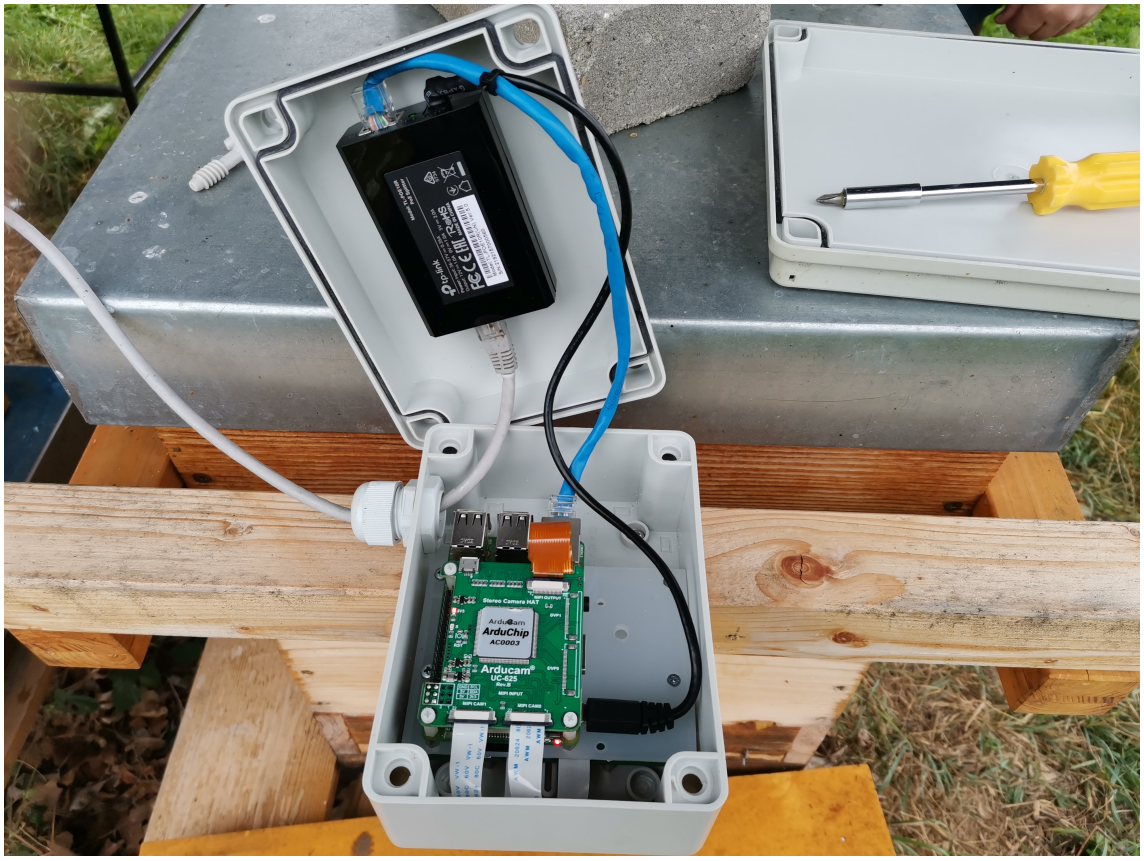
Obrázek 6.4: Schéma instalace Arducam

[29]

6.1.3 Rozvržení systému v krabici

Do dna krabice byly vyvrtány dvě díry, vzdálené od sebe 8 cm, pro čočky kamer. Nad díry byl umístěn kamerový plošný spoj, tak aby čočky kamer koukaly přes díry, ven. Poté byla nad desku umístěna separační plastová deska, na kterou byl umístěn RasberryPi s již připojeným HATem. Na vrchní část krabice bylo umístěno POE

které napájí systém, samotné POE je napájeno pomocí Ethernet kabelu. Z krabice je vyveden pouze Ethernet Kabel. Rozvržení systému jsi můžete prohlédnout na obrázku 6.5.



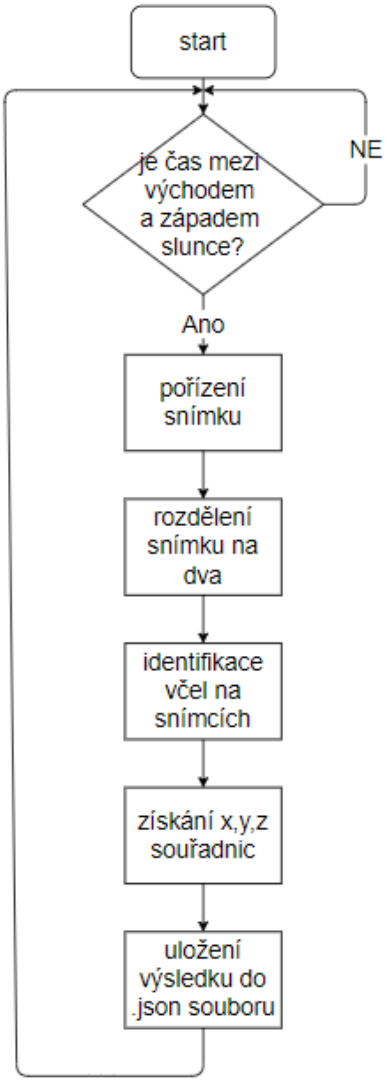
Obrázek 6.5: otevřený systém

6.1.4 Doporučená údržba systému

Včelař by měl při pravidelných návštěvách odstraňovat mrtvé včely, na sledované desce. Mrtvé včely budou způsobovat falešnou pozitivitu nalezení živých včel. Včelař by měl také na měsíční bázi provést umytí desky čistícím prostředkem, kvůli skvrnám které mohou teoreticky také způsobovat falešnou identifikaci i když softwarové řešení malých skvrn je v provozu.

6.2 Softwarová implementace

Hlavní část programu probíhá pomocí skriptu findbee.py, jenž se stará o časový harmonogram, odebírání snímku, rozdělení snímku, identifikaci včel na snímcích, spočtení včel na snímcích, nalezení x,y,z souřadnic včel a uložení výsledků do .json souboru. Celý průběh skriptu jsi můžete prohlédnout na obrázku 6.6.



Obrázek 6.6: diagram

6.2.1 Určení času mezi východem a západem Slunce

Knihovna Astral umožňuje vypočítat východ a západ Slunce pro daný den. Výpočet je proveden na základě známe zeměpisné délky a šířky Českých Budějovic. Samotné určení probíhá takto:
Nejdříve je získán aktuální datum a čas ve správném formátu:

```
now = datetime.datetime.now(datetime.timezone.utc).astimezone()
```

Poté jsou nastaveny informace o lokalitě:

```
loc = LocationInfo(name="CB",region="CZ",timezone="Europe/Prague",
latitude=4.973911,longitude=14.475020)
```

Následně je proveden výpočet pohybů Slunce:

```
s = sun(loc.observer,date=datetime.date.today(),tzinfo=loc.timezone)
```

Nakonec je již vytvořena podmínka, jenž je splněna, když je čas mezi východem a západem Slunce:

```
if (now > (s["sunrise"]) and now < (s["sunset"])):
```

6.2.2 Pořízení snímků a rozdělení snímků na dva

Pro získání snímku jsou použity metody openCV, určené pro práci s kamerami. Snímek je získán v rozlišení 1280x720 pixelů. Snímek obsahuje synchronizovaný obraz, pravé i levé kamery. Následně je získaný snímek o rozměrech rozdělen na dva snímky, pomocí funkcí knihovny NumPy. Získané snímky mají rozlišení 640x720 pixelů a obsahují obraz z pravé respektive levé kamery. Samotné pořízení snímku probíhá takto.
Nejdříve je inicializována kamera a vytvořen VideoCapture objekt pomocí metody:

```
cam = cv2.VideoCapture(-1)
```

Hodnota v metodě označuje číslo kamery, hodnota -1 použije jakoukoliv dostupnou kameru.
Je nastaveno rozlišení kamery na 1280x720 pixelů pomocí metod set:

```
cam.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 1280)
cam.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 720)
```

Následně je získán snímek pomocí metody:

```
ret, frame = cam.read()
```

Následně je získaný snímek rozdělen:

```
sub_image = frame[0:720, 0:640]
sub_image2 = frame[0:720, 640:1280]
```

Tato operace probíhá periodicky pomocí while True cyklu.

6.2.3 Identifikace včel na snímcích

Probíhá pomocí skriptu findbee.py. Nejdříve je daný snímek převeden z formátu RGB do formátu HSV. Následně jsou pomocí předem nalezených rozsahů jednotlivých částí HSV prostoru určeny dolní a horní hranice propustnosti barev. Pomocí těchto hranic je vytvořena maska, která propouští barvy jen vhodného rozsahu odpovídající barvě včel na snímcích. Kolem těchto barevných objektů je použita dilatace pro ucelení kontur. Pak jsou nalezeny kontury těchto barevných objektů, které odpovídají včelám na snímku. Pokud velikost nalezené kontury neodpovídá možné velikosti včely, kontura je nepoužita. Tímto jsou eliminovány drobné nečistoty na snímku je již barva odpovídá barvě včel.

Zjednodušeně algoritmus probíhá takto: Nejdříve je snímek převeden z RGB do HSV.

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Poté jsou nastaveny horní a dolní práh barev, z metod paramr nebo paraml, záleží o jakou kameru se jedná.

```
l_b, u_b = paramr()
```

Posléze je vytvořena separační maska.

```
mask = cv2.inRange(hsv, l_b, u_b)
```

Následně je vytvořen prahová hodnota pro dilataci.

```
ret, thresh = cv2.threshold(res, 0, 255, cv2.THRESH_BINARY)
```

Poté je použita dilatace pro ucelení děr v nalezených konturách.

```
dilated = cv2.dilate(thresh, None, iterations=3)
```

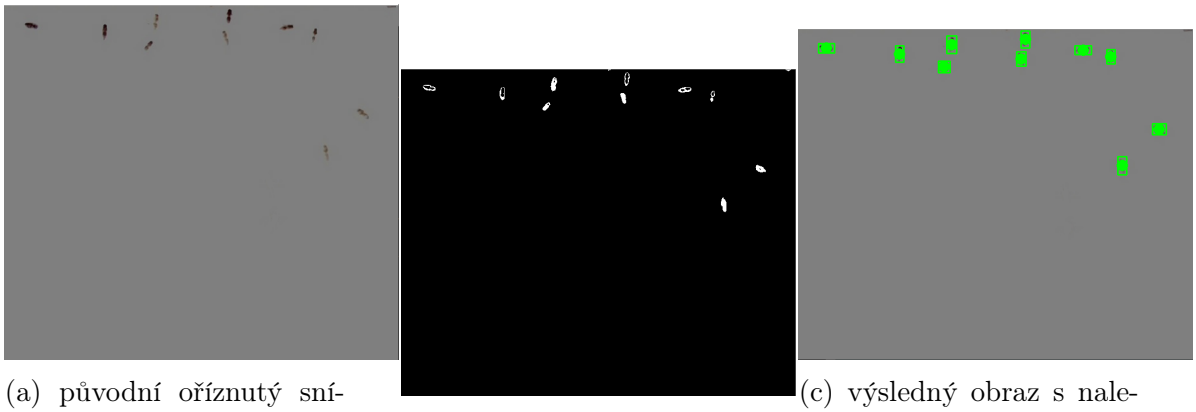
Pak jsou již vytvořeny kontury, kolem nalezených objektů.

```
contours, _ = cv2.findContours(dilated, cv2.RETR_CCOMP,  cv2.CHAIN_APPROX_NONE)
```

Poté následuje cyklus, který hodnotí velikost jednotlivých kontur a zahodí neodpovídající velikosti. (Příliš malé nebo příliš velké.)

```
if cv2.contourArea(contour) < 100 or cv2.contourArea(contour) > 1500  :
                    continue
```

Pozn. Jedná se o zjednodušenou ukázku algoritmu z důvodu poměrně velké komplexnosti a rozsahu reálného algoritmu. Celý algoritmus jsi můžete prohlédnout na přiloženém softwaru.



(a) původní oříznutý snímek

(b) vytvořená maska

(c) výsledný obraz s nalezenými včely

Obrázek 6.7: postup nalezení

6.2.4 Získání x,y,z souřadnic

Probíhá uvnitř skriptu findbee.py. Kolem každé kontury nalezené včely je vytvořen obdélník. Objekt obdélníku obsahuje x,y souřadnice jeho levého rohu a jeho výšku a šířku. Díky těmto parametrům můžeme jednoduchým výpočtem určit souřadnice středu nalezené včely. Výsledkem je x,y hodnota včely v pixelech.

Pro Z parametr vytvoříme okolo nalezené včely rotační obdélník. Rotační obdélník je tvar jehož velikost odpovídá velikosti kontury okolo nalezené včely, potažmo velikosti včely v pixelech. Tuto velikost vezmeme a aplikujeme do našeho vzorce pro získaní hloubky viz sekce 7.2.1.

Výsledné souřadnice pro každou včelu jsou uložené do datového typu tuple, a poté do pole a to v pořadí od nejvíce levé včely až po nejvíce pravou včelu.
Konkretní nalezení souřadnic probíhá takto:

```
for contour in contours:
                (x, y, w, h) = cv2.boundingRect(contour)
```

Poté jsou nalezeny středy těchto obdélníků, které odpovídají středům nalezených včel.

```
bodx = (x+w//2)
```

```
body = (y+h//2)
```

Následně je okolo každé nalezené včely vytvořen rotační obdélník.

```
rect2 = cv2.minAreaRect (contour)
```

Posléze je vzata jeho větší velikost, (Může se jednat o šířku ale i o výšku záleží na poloze včely na snímku.), a je aplikovaná do vzorce pro získání hloubky viz.

```
bodz = zpointr (max( rect2[1]))
```

Nakonec jsou nalezené souřadnice uloženy do tuplu a poté do listu.

```
xyz = (bodx,body,bodz)
listxyz.append(xyz)
```

Pozn. Jedná se o zjednodušenou ukázku algoritmu z důvodu poměrně velké komplexnosti a rozsahu reálného algoritmu. Celý algoritmus si můžete prohlédnout na přiloženém softwaru.

6.2.5 Uložení výsledků do .json souboru

Probíhá ve skriptu findbee.py. Data se ukládají do souboru data.json. Ukládaná data jsou počet včel, souřadnice x,y,z nalezených včel, časové razítko a značku pravé (R) nebo levé (L) kamery.

Konkrétně uložení probíhá takto: Metoda má jako vstupní parametry nalezené výsledky a to počet včel, souřadnice včel a označení kamery . Nejdříve je určen současný čas pomocí knihovny time.

```
t = time.localtime()
```

Poté je získaný čas převeden do vhodného formátu pro časové razítko. Konkrétně je formát rok, měsíc, den, hodina, minuta, vteřina.

```
current_time = time.strftime("%Y:%m:%H:%d:%M:%S", t)
```

Jednotlivá data jsou poté uloženy v datovém typu slovník.

```
data = {
        'pocet': counter,
        'xyz': listxyz,
```

```
        "cas": current_time
         "kamera": flag
}
```

Nakonec jsou data uložena do souboru data.json pomocí metody json.dump(). Pokud soubor neexistuje tak je vytvořen.

```
with open('data.json', 'a') as json_file:
        json.dump(data, json_file)
        json_file.write("\n")
```

6.2.6 Odeslání získaných dat na server

Probíhá jednou denně v noci po ukončení zpracování snímku pro daný den. Soubor s daty data.json je odeslaný na server pomocí protokolu SCP.
Konkrétní podoba příkladu je:

```
scp -P 9104 pi@localhost:/home/pi/data.json /home/novotp38/data.json
```

6.2.7 Uložení výsledků do influx databáze

Probíhá jednou denně pomocí skriptu muj_influx.py. Nejdříve se skript připojí k influx databázi bee_cams. Data ze souboru data.json jsou poté načtena a zpracovaná do formátu vhodného pro influx databázi. Nakonec jsou data uložena do databáze.

Konkrétně uložení do databáze probíhá takto:

Nejdříve je navázaná komunikace s influx databází.

```
client = InfluxDBClient(host='localhost', port=8086, database="bee_cams")
```

Poté je otevřený soubor data.json a jsou načtena jeho data po řádcích.

```
file = "data.json"
with open(file) as f:
data=[]
for line in f:
data.append(json.loads(line))
```

Poté jsou získaná data rozdělena do příslušných proměnných. Souřadnice jsou převedena z datového typu list na string protože influx nepodporuje datový typ list.

```
for index in range(len(data)):


     write0=(data[index].get("pocet"))
     write1=str(data[index].get("xy"))
     write2= (data[index].get("kamera"))
     write3 =(data[index].get("cas"))
```

Poté jsou data převedena na formát slovníku, ve vhodném formátu pro zápis do databáze. Následně je opět převeden do formátu list.

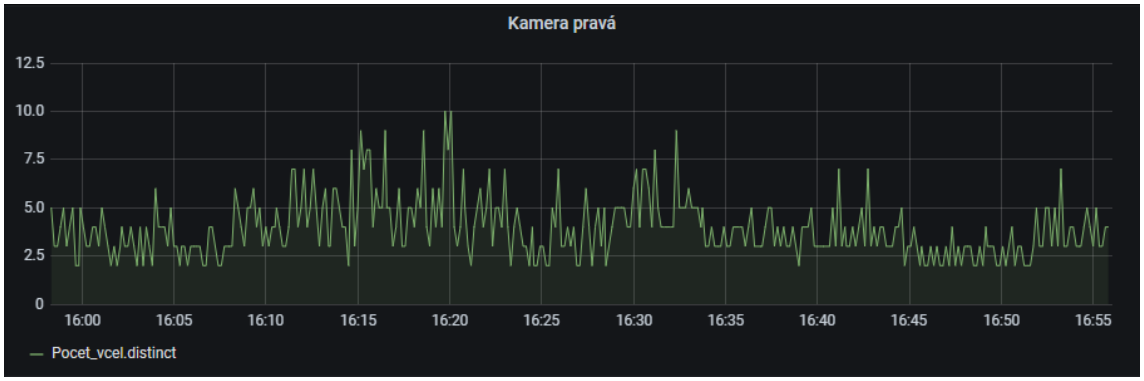
```
json_body = [
{
    "measurement": "Pocet_vcel",
    "fields": {
    "pocet":write0,
    "xy":write1,
    "kamera":write2,
              },
                "time": write3,
}
]
```

Nakonec je proveden samotný zápis.

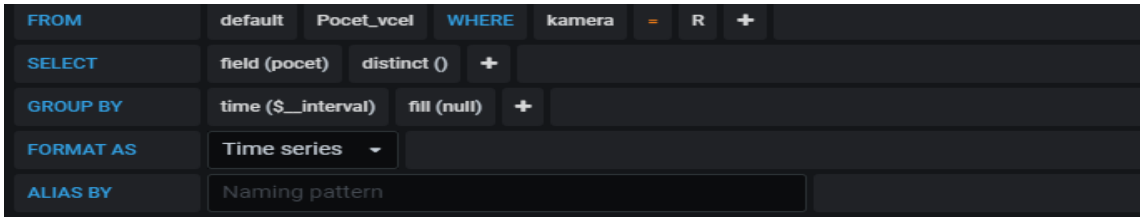
```
client.write_points(json_body, time_precision = "s")
```

6.2.8 Grafické zobrazení výsledků

Část výsledků je zobrazena, konkrétně počet včel z pravé i levé kamery v čase, pomocí vizualizačního nástroje Grafana, který čte získaná data z Influx databáze, pomocí vhodného select dotazu. Ukázku vygenerovaného grafu, z pravé kamery, jsi můžete prohlédnout níže na obrázku.6.9



Obrázek 6.8: ukázka grafu



Obrázek 6.9: ukázka vytvořeného SELECT dotazu v Grafaně

6.2.9 Nastavení parametrů kamery

Nastavení parametrů kamery probíhá pomocí skriptu settings.py. Pro nastavení je použita technologie Video4Linux. Nastavované parametry jsou jas, kontrast, saturace a ostrost.

Konkrétně nastavení probíhá takto:

```
subprocess.call(["v4l2-ctl -d /dev/video0 -c brightness=35"],shell="True")
subprocess.call(["v4l2-ctl -d /dev/video0 -c contrast=0"],shell="True")
subprocess.call(["v4l2-ctl -d /dev/video0 -c saturation=0"],shell="True")
subprocess.call(["v4l2-ctl -d /dev/video0 -c sharpness=0"],shell="True")
```

Nejdůležitějším parametrem je jas jenž bylo potřeba nastavit tak aby fungoval při co nejširším rozsahu světelných podmínek.

6.2.10 Výpočty potřebné pro získání vzdálenosti

Pomocí výsledků z kalibrace kamer, triangulací a principu podobnosti trojúhelníku můžeme získat vzdálenost objektu. Následuje popis výpočtu pro levou kameru.

Pro začátek budeme potřebovat znát fokální vzdálenost udávanou výrobcem, a velikost včely, které jsou:

f = 3.04mm

velikost včely = 15mm

První krok je vypočítat počet pixelů na milimetr v nativním rozlišení: Díky kalibraci známe hodnoty α_x a α_y, které odpovídají fokální vzdálenosti krát měřítko pixelů na milimetr. Protože známe fokální vzdálenost i hodnoty α_x a α_y, můžeme vypočítat měřítko v nativním rozlišení snímku.

$$\alpha_x = f * m_x \tag{6.1}$$

$$\alpha_y = f * m_y$$

po úpravě:

$$m_x = \alpha_x / f$$

$$m_y = \alpha_y / f$$

po dosazení:

$$m_y = 1051/3.04$$

$$m_x = 1056/3.04$$

výsledek:

$$m_y = 345,72$$

$$m_x = 347,36$$

Vzhledem k velké blízkosti výsledků obou výpočtů můžeme použít jako výsledek jejich průměr po zaokrohlení (m = 346,5).

Poté musíme převést velikost pixelu na milimetr do našeho rozlišení:

$$n/m = r/x \tag{6.2}$$

po úpravě:

$$x = r/(n/m)$$

po dosazení:

$$x = 635/(720/346.5)$$

výsledek:

$$x = 305.73$$

kde:

r = výškové rozlišení snímku po oříznutí
n = výškové rozlišení snímku před oříznutím
m = měřítko

Poté musíme znát velikost měřené včely v pixelech. Můžeme ji získat vytvořením rotačního obdélníku kolem nalezené včely, pomocí metody cv2.minAreaRect() a změřením jeho velikosti.

Řekněme že, měřená včela měla například 30 pixelů.
Nyní vezmeme velikost měřené včely a vydělíme ji naším poměrem x.

$$r = 30/305.73 \tag{6.3}$$

výsledek:

$$r = 0.098mm$$

Nyní již máme vše potřebné pro vzorec na určení vzdálenosti:

$$d = R * f/r$$

po dosazení:

$$d = 15 * 3.04/0.098$$

výsledek:

$$d = 480mm$$

Toto je vzdálenost včely od kamery, která má na snímku 30 pixelů na levé kameře.

Stejné výpočty byly provedeny pro pravou kameru, kde se výsledky mírně liší

kvůli rozdílu v čočkách pravé a levé kamery.

6.2.11 Kalibrace kamery

Je provedena skriptem calib.py. Jako vzor pro kalibraci byla zvolena šachovnice 5x8 polí 6.10a, ale byla použita pouze vnitřní část polí 3x6, protože ne každý snímek zaznamenal celou šachovnici. Pro získaní snímku šachovnice byly užity lehce upravené skripty capture.py a crop.py. Při pořizovaní snímku byla měněna poloha, úhel a vzdálenost šachovnice. Šachovnice byla přidělaná k rovnému a pevnému kartonu. Bylo pořízeno celkem 50 kalibračních snímku z pravé i levé kamery a rozděleny do složek R a L.

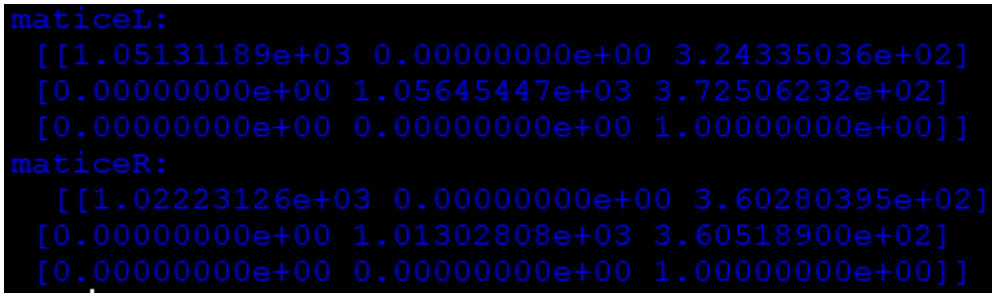
(a) ukázka kalibračního snímku

(b) ukázka nalezených bodů

Obrázek 6.10: kalibrační snímek a nalezené body

Samotná kalibrace probíhá takto:

Nejdříve jsou nastavena kritéria pro kalibraci. Poté jsou inicializované pole do kterých budou ukládány body objektu a body snímku objektu. Následně jsou hledány rohy vnitřní šachovnice v každém získaném snímku 6.10b. Pokud jsou rohy nalezeny pro pravý i levý snímek jsou body uloženy do odpovídajících polí. Pak je provedena kalibrace kamer a získaná nová intersekční matice kamery pomocí bodů, které byly nalezeny 6.11. Výsledná matice byla opsaná pro další operace.



Obrázek 6.11: Výsledky kalibrace

Zjednodušené fungování algoritmu:

Nejdříve jsou importovány kalibrační snímky:

```
imagesL = glob.glob('C:/Users/petr/Desktop/L (1)/*.png')
imagesR = glob.glob('C:/Users/petr/Desktop/R (1)/*.png')
```

Poté jsou stanovena kritéria pro běžnou i stereo kalibraci:

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
criteria_stereo= (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
```

Poté jsou připraveny body objektu.

```
objp = np.zeros((3*6,3), np.float32)
objp[:,:2] = np.mgrid[0:3,0:6].T.reshape(-1,2)
```

Poté jsou připravena pole pro skladování 3D bodů v reálném objektu a pro skladování 2D bodů na snímcích.

```
objpoints = []
imgpoints = []
```

Potom je každý snímek načten, převeden na černobílý a jsou nalezeny rohy šachovnice.

```
for fname in images:
        img = cv2.imread(fname)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        ret, corners = cv2.findChessboardCorners(gray, (3,6),None)
```

Pokud jsou body úspěšně nalezeny jsou uloženy do obou polí a jsou na snímcích vykresleny.

```
if ret == True:
            objpoints.append(objp)
            corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
            imgpoints.append(corners2)
            img = cv2.drawChessboardCorners(img, (3,6), corners2,ret)
            cv2.imshow('img',img)
```

Nakonec je provedena samotná kalibrace kamer a jsou získané nové matice.

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1],criteria_stereo,flags)
img = cv2.imread('C:/Users/petr/Desktop/R (1)/r1.png')
    h,w = img.shape[:2]
    newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))
```

Pozn. Jedná se o zjednodušenou ukázku algoritmu z důvodu poměrně velké komplexnosti a rozsahu reálného algoritmu. Celý algoritmus jsi můžete prohlédnout na přiloženém softwaru.

6.3 Testování

6.3.1 Testování přesnosti měření hloubky a vliv rozlišení snímku

Byl proveden pokus na mrtvých včelkách umístěných na sledované podložce. Díky známe hloubce podložky (575mm) můžeme zjistit o kolik % se liší naměřená hloubka od očekávané hloubky. Stejné včely byly testovány pro pravou i levou kameru.
Pravá kamera:

Číslo včely	očekavaná hloubka	naměřená hloubka	rozdíl v %
1	575mm	669.08mm	+16.36%
2	575mm	638.669mm	+11.07%
3	575mm	625.704mm	+8.81%
4	575mm	529.69mm	-8.9%
5	575mm	638.669mm	+11.07%

Levá kamera:

Číslo včely	očekavaná hloubka	naměřená hloubka	rozdíl v %
1	575mm	649.41mm	+12.94%
2	575mm	619.89mm	+7.81%
3	575mm	607.30mm	+5.62%
4	575mm	518.38mm	-9.85%
5	575mm	619.89mm	+7.81%

Přesnost systému se tedy jeví být mezi 5 - 20%. Další testy na živých včelách jsou obtížné kvůli neznámé reálné hloubce letící včely.

Nejvyšší vliv na měření hloubky systému má rozlišení snímku, čím vyšší rozlišení tím přesněji by bylo možné určit velikost včely na snímcích a zvýšit přesnost systému. Při změně rozlišení by bylo potřeba adekvátně upravit výpočty.

6.3.2 Testování přesnosti identifikace včel na snímcích

Testování bylo prováděno metodou ručního vizuálního spočtení včel na snímcích a jeho porovnání s výsledky získané algoritmem. Posuzovaná byla kvalita identifikace v ruzných světelných podmínkách. Testování bylo prováděno na nahodně vybraných snímcích.

Světelné podmínky	přesnost identifikace
Denní světlo zataženo	Vysoká přesnost identifikace.
Přímé sluneční světlo	Snížena přesnost identifikace na středu snímku.
Rozednění	Zvýšená falešná identifikace.
Stmívání	Zvýšená falešná identifikace.

6.3.3 Testování rychlosti systému

Byla měřena rychlost systému od získání snímku až po zápis získaných výsledků do .json souboru. Systém dokáže spracovat jeden pár snímků za 0.15 vteřiny, tudíž systém je schopen zpracovat až 7 páru snímku za vteřinu. Vzhledem k příliš velkému objemu dat, který by při této rychlosti odebíraní snímků vzniknul, byl systém upraven aby zpracovával jeden pár snímků za vteřinu.

6.4 Návrhy na budoucí zlepšení projektu

Výměna rasberry-pi za výkonnější mikropočítač by umožnila zvýšení rozlišení získaných snímků. Vyšší výkon by dále umožnil zpracovávat video místo statických snímků. Projekt by dále profitoval ze zavedení strojového učení, které by umožňovalo rozpoznávání jednotlivých druhů včel. Bylo by dobré zavést systém, který rozpoznává aktuální počasí a přizpůsobuje parametry systému podle něj.

Kapitola 7

Závěr

Hlavním cílem práce bylo navrhnutí a zprovoznění kamerového systému pro inteligentní včelí úl umístěný na Experimentální a výukové včelnici Biologického centra AV ČR, v. v. i. v areálu Dendrologické zahrady na Branišovské ulici č. 31 v Č. Budějovicích

Systém je tvořen mikropočítačem Raspberry Pi ke kterému byly připojeny dvě kamery přes zařízení synchronizující jejich obraz. Systém je napájeno ze sítě pomocí POE. Systém byl umístěn do elektroinstalační krabice, která byla umístěna nad sledovaný prostor před včelím úlem. Polohu systému je možné částečně regulovat pomocí nastavitelného držáku.

Programové vybavení pro zpracování obrazu z kamer za účelem monitorování pohybu včelstva před včelím úlem, bylo tvořeno v jazyce Python a hojně využívá technologii OpenCV. Systém dokáže na snímcích včely rozeznat, spočíst je, zjistit jejich polohu v prostoru. Časový harmonogram spuštění a vypnutí systému se přizpůsobuje každodennímu východu a západu slunce. Získaná data jsou uložena do .json souboru, jednou za den jsou odeslána na server kde dojde k jejich zápisu do Influx databáze. Následně je část získaných dat, konkrétně počet včel v čase zobrazen pomocí vizualizačního nástroje Grafana.

Bylo vytvořeno celkem 6 skriptů. Z nichž dva se nacházejí na RaspberryPi a slouží k identifikaci včel, získání jejich x,y,z souřadnic, uložení do souboru a nastavení kamer. Dva se nacházejí na serveru a slouží ke stažení dat z RaspberryPi a k zápisu do Influx databáze. Poslední dva slouží pro kalibraci kamer a testování použitého řešení.

Systém byl nasazen do provozu přibližně v polovině Února roku 2021. Drobné

aspekty systému jsou od jeho spuštění upravovány dle potřeby.

Jednotlivé aspekty systému byly testovány a byla zhodnocena jejich přesnost, při různých světelných podmínkách. Dále byla měřena rychlost systému, která dosahovala rychlosti zpracování až 7 párů snímků za vteřinu, kvůli příliš velkému objemu dat, který by při této rychlosti vznikl byl systém upraven na rychlost odebírání snímků jednou za vteřinu.

Systém je užitečný pro sledování aktivity včelstva během roku, která se řídí ročním obdobím, počasím (teplota, osvit, srážky, atm. tlak), střídáním dne a noci, fenologií v přírodě, biologickými zákonitostmi rozvoje včelstva během roku a konečně zootechnickými zásahy včelaře.

Bibliografie

[1] Gholamreza Anbarjafari PhD. *Introduction to image processing [online]*. [cit. 2021-15-2]. URL: https://sisu.ut.ee/imageprocessing/book/1.

[2] Chris Walker. *Stereo Vision Basics [online]*. 2014 [cit. 2021-10-3]. URL: https://chriswalkertechblog.blogspot.com/2014/03/stereo-vision-basics.html.

[3] PhD Adrian Rosebrock. *Find distance from camera to object/marker using Python and OpenCV [online]*. 2016 [cit. 2021-15-2]. URL: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/.

[4] Cameron Lowell Palmer. *How to calculate distance given an object of known size [online]*. 2016 [cit. 2021-15-2]. URL: https://stackoverflow.com/questions/14038002/opencv-how-to-calculate-distance-between-camera-and-object-using-image/.

[5] Alexander Mordvintsev & Abid K. *Camera Calibration [online]*. 2013 [cit. 2021-10-3]. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.htm.

[6] The MathWorks. *What Is Camera Calibration? [online]*. 2020 [cit. 2021-10-3]. URL: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[7] Programmingdesignsystems. *Color models and color spaces [online]*. [cit. 2021-10-3]. URL: https://programmingdesignsystems.com/color/color-models-and-color-spaces/index.html.

[8] Ravindran G. *Illustration of the HSV Color Space [online]*. 2017 [cit. 2021-10-3]. URL: https://www.researchgate.net/figure/Illustration-of-the-HSV-Color-Space-B-Color-Feature-Extraction-Color-feature-is-extracted_fig1_321126312.

[9] Sourabh Sinha. *Filter Color with OpenCV [online]*. 2021 [cit. 2021-10-3]. URL: https://www.geeksforgeeks.org/filter-color-with-opencv/.

[10] OpenCV. *Contours : Getting Started [online]*. 2021 [cit. 2021-10-3]. URL: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html.

[11] OpenCV. *Smoothing Images [online]*. 2021 [cit. 2021-10-3]. URL: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html.

[12] Allison Boatman. *JPG vs. PNG: Which Should I Use?[online]*. [cit. 2021-10-3]. URL: https://www.techsmith.com/blog/jpg-vs-png/.

[13] Raspberry Pi Foundation. *Raspberry Pi 3 Model B [online]*. [cit. 2021-10-2]. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[14] Arducam. *Synchronized Stereo Camera HAT for Raspberry Pi [online]*. 2021 [cit. 2021-10-2]. URL: https://www.arducam.com/product/b0195-synchronized-stereo-camera-hat-raspberry-pi/.

[15] Arducam. *Arducam 8MP Synchronized Stereo Camera Bundle Kit for Raspberry Pi[online]*. 2021 [cit. 2021-10-2]. URL: https://www.arducam.com/product/synchronized-stereo-camera-hat-raspberry-pi-8mp-imx219-board/.

[16] CZC.cz. *TP-LINK TL-PoE10R POE [online]*. 2012 [cit. 2021-10-3]. URL: https://www.czc.cz/tp-link-tl-poe10r-poe/62784/produkt.

[17] Python Software Foundation. *Python [online]*. 2021 [cit. 2021-10-2]. URL: https://www.python.org/about/.

[18] Alexander Mordvintsev & Abid K. *Introduction to OpenCV-Python Tutorials [online]*. 2013 [cit. 2021-1-2]. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html#intro.

[19] The SciPy community. *What is NumPy? [online]*. 2021 [cit. 2021-12-2]. URL: https://numpy.org/doc/stable/user/whatisnumpy.html.

[20] Python Software Foundation. *time — Time access and conversions [online]*. 2021 [cit. 2021-12-2]. URL: https://docs.python.org/3/library/time.html.

[21] Python Software Foundation. *subprocess — Subprocess management [online]*. 2021 [cit. 2021-13-2]. URL: https://docs.python.org/3/library/subprocess.html.

[22] Python Software Foundation. *glob — Unix style pathname pattern expansion [online]*. 2021 [cit. 2021-10-3]. URL: https://docs.python.org/3/library/glob.html.

[23] Simon Kennedy. *Astral v2.2 [online]*. 2020 [cit. 2021-14-2]. URL: https://astral.readthedocs.io/en/latest/.

[24] Raspberry Pi Foundation. *Raspberry Pi OS [online]*. 2021 [cit. 2021-13-2]. URL: https://www.raspberrypi.org/software/.

[25] Linux Kernel's documentation. *V4L2 drivers [online]*. [cit. 2021-13-2]. URL: https://www.kernel.org/doc/html/v4.12/media/kapi/v4l2-intro.html.

[26] Dave Mckay. *How to Use Linux's screen Command [online]*. 2020 [cit. 2021-14-2]. URL: https://www.howtogeek.com/662422/how-to-use-linuxs-screen-command/.

[27] nethost. *influxDB [online]*. [cit. 2021-10-3]. URL: https://www.nethost.cz/influxdb.

[28] Shivang. *What is Grafana? Why Use It? Everything You Should Know About It [online]*. [cit. 2021-14-2]. URL: https://www.8bitmen.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/.

[29] Arducam. *Sync Stereo Camera HAT User Manual*. 2020 [cit. 2021-10-3]. URL: https://www.arducam.com/docs/cameras-for-raspberry-pi/synchronized-stereo-camera-hat/sync-stereo-camera-hat-user-manual/.

Seznam obrázků

Seznam rovnic a tabulek

Seznam Příloh

Přílohy

A. Fotografie systému

Obrázek 7.1: Čelní pohled na experimentální úl se zabudovaným kamerovým systémem

Obrázek 7.2: Posuvné úchyty pro nastavení vzdálenosti systému

Obrázek 7.3: Pohled na uzavřený systém

B. Zdrojové Kódy

Kompletní zdrojové kódy jsou umístěny ve školním repozitáři na Gitlabu.

odkaz: https://gitlab.prf.jcu.cz/novotp38/bp-sledovani-vcel-na-vstupu-vceliho-ulu

Nebo na přiloženém .zip souboru.