VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

RASPBERRY PI: PROGRAMOVÁNÍ V PROSTŘEDÍ MATLAB/SIMULINK
RASPBERRY PI: PROGRAMMING BY MEANS OF MATLAB/SIMULINK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vincent Dadej**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Radomil Matoušek, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Vincent Dadej**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Raspberry Pi: programování v prostředí Matlab/Simulink

Stručná charakteristika problematiky úkolu:

S využitím prostředí Matlab/Simulink (MATLAB® Support Package for Raspberry Pi® Hardware) budou vytvořeny dvě demonstrační úlohy pro vestavěnou platformu Raspberry Pi 2/3 a kamerový modul. Vývojový kit s příslušenstvím i programové vybavení je k dispozici.

Cíle diplomové práce:

Seznámení s vývojovou platformou Raspberry Pi.
Seznamení s MATLAB® Support Package for Raspberry Pi® Hardware.
Po konzultaci se školitelem navrhnout dvě demonstrační úlohy využívající kamerový modul.
Vytvořit popis úloh, popis implementace a základní popis užitých algoritmů.
Realizace úloh a vytvoření video záznamu.

Seznam doporučené literatury:

Eben, Upton; Gareth, Halfacree: Raspberry Pi, COMPUTER PRESS, 2016. Martin Stříž, Bučovice, 2015. 200 s. ISBN: 978-80-251-4819-8

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Táto diplomová práca sa zaoberá programovaním v prostredí Matlab pre platformu Raspberry Pi 3. Za účelom predvedenia sú spracované dve úlohy pre Raspberry Pi, ktoré využívajú dostupný hardware, kameru. Prvou je detegovanie farebného objektu a možnosti presného track-ovania využitím kalibrácie kamery. Druhou je detekcia a rozpoznávanie tváre. V práci je pre každú úlohu vytvorená aplikácia, v ktorej sú implementované moderné metódy a poznatky počítačového videnia. Presnosť track-ovania objektov a rozpoznávania tváre je overená experimentom, ktorý preukázal skutočnú presnosť použitých metód.

ABSTRACT

The diploma thesis focuses on programming in the Matlab for the Raspberry Pi 3 platform. For the purpose of the presentation, there are two applications designed for Raspberry Pi that are using available hardware, camera and servos. The first application serves as colour object detecting and accurate tracking by using camera calibration. The second application serves as a face detection and recognition. These applications are implemented by modern methods and knowledge of computer vision. Tracking of the objects and face recognition are verified by an experiment that reveals the accuracy of the used methods.

KĽÚČOVÉ SLOVÁ

Raspberry Pi, program Matlab, detekcia farby, detekcia tváre, rozpoznávanie tváre, kalibrácia kamery, presné určovanie polohy

KEYWORDS

Raspberry Pi, Matlab, color detection, face detection, face recognition, camera calibration, accurate positioning

BIBLIOGRAFICKÁ CITÁCIA

DADEJ, V. *Raspberry Pi: programování v prostředí Matlab/Simulink,* Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2017, XX s., Vedoucí diplomové práce doc. Ing. Radomil Matoušek, Ph.D..

POĎAKOVANIE

Chcem poďakovať svojmu vedúcemu diplomovej práce doc. Ing. Radomil Matoušek, Ph.D. za odborné vedenie, poskytnutý hardware, cenný čas a rady, ktoré som dostával pri početných konzultáciách. Tak isto by som chcel poďakovať svojej rodine za podporu počas celého štúdia.

ČESTNÉ PREHLÁSENIE

Prehlasujem, že táto práca je mojím pôvodným dielom, spracoval som ju samostatne pod vedením doc. Ing. Radomil Matoušek, Ph.D a s použitím literatúry uvedenej v zozname.

V Brne dne 26.5.2017 ……

Bc.Vincent Dadej

OBSAH

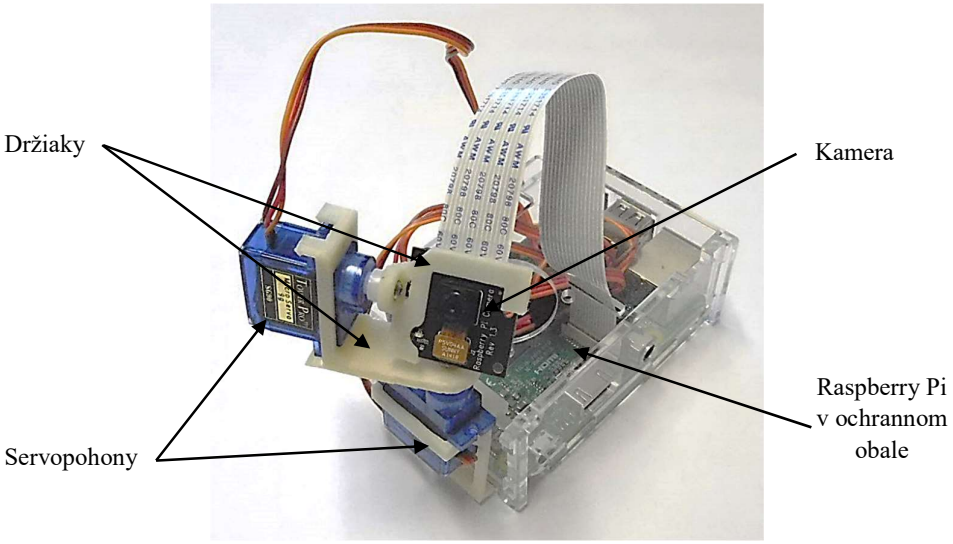
1 ÚVOD

Vstavané systémy (embedded systems) sú v súčasnej dobe vďaka vyspelej technike dostatočne výkonné a tak je ich možné využiť pri riešení najrôznejších technických problémov. Sú vybavené množstvom rozhraní a tak schopné komunikovať a riadiť rôzne elektronické zariadenia. Môžu byť optimalizované na konkrétne účely alebo súčasťou komplexných systémov. Vstavané systémy sa vyrábajú čo najviac univerzálne a tak ich je možno produkovať vo veľkých sériách pri nízkych nákladoch. Jedným z týchto vstavaných systémov je Raspberry Pi. Vyznačuje sa hlavne vynikajúcim pomerom ceny a výkonu. Je veľmi rozšírený po celom svete a svoje schopnosti platforma preukázala v rôznych implementáciách, ako napríklad mini PC, foto pasca alebo riadiaci systém robota.

Cieľom práce bolo navrhnúť dve úlohy pre Raspberry Pi a ako programovacie prostredie použiť program Matlab. Pri návrhu úloh boli použité kamerový modul a servopohony. Ako zaujímavé úlohy, ktoré budú spĺňať požiadavky, boli vybrané úlohy:

- detekcia a track-ovanie farebných objektov a
- detekcia a rozpoznávanie tvári človeka.

Na obrázku Obr. 1) je ukážka spoločnej zostavy prvkov, ktorá bola využitá pri realizácii v obidvoch úlohách. V práci je použitý vstavaný systém Raspberry Pi 3 model B a software Matlab verzie 2016b.



Obr. 1) Zostava použitých prvkov

V práci je popísané hardware a software vybavenie, nutné k realizácii úloh. Ako hardware bude predstavený vstavaný systém Raspberry Pi, jeho detaily a vlastnosti. Taktiež kamerový modul, servopohony a ich vzájomná súčinnosť s platformou. Ako software sa predstaví program Matlab a ozrejmí sa jeho spolupráca s Raspberry Pi. Najprv bude popísané jeho prostredie, použité toolboxy a ich inštalácia. Tieto toolboxy, obsahujú funkcie pre komunikáciu s Raspberry Pi a jeho periférami, a funkcie počítačového videnia. Pre ukážku bude spracovaná podkapitola ktorá obsahuje základné príkazy práce s platformou Raspberry Pi v Matlab-e.

V úlohách sa budú implementovať súčasné poznatky v oblasti počítačového videnia, ktorého cieľom je strojovým spracovaním obrazu, nahradiť schopnosť ľudského vnímania objektov v obraze. V teoretickej časti práce budú popísané princípy procesu počítačového videnia, od získania a spracovania obrazu až po algoritmy ktoré dokážu vyhľadať požadované informácie v obraze.

Pre každú úlohu sa v prostredí Matlab, naprogramujú potrebné algoritmy a vytvorí aplikácia s grafickým užívateľským prostredím (GUI). Popíše sa fungovanie a práca v GUI a tak isto dôležité detaily implementovaných algoritmov, ktorých princíp sa ozrejmí v teórii.

Na záver bude experimentom overená funkčnosť a správnosť realizácie úloh a ich ďalšie možné využitie.

2 HARDWARE VÝBAVA

2.1 Vstavané systémy

Vstavané systémy, sú špecializované počítačové systémy, ktoré slúžia na jednu alebo viac špecializovaných funkcií. Často tieto systémy nachádzame v spotrebnej elektronike, v automatizácii, v priemyselných zariadeniach, automobiloch a podobne. Základnou konštrukčnou jednotkou ich elektronickej časti býva tzv. mikrokontrolér (MCU), čo je mikroprocesor doplnený o operačnú pamäť a množinu potrebných periférnych zariadení. Vstavané systémy sa začali objavovať v 60. rokoch 20. storočia. Prvý vstavaný systém bol použitý ako navádzací počítač pre vesmírny modul Apollo, vyvinutý Chalsom Stark Draperom. Pokles cien procesorov urýchlilo nahradzovanie analógovej elektroniky za digitálnu, a tak od konca 80. rokov sa vstavané systémy využívajú vo väčšine elektronických zariadení. Vstavané systémy sú v dnešnej dobe z veľkej časti plnohodnotné počítačové systémy, obsahujúce jeden alebo viac procesorov. Tieto systémy sú aj súčasťou segmetu Internet vecí(IoT), ktorý zažíva obrovský rozmach vďaka nim [1]. Medzi ich výhody patrí dostatočný výkon pri zachovaní minimálnych rozmerov, univerzálne využitie, nízka spotreba energie, spoľahlivosť a účinnosť systémov, a ich výrobné náklady. Naopak nevýhodami sú nemožnosť upgradu ich hardware, alebo voľba operačného systému, ktorá závisí na architektúre použitej v vstavanom systéme. Platformy sa viacmennej navzájom líšia výkonom a možnosti aplikácie, ktorá určuje jej cenu. Medzi lacné (60USD) vstavané systémy patria Arduino, Raspberry Pi, Espressif, alebo Orange Pi, naopak medzi drahšie (500USD a viac), NVIDIA Jetson alebo Microsoft Newton [1],[2].

2.1.1 Raspberry Pi

Miniaturizácia elektronických komponentov umožňuje situovať dostačujúci výpočetný výkon na ploche veľkosti zopár centimetrov. Tento technologický pokrok zosobňuje aj platforma Raspberry Pi (ďalej len Rpi). Ide o mini počítač veľkosti debetnej karty, ktorého prvý model bol uvedený na trh britskou spoločnosťou Raspberry Pi Foundation vo februári 2012. Dôraz sa kladie na dostačujúci výkon, možnosť pripojiť rôzne periférne zariadenia a zachovať nízku cenu platformy. Pôvodná myšlienka bola vytvoriť dostupný počítač pre potreby vzdelávania na školách. Rpi sa stalo veľmi rozšírenou a obľúbenou platformou. To dokazuje aj momentálne už tretia generácia platformy a obrovská komunita užívateľov po celom svete. Rpi preukazuje svoje prednosti hlavne v spojení s ďalšími prístrojmi. Jedny z mnohých už realizovaných projektov sú multimediálne centrum pre domácnosť, náhrada PC, PiTablet, PiPhone, TorRouter,3D tlačiareň, meteo stanica, rôzna robotika, quadrokoptéra atď [4].

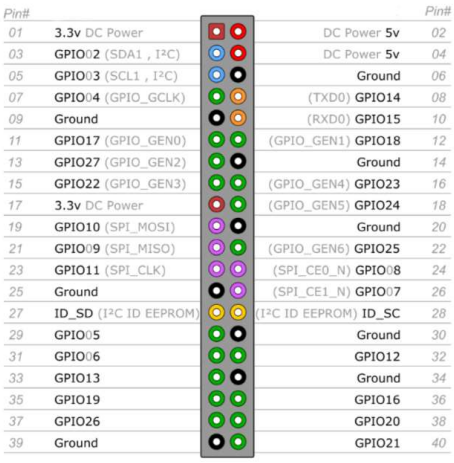
Prvá generácia Rpi bola uvedený v roku februáru 2012.Výrobca z počiatku vyrobil dve platformy Rpi 1 modely A a model B, ktoré sa líšili v špecifikácii a výkone. Procesor mal jedno jadrový ARM v6 s 700MHz a pamäť 256 MB RAM a osem GPIO pinov, model B mal pamäť 512MB RAM. Produkt bol mimoriadne úspešný, kvôli nízkej cene, ktorá sa pohybovala od 25 do 35 USD. Energetická efektivita a životnosť Rpi spravila ľahko modifikovateľnú platformu pre rôzne projekty. Raspberry Pi Foundation ponúka operačný systém Rasbian, ktorý je odvodená distribúcia Linux OS pre Rpi. Platforma podporuje programovanie v jazykoch Phyton, C/C++, PERL a iné. Z úvodných modelov sa predalo viac ako 4,5 milióna kusov a neskôr na základe tohto úspechu v roku 2014 prišlo ich zlepšenie v podobe Modelu A+ a Modelu B+. S novými modelmi došlo k rozšíreniu na 17 GPIO pinov a spotrebovali menej

energie na prevádzku. V roku 2015 bol vydaný Rpi 2 ktorý mal o 200 MHz rýchlejší procesor a zdvojnásobila sa pamäť RAM. Pre embeded aplikácia vyšla verzia Rpi Zero, u ktorej boli zredukované rozmery, vstupy-výstupy, GPIO piny a uvádzacia cena bola 5 USD. Posledný a aktuálny model v čase písania tejto diplomovej práce je model Rpi 3 [3],[4].

V diplomovej práci je použitá verzia Raspberry Pi 3 model B (obrázok Obr. 2). Bol vydaný vo februári 2016 a nadväzuje na model Raspberry Pi 2, ktorý bol vydaný rok predtým. Je vylepšený o 64-bitový procesor a taktovacia rýchlosť procesora sa z výšila z 900 MHz na 1,2 GHz. Naviac platforma obsahuje bezdrôtové pripojenie 802.11n Wireless LAN a Bluetooth 4.1 a Bluetooth Low Energy (BLE) [6].

Obr. 2) Platforma Raspberry Pi 3 model B

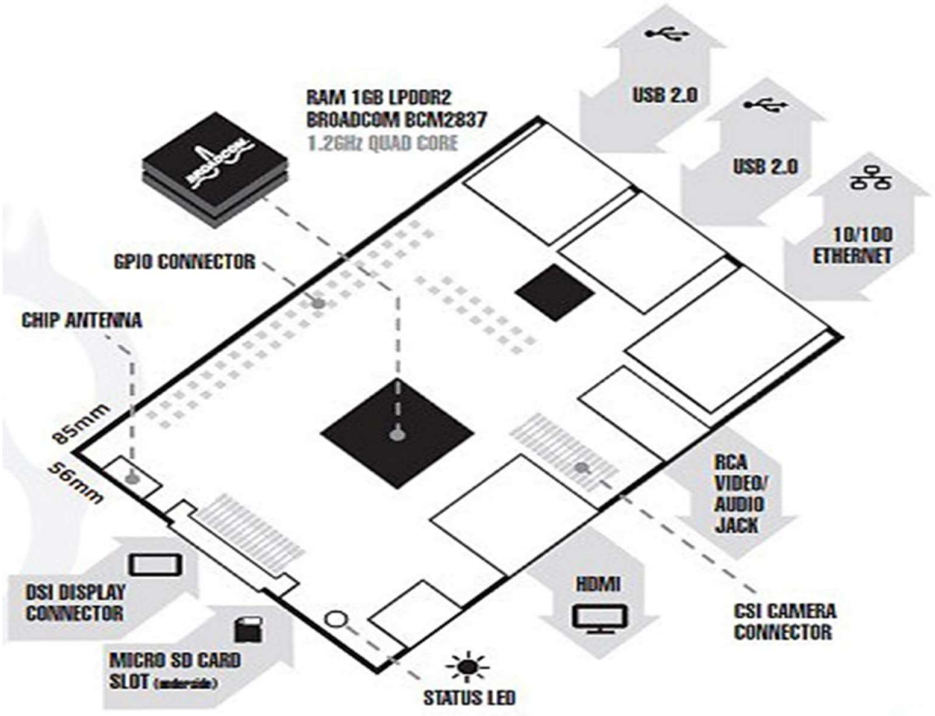
Konektorová výbava a situovanie sa nezmenilo. Rozloženie komponentov je znázornené na obrázku Obr. 4). Pre pripojenie kamery je slúži port CSI camera interface. LCD displej je možné pripojiť do DSI display interface portu. Monitor je možné pripojiť cez HDMI. Ďalej sú k dispozícii 4 USB 2.0, RJ45, 40-pinový GPIO konektor a micro USB port určený na napájanie. GPIO (z anglického *general purpose input/output*, viacúčelové rozhranie pre vstup/výstup) konektor je silná stránka Rpi pretože umožňuje pripojiť rôzne rozširujúce moduly, čidlá, servo motory a rôzne iné zariadenie. Konektor obsahuje 26 pinov a u 17 z nich je možné užívateľsky meniť vstup/výstup. Dva piny UART, dva pre I2C a ďalších päť pre SPI. Ostatné 4 piny pre napájanie 3,3V alebo 5V a šesť uzemňovacie piny. Rozmiestnenie pinou je na obrázku Obr. 3). Rozloženie GPIO pinou je korešponduje s predošlými verziami Rpi 2B,1B+ a 1A+ a takto je zaistená kompatibilita s rozširujúcimi modulmi [6],[7].



Obr. 3) GPIO piny platformy Raspberry Pi 3 [7]

Technické parametre [7][6]:

- **Architektúra:** ARM Cortex A53 CPU (64-bit)
- **Platforma:** Broadcom BCM2837
- **Grafický čip:** Broadcom VideoCore IV @ 250 MHz, OpenGL ES 2.0 MPEG-2 a VC-1, 1080p30 H.264 / dekodér a enkodér vysoko profilového MPEG-4 AVC
- **Operačná pamäť :**1GB SDRAM
- **Úložný priestor:** MicroSD slot
- **Operačný systém:** bootovatelni z micro SD karty, distribúcie Linux, aj Win 10 IoT.
- **Vstupy/Výstupy:** 4xUSB 2.0, HDMI port, 100Mbit Ethernet port, 40 GPIO piny, kombinovaný 3,5 audio jack a kompozitný video výstup, CSI- rozhranie pre pripojenie kamery schopný viesť vysoký tok dát, DSI- rozhranie pre pripojenie LCD displeja
- **Bezdrôtové pripojenia:** 802.11n Wireless LAN, Bluetooth 4.1
- **Rozmery:** 85 mm x 56 mm
- **Napájanie:** micro USB, 5V/1A



Obr. 4) Rozloženie komponentov na platforme [13]

2.2 Servo pohon

Pre rotáciu kamery vo vertikálnom a horizontálnom smere zabezpečuje dvojica servo motorov. Použité sú serva s označením SG90 9g Micro Servo od výrobcu Tower Pro Obr. 5) Servo je napájane jednosmerným napätím a rotáciu serva ovláda pulzne šírkovo modulovaný signál (PWM). Rozsah rotácie je 180°(90° kladným aj záporným smerom). Konektor je trojžilový, ktorý sa pripája priamo na GPIO piny Raspberry Pi. Vodiče sú rôznej farby, na ktoré sa privádza, čierny vodič zem(-), červený vodič (+) a oranžový vodič signál. Pozíciu „0“ (stred) reprezentuje pulzný signál o šírke (1,5 ms), pozícia „90“ (~2 ms) je otočenie v zápornom smere a pozícia „90“ (~2 ms) je otočenie v kladnom smere. Súčasť servo pohonu je aj vstavaná prevodovka, ktorá poskytuje dostatočný krútiaci moment a unášač umožňujúci jednoduchú inštaláciu [8],[9].

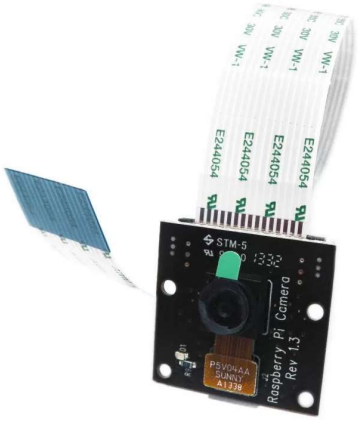
Technické parametre [8]:

- **Hmotnosť:** 9g
- **Rozmery:** 22,2 x11,8 x31 mm
- **Pracovné napätie:** 4,8 V (~5V)
- **Krútiaci moment:**1,6 kg/cm
- **Rýchlosť rotácie:** 0,1s/60°
- **Pracovná teplota:** -30 C° až 60 C°

Obr. 5) Servo pohon a popis jehočasti

2.3 Kamera

Pi NoIR v1.3 je oficiálny kamerový modul pre Raspberry Pi Obr. 6).V module je osadený senzor s rozlíšením 5 Mpx (2592×1944), s podporou videa v 1080p pri 30 fps, 720p 60 fps a 640x480p 60/90 fps. NoIR znamená že kamera nemá infračervený filter. Obraz získaný za denného svetla vyzerá trocha „dočervena“. Získava však možnosť nočného videnia pokiaľ je použité infračervené osvetlenie. Modul sa pripojuje k Raspberry Pi 15-pinovým flex káblom pomocou CSI portu za Ethernet port. Modul je spätne kompatibilný s predošlými modelmi. Malé rozmery a nízka hmotnosť zjednodušuje aplikáciu. Modul je osadený v navrhnutom držiaku [5].



Obr. 6) Kamerový modul Pi NoIR v.1.3

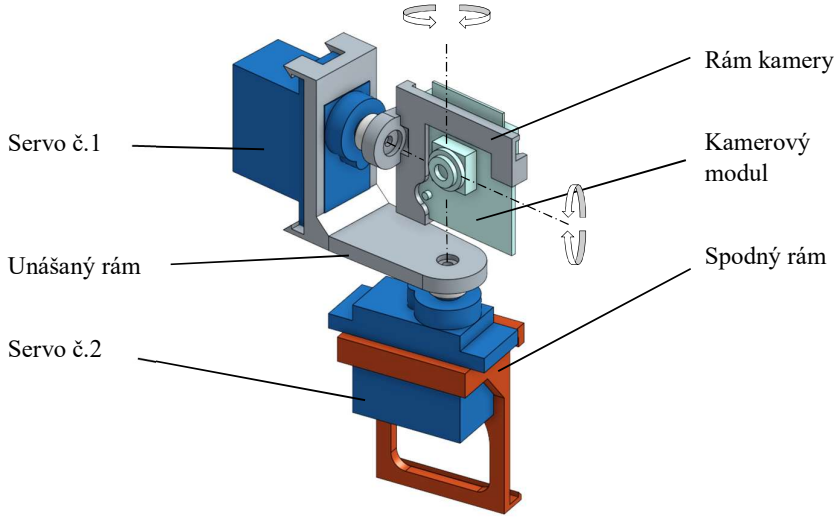
Technické parametre [5]:

- **Senzor:** CMOS OmniVision OV5647 Color CMOS QSXGA
- **Rozlíšenie:** 5-megapixlov
- **Počet pixelov:** 2592×1944
- **Veľkosť pixelu:** 1,4×1,4 µm
- **Veľkosť šošovky:** ¼“
- **Ohnisková vzdialenosť:** 3,6 mm

- **Fixný fokus:** 1 m do nekonečna
- **Zorný uhol:** 53,5° horizontálne 41,41° vertikálne

2.4 Držiak kamery

Držiak kamery je navrhnutý, tak aby vertikálna a horizontálna os otáčania kamery mali prienik v objektíve. Ako vhodný návrh držiaku je model 3 rámov podľa obrázka Obr. 7). Model držiaku bol vytvorený v cloud CAD OnShape. Návrh využíva možnosť upevniť kamerový modul a servo pohony tak aby pripájacie kábli neboli nevhodne namáhané v celom rozsahu pohybu, nevyžadovali predlžovacie vodiče a naviac umožňuje praktické prichytenie na ochranný obal Raspberry Pi a tak zabezpečiť manipuláciu a stabilitu zostavy počas otáčania servami. Kamerový modul je osadený v ráme kamery, ktorý sa pripája na unášač serva č.1 a umožňuje natáčať kameru podľa horizontálnej osi. Servo č.1 je fixované na unášaný rám pripojený na unášač serva č.2, a tak je zabezpečené otáčanie okolo horizontálnej osi. Následne je servo č.2 usadené do spodného rámu napojenom na ochranný obal Raspberry Pi.



Obr. 7) 3D model rámu držiaku kamery

3 SOFTWAROVÁ VÝBAVA

Ako softwarová výbava je použitý Matlab a jeho toolboxy. V nasledujúcich podkapitolách je popísaný, samotný software Matlab, jeho úžívateľské prostredie, použité toolboxy, jeho nastavenie tak aby spolupracoval s Raspberry Pi a export naprogramovaných súborov Matlab.

3.1 Matlab

Matlab alebo MatrixLaboratory je, dnes považovaný za komplexný výpočtový systém s veľkým množstvom funkcií, ktorý je produktom americkej spoločnosti The MathWorks. Funkcie a možnosti softwaru je možné rozširovať pomocou ďalších toolboxov, ako aj špecializovaných nadstavieb pre simuláciu dynamických systémov Simulink, alebo udalostných systémov Stateflow [11]. Používa vlastný objektovo orientovaný programovací jazyk blízky jazyku C. Jazyk je takzvaný „matrix-based“. Vo voľnom preklade znamená, že na výpočty sú predovšetkým používané matice ako cesta prirodzeného spôsobu vyjadrovania výpočtovej matematiky. Je často využívaný v oblastiach výskumu, akademickej aj súkromnej sfére inžinierov, vedcov či ekonómov. Umožňuje užívateľom spracovať a analyzovať dáta, pomocou grafov, tabuliek, algoritmov a to poskytuje vyššiu úroveň matematickej presnosti pre výpočty. Matlab ponúka riešenia v oblastiach ako je aplikovaná matematika, strojové učenie, spracovanie signálu a komunikácie, spracovanie obrazu a počítačové videnie, finančná analýza a modelovanie, návrh riadiacich systémov, robotika a mnoho ďalších oblastí [12],[13] .

Obr. 8) Logo spoločnosti MathWorks [12]

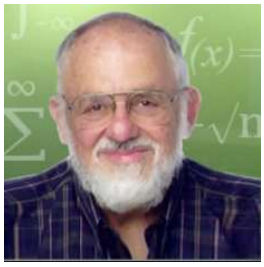
Hlavné výhody Matlabu [12]:

- Matlab “rozpráva matematikou“- umožňuje priamo zadávať matice. Lineárna algebra v programe vyzerá ako algebra v učebnici.
- Je navrhnutý pre inžinierov a vedcov -názvoslovie funkcií je ľahko zapamätateľné. Prostredie je navrhnuté na opakované pracovné postupy a dokumentácia je zrozumiteľne napísaná pre inžinierov a vedcov, ktorí sa nevenujú informačným technológiám.
- Množstvo toolboxov- obrovská ponuka profesionálne vyvíjaných, testovaných nástrojov, ktoré majú širokú pôsobnosť vo vedeckých a inžinierskych aplikáciách. Dokážu pracovať navzájom a integrovať do paralelne výpočtového prostredia, a kódmi v jazyku C.
- Matlab „appky“- sú interaktívne aplikácie, ktoré kombinujú priami prístup do veľkej kolekcie algoritmov s okamžitou vizuálnym zobrazením ako rôzne algoritmy pracujú s dátami. Opakované kroky pri práci je matlab schopný generovať automatizovanie úkonov.
- Integrácia pracovného procesu- matlab zvláda všetky kroky od myšlienky až k implementácii. Program sa dokáže spojiť s viac ako tisíc rôznymi hardware zariadeniami. Podpora analýzy, cloud computing, simulácii a tak isto aj implementácia Matlab kódu do C alebo HDL do produktu.

- Rýchlosť- Matlab kód je vysoko optimalizovaný a tak dokáže využiť celý vypočetný výkon multijadrových procesorov.

3.2 História Matlab

Svoje počiatky datuje do 50-tych rokov 20.storočia, kedy Cleve Moler (na obrázku Obr. 9), vtedy študent Caltechu (Kalifomský inštitút technológie), spolu s jeho profesorom Johnom Toddom pracovali na algoritmizácii úloh pre počítacie stroje ako elektrónkový počítač Burroughs 205 Datatron. V roku 1962 naprogramoval Moler vo Fortrane program na riešenie systému lineárnych rovníc, ktorý bol uložený v podobe diernych štítkov. Neskôr v postgraduálnom štúdiu u profesora Forsythea kde svojej kandidátskej práce vykonával experimenty na tzv. L-funkcii. Táto funkcia sa stala logom spoločnosti The MathWorks Obr. 8). Neskôr po obhájení práce vydáva s profesorom Forsythea vydáva knihu algoritmov počítania s maticami. [10]



Obr. 9) Otec matlabu Cleve Moler [16]

V roku 1971 vychádza kniha o rôznych aspektoch počítania s maticami od profesora Wilkinsona, na základe ktorej, vznikli knižnice funkcií vo Fortrane pre výpočet vlastných čísel matíc EISPACK. Neskôr nasledovala knižnica LINPACK, ktorá sa zamerala na riešenie sústav lineárnych rovnic. Vtedy Cleve Moler bol profesorom na univerzite v Novom Mexiku, kde prednášal numerickú analýzu a teóriu matíc. Vtedy naprogramoval špeciálne rozhranie vo Fortrane pre knižnice EISPACK a LINPACK pretože práca s nimi bola príliš komplikovaná pre študentov. Vtedy vznikol Matlab 1.0 a obsahoval s 80 funkcií, ktoré sú v Matlabe dodnes. Program bol veľmi obľúbený medzi študentami pretože umožňoval jednoducho spracovávať matice a tak pri riešení technických problémov odbremeniť od programovania algoritmov pre matice. Táto myšlienka je dodnes filozofiou spoločnosti The MathWorks. Matlab 1.0 bežal na sálových počítačoch a obsahoval aj jednoduché grafické funkcie.

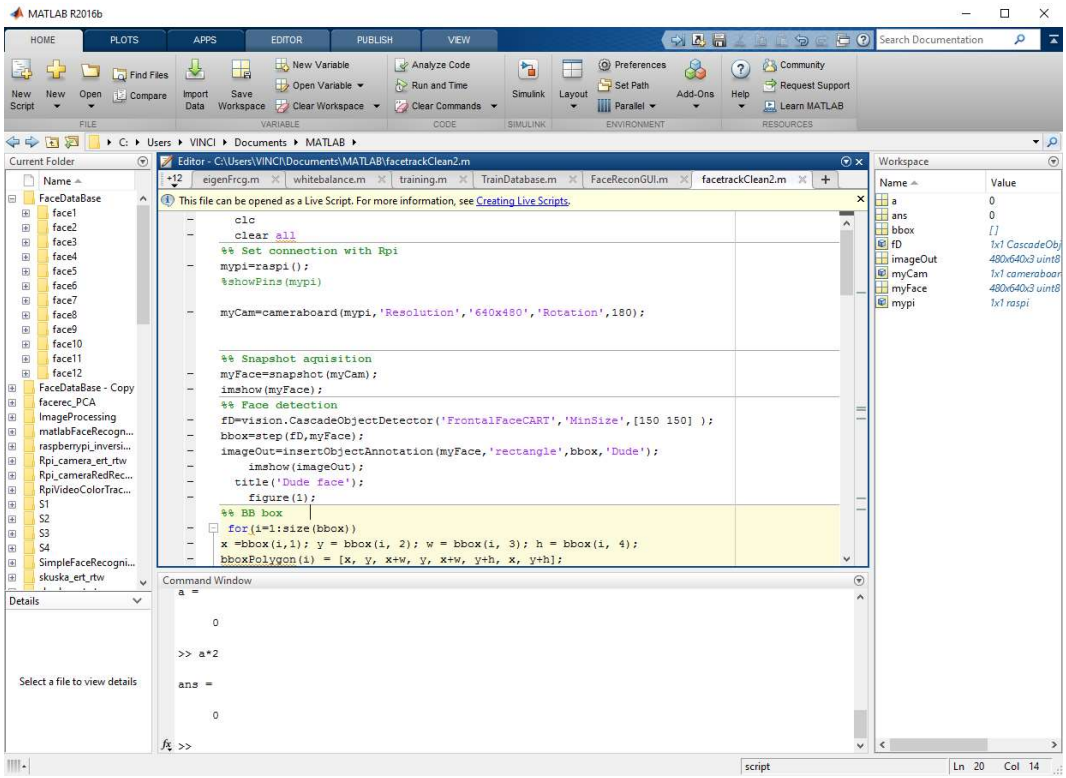
V roku 1978 Moler upravil program pre osobný počítač Tektronix 4081. Išlo o prvú implementáciu Matlabu pre osobné počítače. Program využívalo niekoľko Molerových študentov, ktorí ho používali pri technických výpočtoch a postupne rozširovali potenciál programu. Jednými zo študentov boli Jack Little a Steve Bangert odhadli možnosti vtedy uvedenej platformy IBM, ktorá uviedla v roku 1981 počítač typu PC a preprogramovali Matlab z Fortranu do C jazyka. Postupným rozširovaním funkciality pomocov .m súborov sa Matlab ukázal ako nástroj s širokým záberom využitia a tak v roku 1984 Moler, Little a Bengert založili spoločnosť The MathWorks, ktorá uviedla komerčnú verziu Matlabu. [10]

3.3 Popis užívateľského prostredia Matlab

Matlab je podporovaný na operačných systémoch Windows, MacOS a tak isto aj Unix. Program sa rok čo rok vylepšuje je obohacuje o nové funkcie a užívateľské prostredie sa každou novou verziou mení len mierne. Naviac je možné jednotlivé okna rôzne presúvať a preskupovať a tak vytvoriť na mieru komfortné prostredie pre prácu. Prostredie je podobné ako u rôznych programov, ovládanie je intuitívne a tak netrvá dlho, kým si užívateľ osvojí prácu v programe. V tejto diplomovej práci je použitá verzia Matlab 2016b.

Užívateľské prostredie je na obrázku Obr. 10) a pozostáva z hlavných časti:

- Lišta ponuky zo záložkami
- Editor
- Prikazové okno (Command Window)
- História príkazov (Command History)
- Pracovný priestor (Workspace)
- Aktuálny adresát (Current Folder)



Obr. 10) Užívateľské prostredie Matlabu

Príkazové okno (Command Window) je okno do, ktorého sa priamo zadávajú príkazy alebo spúšťajú skripty. Príkazy sa potvrdzujú tlačidlom Enter. Po spracovaní príkazu Matlab zobrazí výsledok hneď do okna alebo do dialógového okna v prípade grafických dát. Inicializované premenné sa spolu s hodnotami zapíšu do *pracovného priestoru (Workspace)* a zoradia v abecednom poradí. Takto ma užívateľ výborný prehľad ako sa spracúvajú dáta

a prípadne kontrolovať správnosť výpočtu. Matlab umožňuje vytvoriť súvislý súbor príkazov ktoré môže neskôr spracovať ako jeden celok alebo po sekciách. V *Hlavnej lište* v záložke *Home* je ikona *New Script* kde po kliknutí sa otvorí nový skript v okne *Editor.* Súbor pred spracovaním je potrebné uložiť v primárnej zložke *MATLAB,* ktorá je umiestnená v zložke *Dokumenty* užívateľa počítača.(Platí pre operačný systém Windows.) Zložku pre ukladanie súborov je možné jednoducho meniť v okne *Aktuálneho adresáta (Current Folder).* Skript má charakteristickú príponu *.m* a tak je jednoduché ho identifikovať. V okne *Editor* je možné otvoriť, niekoľko skriptov a priamo v okne *Editor* rozložiť, napríklad vedľa seba. Je možné vyberať a spúšťať príkazy kombinované zo scriptov. Matlab umožňuje spustiť kód riadok po riadku, zo skriptu alebo zadávať súčasne z príkazového okna. Vďaka tomu Matlab umožňuje rýchle riešenie prípadných chýb v kóde a viackrokové operácie sú priehľadnejšie.

3.4 Podporné balíky

Podporné balíky rozširujú použiteľnosť Matlabu a tým ho robia robustným nástrojom. Pri realizácii úloh diplomovej práce boli použité balíky popísane v nasledujúcich podkapitolách.

3.4.1 Podporný balík pre Raspberry Pi

Pre prácu s Raspberry Pi MathWorks vytvoril špecializovaný podporný balík nástrojov s názvom Raspberry Pi Support from MATLAB/Simulink. Umožňuje komunikovať s Raspberry Pi, ovládať a získavať dáta z periférnych zariadení pripojených k Raspberry Pi. Balík obsahuje funkcie schopné komunikovať prídavkami a rozhraniami ako Raspberry Pi Sense HAT (prídavná doska, ktorá bola špeciálne vyrobená pre Astro Pi misiu k medzinárodnej vesmírnej stanici v roku 2015 obsahuje 8x8 RGB maticu, tlačidlá, joystik, gyroskop, akcelerometer, magnetometer ,snímač teploty, barometrického tlaku a vlhkosti), Raspberry Pi Camera Board (oficiálny kamerový modul výrobcu), I2C, SPI a Serial rozhranie, podpora pulzne-širkovej modulácie a riadenie servo pohonu, ovládanie GPIO a Linux system shell. [14]

3.4.2 Balík spracovania obrazu a počítačového videnia Matlab

Pre potreby spracovanie obrazu a počítačového videnia sú dostupné „open souce“ knižnice OpenCV. OpenCV podporuje programovacie jazyky ako C/C++, Python, Java a je dostupná pre platformy Windows, Linux, MacOS a Android. Matlab ponúka však vlastný balík nástrojov s názvom *Image proccessing and Computer Vision System Toolbox*. Balík poskytuje algoritmy, funkcie a aplikácie na navrhovanie a simuláciu systémov počítačového videnia a spracovania obrazu. Umožňuje vykonať detekciu, extrakciu a porovnávanie charakteristických prvkov, identifikáciu a sledovanie objektov, odhad pohybu a spracovanie obrazu. Pre trojrozmerné počítačové videnie, systémová sada nástrojov podporuje kalibráciu kamier, stereo kamier, 3-D rekonštrukciu a 3-D bodové cloud spracovanie. S algoritmami založenými na strojovom učení umožňuje trénovať detekciu objektov, rozpoznávanie tvaru objektov a systémy na vyhľadávanie snímok [17].

3.4.3 Inštalácia podporného balíka pre Raspberry Pi

Balík je možné stiahnuť samostatne zo stránky (https://www.mathworks.com/hardware-support/raspberry-pi-matlab.html) alebo priamo v programe. Kliknutím v hlavnej lište programu na ikonu *Add-Ons* následne po vysunutí lišty možnosť *Stiahnuť hardware balík*

podpory (Get Hardware support package). Umiestnenie ikony Add-Ons na hlavnej lište je na obrázku Obr. 11). Následne sa otvorí okno, ktoré ponúka je na výber inštaláciu z internetu alebo inštaláciu už siahnutého balíka. Súčasť inštalácie balíčka je aj nastavenie Rpi a inštalácia potrebných súborov pre úspešnú komunikáciu s Matlabom.

Obr. 11) Umiestnenie ikony Add-Ons v hlavnej lište Matlabu

Najprv je potrebné pripojiť Rpi k host počítači cez Ethernet kábel a po detekcii zariadenia inštalácia pokračuje ďalším krokom. Matlab si vyžiada vloženie mikro SD karty s veľkosťou pamäte aspoň 8GB na inštaláciu potrebných súborov. Po dokončení, SD kartu vložíme do Rpi a nasleduje konfigurácia pripojenia. Matlab automaticky doplní informácie o pripojení a tak ostáva skontrolovať IP adresu, Host name, užívateľské meno a heslo. Ukážka nastavenia na obrázku (Obr. 12). Nastavenie je pôvodné od výrobcu a možné prenastaviť. Tlačidlom *Test connection* otestujeme pripojenie. Po úspešnej inštalácii je Rpi pripravené a Matlab je schopný spolupracovať so všetkými rozhraniami Rpi bez nutnosti inštalácii ďalších knižníc a doplnkov.

Obr. 12) Ukážka nastavenia počas inštalácie podporného balíka pre Raspberry Pi

Podobný postup je aj pri inštalácii balíka spracovania obrazu a počítačového videnia MATLAB

3.5 Kompilácia samostatne spustiteľnej aplikácie

Matlab poskytuje vlastný kompilátor *MATLAB Compiler,* ktorým je možné programy naprogramované v Matlab, spustiť ako samostatnú aplikáciu aj na počítači, na ktorom nie je nainštalovaný Matlab. Aplikácie vytvorené pomocou kompiléru vyžadujú *MATLAB* Compiler Runtime, ktorý je poskytovaný ako voľná distribúcia pre užívateľov, ktorý Matlab nemajú nainštalovaný. Takto boli prevedené aj aplikácie naprogramované pri realizácii úloh diplomovej práce [18].

3.6 Základné príkazy Matlab pre prácu s Raspberry Pi

V tejto kapitole sú popísané základné príkazy s ktorými sa užívateľ stretne pri programovaní v Matlab pre Raspberry Pi. Ide o základné príkazy pripojenia k Raspberry Pi, ovládanie kamerového modulu a ovládania seriev.

Vytvorenie komunikácie s Raspberry Pi

Vytvorenie komunikácie s Rpi sa uskutočňuje pomocou nasledujúcich príkazov. Vo funkcii *raspi* sú za argumenty *ipaddress*- IP adresa zariadenia Rpi, *username*-prihlasovacie meno a *password*- prihlasovacie heslo. Tieto argumenty nie je potrebné zadávať pokiaľ nepracujeme s viacerými platformami Rpi naraz. Argumenty si nahraje Matlab z nastavení, ktoré boli uložené potom ako bol nainštalovaný podporný balíček pre Rpi.

```
%Vytvorenie komunikácie s Rpi
mypi = raspi(ipaddress,username,password)
```

Ukončenie komunikácie s Rpi sa realizuje príkazom *clear mypi.*

```
%Ukončenie komunikácie s Rpi
clear mypi;
```

Ovládanie kamery

Spustenie kamery sa realizuje príkazom *cameraraboad(mypi)*.Kde za argument *mypi* sa doplní vytvorený objekt komunikácie s Rpi. Ďalšie vlastnosti je možné špecifikovať pridávaním ďalších argumentov. Celý zoznam vlastností je prístupný ak sa do príkazového riadna napíše vytvorený objekt komunikácie s Rpi bez bodkočiarky a potvrdí tlačidlom *Enter*. V tomto prípade je špecifikované rozlíšenie *(Resolution)* a rotácia obrazu *(Rotation)*.

```
%Spustenie kamery
myCam=cameraboard(mypi,'Resolution',‘800x600’,'Rotation',180);
```

Získanie obrazu z kamery sa uskutočnuje príkazom *snapshot(myCam)*.

```
%Ziskanie aktuálneho obrazu
frame=snapshot(myCam);
```

Ak príkaz *snapshot* vložíme do slučky získame prenos obrazu z kamery do PC. Funkcia *imagecs* vykreslí aktuálnu snímku na obrazovku PC.

```
%Spustenie obrazového prenosu
while 1
frame = snapshot(mycam)
        imagesc(frame)
        drawnow
end
```

Ovládanie servo pohonov

Natočenie serva sa ovláda šírkovo modulovaným signálom (PWM). Signál je produkovaný softwarovo priamo platformou Raspberry Pi(Rpi). Súčasťou podporného balíka pre Rpi sú funkcie určené na ovládanie PWM signálu.

Pripojenie sa uskutočňuje príkazom *servo(mypi,pin)*, ktorého argumenty sú *mypi*(vytvorený objekt Rpi) a *pin* (GPIO pin platformy, na ktorý PWM signál budeme privádzať).Takto vytvoríme objekt *horS,* ktorý reprezentuje servo pripojene k Rpi. Ďalšie vlastnosti sú pridané minimálna a maximálna dĺžka PWM signálu podľa technických parametrov servopohonu. V tomto prípade ovládame servo pripojené na GPIO pin 18 a špecifikujeme dĺžky PWM signálu. Minimálny *(MinPulseDuration)* na 0,55 ms a maximálny *(MaxPulseDuration)* na 2,2 ms.

```
%Ovládanie serva
horS = servo(mypi,18,'MinPulseDuration',0.55e-3,'MaxPulseDuration',2.20e-3);
```

Otočenie serva o požadovaný uhol realizujeme príkazom *writePosition(s, position).* Kde *s* je objekt servopohonu a *position* špecifikuje natočenie serva. V tomto prípade požadujeme natočenie 90 stupňov. To reprezentuje strednú polohu serva, od ktorej je možne servo natáčať v kladnom alebo zápornom smere.

```
%Prikaz natočenia serva o daný uhol
writePosition(horS,90)
```

4 POČITAČOVÉ VIDENIE

Keď vychádzame zo všeobecnej definície počítačového videnia ako „vednej disciplíny, ktorá sa snaží počítačovými prostriedkami napodobniť ľudské videnie“, tak je zrejmá jej prirodzená väzba na príbuzné vedné disciplíny. Základom je spracovanie 2D obrazu (Image Processing). Túto etapu väčšinou predstavujú algoritmy na nízkej úrovni abstrakcie, kým vyššiu úroveň predstavujú algoritmy, ktoré extrahujú z nižšej úrovne znalosti, dávajú ich do súvislostí a snažia sa im porozumieť (Image understanding, machine learning). To by nebolo možné bez znalosti fyzikálnej podstaty senzorov, optiky, okolitého prostredia a tiež matematických a štatistických metód na ich hodnotenie [19],[20].

Celý proces spracovania a rozpoznávania obrazu reálneho sveta môžeme rozdeliť do niekoľkých základných krokov. Rozdelenie procesu spracovania obrazu môže byť v každej literatúre iné a tak nie úplne jednoznačné. Avšak uplatnenie všetkých krokov v poradí, záleží až na konkrétnej aplikácii. V nasledujúcich kapitolách sa bližšie popíše hlavná myšlienka jednotlivých krokov [21].

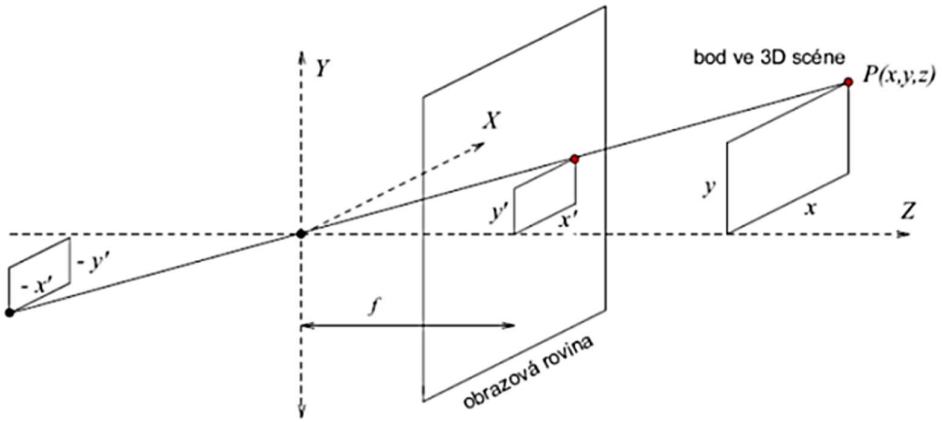
Všeobecná postupnosť základných krokov procesu spracovania obrazu je:

- Snímanie obrazu
- Digitalizácia
- Predspracovanie obrazu
- Segmentácia
- Klasifikácia

4.1 Snímanie obrazu

Je prvou etapou zberu obrazu a jej cieľom je previesť optické veličiny na elektrické. Tejto etape je treba venovať veľkú pozornosť. Chyby spôsobené nekvalitnou kamerou alebo zlým usporiadaním snímacieho reťazca je potom veľmi ťažké korigovať. Vstupnou veličinou pri snímaní nemusí byť len jas z kamery, ale môžu byť aj iné veličiny ako sú intenzita röntgenového žiarenia, tepelná intenzita alebo ultrafialové žiarenie [20].

Z geometrického hľadiska je možné snímanie obrazu definovať ako prevod trojrozmerného (3D) obrazu sveta okolo nás do obrazu dvoch rozmerov (2D). Toto prevedenie je výsledok perspektívneho zobrazenia. Snímanie obrazu modeluje najjednoduchšia kamera nazývaná tiež dierková kamera. Je založená na reálnom fyzikálnom zariadení schopného získať obraz [21]. Geometria perspektívneho zobrazenia je na obrázku Obr. 13).

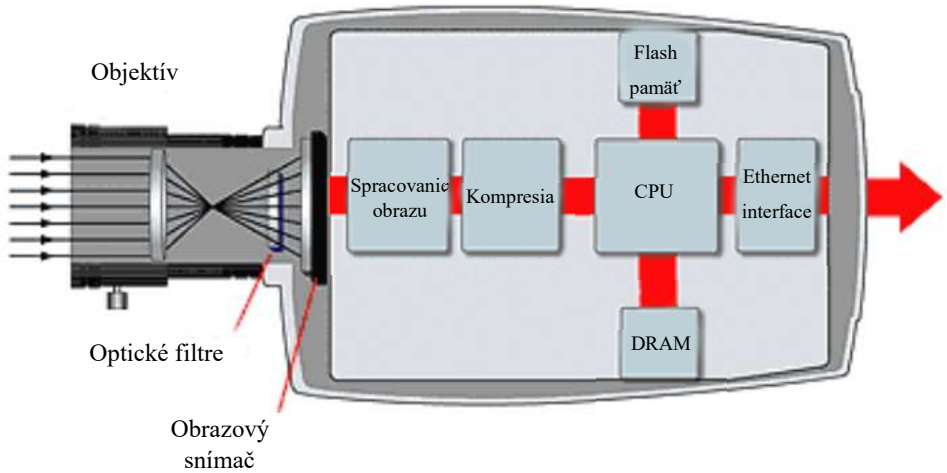


Obr. 13) Geometria perspektívneho premietania [21]

Obraz sa premieta do obrazovej roviny vzdialenej od dierky o ohniskovú vzdialenosť f. Súradnice bodu P v 3D (x, y, z) sú premietané do 2D obrazovej roviny kde má súradnice (x', y'). Vzdialenosť obrazovej roviny od stredu premietania je f [21]. Potom pre perspektívne zobrazenie z podobnosti trojuholníkov vyplývajú rovnice (19).

$$x' = \frac{xf}{z}, \qquad y' = \frac{yf}{z} \tag{1}$$

Snímaný obraz ovplyvňuje mnoho faktorov. Tieto faktory môžu byť napríklad osvetlenie snímaného objektu a jeho vlastnosti, tvar, atď. Znalosť týchto vlastností nám neskôr môže pomôcť pri spätnej rekonštrukcii 3D obrazu z nasnímanej scény. Aby sme mohli vykonať snímanie jasu z kamery a previesť tento jas na elektrický signál, musíme použiť svetlocitlivé snímače dopadajúceho svetelného žiarenia [20],[21]. Uloženie obrazového snímača v tele digitálnej kamery je znázornené na obrázku Obr. 14)



Obr. 14) Uloženie obrazového snímača v tele digitálnej kamery [26]

V súčasnosti sa používajú snímače typu:

- CCD (Chared Coupled Device)
- CMOS (Complementary Metal Oxide Semiconductor)

CCD

Objav CCD snímačov sa datuje do roku 1969. Každý CCD snímač sa skladá z veľkého počtu svetlocitlivých buniek, ktoré pri reakcii so svetlom vytvárajú elektrický náboj. Čím viac svetla na snímač dopadne, tým väčší náboj vznikne. Vďaka tomu sa v pripojených kondenzátoroch nahromadí energia, ktorá je úmerná dopadajúcemu žiareniu. CCD čip potom v podstate funguje ako analógový posuvný register. Každý kondenzátor odovzdáva svoj získaný náboj do susedného kondenzátora. Posledný z kondenzátorov je napojený na výstupný zosilňovač, ktorý náboj prevádza na elektrické napätie a pomocou A/D prevodníka, prevedené do digitálnej podoby. Hlavnou výhodou CCD snímačov je predovšetkým ich vysoká citlivosť a v porovnaní s CMOS tiež malý šum. Ako nevýhody môžeme spomenúť možnosť vzájomného ovplyvňovania pixelov, malý rozsah intenzít a tiež nemožnosť adresovať jednotlivé pixely. CCD čipy sú hlavne využívané v televíznych kamerách [22],[21].

CMOS

Každý pixel CMOS čipu obsahuje fotodiódu, ktorá je napojená na tzv. aktívny tranzistor. Podľa veľkosti dopadajúceho žiarenie sa na tranzistore nahromadí elektrická energia. Tranzistor je napojený na čítací a resetovací obvod. CMOS snímač tvorí matica takýchto detektorov. Vďaka technologickému pokroku sa CMOS snímač kvalitou obrazu priblížili CCD snímačom, ale stále nie sú vhodné pre kamery, od ktorých požadujeme najvyššiu možnú kvalitu obrazu. Výhodou CMOS snímača sú nízke náklady a jednoduchá technológia výroby. Oproti CCD majú nízku spotrebu a umožňujú vytvoriť snímač o malých rozmeroch. CMOS čipy majú väčší rozsah intenzít a veľkú rýchlosť čítania. Nevýhodou môže byť zlá citlivosť na svetlo, ktorá ešte stále predstavuje obmedzenie pre využitie CMOS snímačov. Táto nevýhoda nie je problém ak bude kamera umiestnená v dobre osvetlenom prostredí, ale u zle osvetleného prostredia, môže byť rozdiel v kvalite obrazu zreteľný. Výsledkom je veľmi tmavý obraz plný šumu. CMOS čipy sú hlavné rozšírene vo fotomobiloch [22],[21].

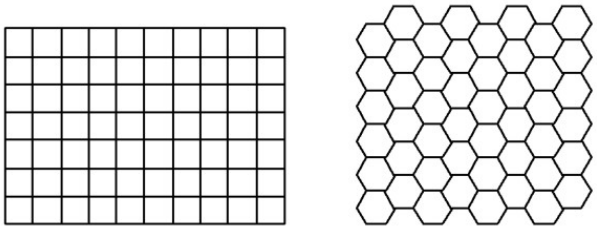
4.2 Digitalizácia obrazu

Snímače obrazu vo väčšine poskytujú na výstupe obrazovú funkciu v spojitom tvare, čo znamená spojitosť v definičnom obore i v obore funkčných hodnôt. Jej spracovanie na číslicovom počítači vyžaduje prevod do diskrétnej podoby, čiže digitalizáciu. Digitalizácia pozostáva zo vzorkovania obrazu v matici MxN bodov a kvavntovania jasovej úrovne [20].

4.2.1 Vzorkovanie

Pre vzorkovanie platí, že čím vyššia je hustota vzorkovania, tým bude diskrétna funkcia aproximovať spojitú funkciu. Pri vzorkovaní sa berie ohľad na charakter spracovania obrazu. Pri vzorkovaní môže dôjsť k strate informácii napríklad stroboskopický efekt. Preto je potrebné prispôsobiť tomu aj vzorkovaciu frekvenciu tak aby spĺňala známy Shanonov teorém. Ten hovorí, že vzorkovacia frekvencia musí byť aspoň dvakrát väčšia ako je maximálna frekvencia, ktorú obsahuje vzorkovací signál. V oblasti spracovania obrazu Shanonov teorém znamená, že

interval vzorkovania musí byť menší alebo rovný polovici rozmeru najmenších detailov na obraze ktorý chceme ešte vidieť [20]. Voľba rozlíšenia obrazu je preto najzásadnejší krok, kedy pri nízkom rozlíšení strácame informácie o detailoch a pri vysokom sa zvyšuje výpočetná náročnosť. Elementárne body, ktoré získame vzorkovaním nazývame pixely. Tie môžeme usporiadať do vzorkovacej mriežky [21]. Najčastejšie sa požívajú mriežky štvorcové a hexagonálne ktoré sú na obrázku Obr. 15)



Obr. 15) Obrázok štvorcovej a hexagonálnej mriežky [21]

Výhoda hexagonálnej mriežky oproti štvorcovej je, že vzdialenosť pixelu od všetkých susedných pixelov je rovnaká. Štvorcová mriežka vychádza z konštrukcie väčšiny snímacích prvkov a je ľahko realizovateľná. Napriek tomu dochádza k problémom pri meraní vzdialenosti a spojitosti pixelov. Tento problém nie je prítomný pri použití hexagonálnej mriežky, pri ktorej naopak vznikajú problémy pri niektorých operáciách ako je Fourierova transformácia. Používa sa pri rôznych filtroch obrazu [20],[21].

Medzi dôležité pojmi a vlastnosti patrí vzdialenosť a susedstvo(okolie).

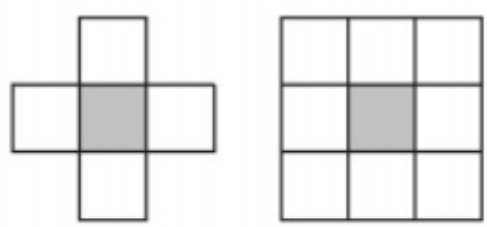
Vzdialenosť predstavuje v usporiadanej mriežke sa vzdialenosť dvoch obrazových bodov. Obecne platí Euklidovská vzdialenosť medzi bodmi (i, j) a (h, k), ktorú definuje vzťah (19):

$$D_E = \sqrt{(i-h)^2 + (j-k)^2} \tag{2}$$

Susedstvo definuje, ktoré body je možné považovať za susedné. Pri využití štvorcovej diskrétnej mriežky môžeme uvažovať 4-susedstvo alebo 8-susedstvo.Ukážka susedstva bodu je na obrázku Obr. 16). Prípad 8-susedstva určitého bodu, predstavuje všetky okolité body, vrátane uhlopriečnych. Naproti tomu 4-susedstvo nepovažuje za susedov body v uhlopriečnej vzdialenosti, čiže susedné body [21],[20]. Vzdialenosti medzi bodmi sú potom definované podľa nasledujúcich vzťahov (3) a (19), pre 4-susedstvo (D_4) a pre 8-susedstvo(D_8):

$$D_4 = |i-h| + |j-k| \tag{3}$$

$$D_8 = max\{|i-h|; |j-k|\} \tag{4}$$



Obr. 16) Ukážka susedstva obrazového bodu v štvorcovej mriežke [21]

Oblasť je potom súvislá množina pixelov navzájom viazaných relácií susedstva a dva pixely v obraze, medzi ktorými existuje cesta sa nazývajú súvislé pixely.

4.2.2 Kvantovanie

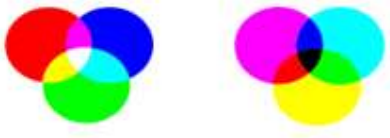
Prevod hodnoty obrazovej funkcie do číslicovej podoby sa nazýva kvantovanie. Počet kvantovacích úrovní musí byť dostatočne vysoký na to, aby nevznikali falošné obrysy. Tento jav vzniká pri kvantizačnej úrovni menšej ako 50. Jav je možné odstrániť nelineárnym kvantovaním kde sa zväčšuje rozsah tých intervalov jasu, ktoré sú v obraze málo zastúpené. Najčastejšie sa kvantuje na 256 úrovní jasu, čo zodpovedá kódovaniu 8 bitmi. Binárne obrazy vystačia s 1 bitom, ktorý rozlišuje pixely objektu(1) a pixely pozadia(0) [21],[20].

4.2.3 Farebné modely

Farebný model je model, ktorý popisuje základné farby a určuje, akým zmiešaním týchto základných farieb je možné dosiahnuť požadované farby. Ukážka miešania farieb je na obrázku Obr. 17). V prírode je farba daná zmesou svetla v rôznych vlnových dĺžkach. Farebné modely sa snažia tieto farby napodobniť čo najvernejšie. Pixely obsahujú dôležitú informáciu o farebných zložkách o danej intenzite [21],[20].

Miešanie farieb sa delí do kategórií:

- **Adatívne miešanie farieb** - zmiešaním všetkých farieb vznikne farba biela (pracuje so svetlom) využívajú ho napríklad monitory alebo projektory.
- **Subtraktívne miešanie farieb** - zmiešaním všetkých farieb vznikne farba čierna (pracuje s odrazom svetla) využíva sa napríklad v tlačiareni.



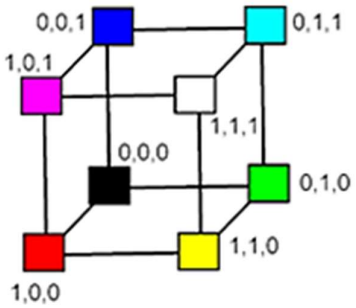
Obr. 17) Ukážka adatívneho miešanie farieb (vľavo) a subtraktívneho miešanie farieb (vpravo) [27]

Medzi najbežnejšie používané farebné modely patrí:

- **RGB**
- **CMYK**
- **HSV**

RGB model

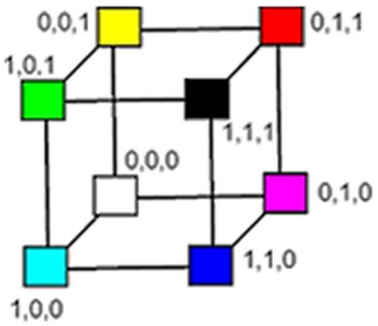
Skratka RGB označuje farebný model troch základných farieb, červená (red), zelená (green) a modrá (blue). Model je možné predstaviť si ako trojrozmerný priestor kde osi reprezentuje farebný odtieň konkrétnej farby. Ukážka modelu je na obrázku Obr. 18) Z toho vychádza aj tak zvané adatívne miešanie farieb. Ak intenzita farebnej zložky je kódovaná 8-bitmi (256 úrovní) tak je možné namiešať ľubovoľnú farbu. Počet kombinácií farieb sa rovná 256^3 čo je 16 777 216 kombinácií. Bielu farbu reprezentuje kombinácia intenzity farebných zložiek (od 0 do 255) kombinácia (255,255,255). Naopak kombinácia čiernej farby je (0,0,0). Tento model sa využíva hlavne v zobrazovacích zariadeniach [21],[20],[19].



Obr. 18) Farebný model RGB

CMYK model

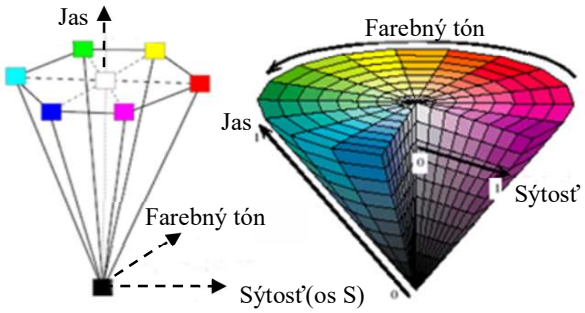
Farebný model CMYK rovnako ako RGB vyjadruje názvy základných farieb. Cyan (Azúrová), Magenta (purpurová), Yellow (žltá) a blacK (čierna). Model je znázornený na obrázku Obr. 19) Na rozdiel od RGB modelu použitie spočíva v miešaní skutočných farieb. Bielu farbu teda získame v absencii základných farieb. Čiernu farbu získame teoreticky zmiešaním všetkých troch základných farieb CMY v maximálnej intenzite. Takto nevznikne čisto čierna farba, ale v skutočnosti zmes hnedej a čiernej. Preto sa tento model dopĺňa o ďalšiu farbu a to čiernu, ktorá zároveň slúži pre stmavenie odtieňov. Takto doplnený model sa nazýva CMYK. Tento model je subtraktivný, preto sa využíva predovšetkým v tlačiarenskej technike [19],[20],[21].



Obr. 19) Farebný model CMYK

HSV model

Farebný model nazývaný HSV lepšie zodpovedá popisu farieb, na ktorý je človek zvyknutý vďaka intuitívnemu popisu farieb. Model HSV má tri základné parametre, farebný tón (Hue), sýtosť (Saturation) a jas (Value). Farebný tón určuje prevládajúcu spektrálnu farbu. Sýtosť prímes iných spektrálnych farieb. Jas prímes bielej farby (bieleho svetla). V niektorých prípadoch sa model HSV označuje ako HSB [19],[21][21]. Pre popis zobrazenie farieb modeli HSV sa používa šesťboký ihlan, ktorý vidíme na obrázku Obr. 20).



Obr. 20) Farebný model HSV [25]

Tento ihlan je umiestnený do súradnicového systému tak, že vrchol ihlanu je v počiatku súradnicového systému a os ihlanu je zhodná so zvislou osou, ktorá zároveň znázorňuje zmeny úrovne jasu. Jas aj sýtosť, ktoré sú umiestnené na vodorovnej osi, sa mení v intervale <0, 1>. Na obvode podstavy ihlanu sa teda nachádza čisté farby. Farebný tón je definovaný ako veľkosť uhla, ktorá sa meria od osi S proti smeru hodinových ručičiek. Môže nadobúdať hodnoty 0 - 360◦. Problémom modelu HSV je prechod medzi čiernou a bielou farbou, ktorý nie je plynulý a pohyb farebného tónu sa neodohráva po kružnici, ale po šesťuholníku [21].

4.3 Predspracovanie obrazu

Po úspešnom získaní obrazu a jeho digitalizáciu máme k dispozícii digitálny obraz pozorovanej scény. Tento obraz však môže byť skreslený vďaka spôsobu snímania alebo nevhodných

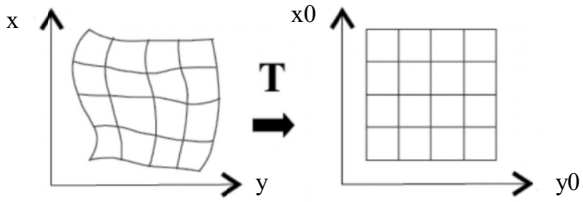
podmienkach v priebehu snímania. Túto chybu spôsobenú skreslením je možné opraviť, ak je známy charakter skreslenia. Napríklad môžeme opraviť pixel skreslený šumom na základe hodnôt jeho susedov. Existuje množstvo metód, ktoré uľahčujú ďalšiu analýzu obsahu obrazu, objektov alebo len zvýrazňujú dôležité rysy obrazu pre uľahčenie pozorovania človekom [19],[21].

Medzi základné rozdelenie metód predspracovanie obrazu patrí:

- Geometrická transformácia
- Jasová transformácia
- Vyhladzovanie a ostrenie obrazu

4.3.1 Geometrická transformácia

Geometrická transformácia 2D obrazu je vektorová funkcia T, ktorá zobrazí bod (x,y) do bodu (x0 , y0). Situáciu ilustruje Obr. 21), kde je časť roviny transformovaná bod po bode. Môžeme si to predstaviť ako vyrovnávanie deformovanej mriežky nakreslenej na elastickej podložke. Hľadáme teda takú kombináciu pôsobiacich síl (t.j. parametrov transformačnej funkcie), aby sa kresba na elastickej podložke čo najviac podobala pravidelnej mriežke [20].



Obr. 21) Geometrická transformácia súradníc v rovine [20]

Geometrické transformácie vypočítajú na základe súradníc bodov vo vstupnom obraze súradnice bodov vo výstupnom obraze. Nejedná sa teda o transformáciu hodnoty funkcie, ale jej súradníc. Vďaka tomu môžeme odstrániť geometrické skreslenie vzniknuté pri snímaní obrazu (napr. korekcie geometrických porúch objektívu kamery, oprava skreslenia družicového snímku spôsobená zakrivením zemegule).Transformácia T je definovaná vzťahmi (19):

$$x_0 = T_x(x, y) \qquad y_0 = T_y(x, y) \tag{5}$$

Transformačné rovnice T_x a $T_y$môžu byť vopred známe (rotácia, posunutie) alebo môžu byť odvodené na základe znalostí vstupného a transformovaného obrazu. Za týmto účelom sa na oboch obrazoch nájdu dvojice zodpovedajúcich si bodov (lícovacie body), ktoré sa na obrázkoch ľahko hľadajú napr. priesečníky vláknitých štruktúr, rohy objektov a pod.

Samotná geometrická transformácia pozostáva z transformácie súradníc a aproximácie jasovej funkcie [19],[20].

Transformácia súradníc.

Body vstupného obrazu, ktoré majú celočíselné súradnice sa transformujú do nových pozícií, ktoré nemusia zodpovedať celočíselnej mriežke, ale majú podobu reálnych čísel. Plošná transformácia má teda bodový charakter. Pre prax sa využíva bilineárna transformácia, na ktorú

postačujú 4 dvojice vzájomne si zodpovedajúcich lícovacích bodov na vstupnom resp. na transformovanom obraze [19]. Pre bilineárnu transformáciu platia vzťahy (6) a (19).

$$x_0 = a_0 + a_1 x + a_2 y + a_3 xy \tag{6}$$

$$y_0 = b_0 + b_1 x + b_2 y + b_3 xy \tag{7}$$

Zvláštnym prípadom biliárnej transformácie je afinná transformácia, ktorá v sebe zahrňuje najčastejšie používané formy transformácie, ako je translácia, rotácia, a škálovanie. Pri tejto transformácii stačia 3 páry lícovacích bodov. Pre affinú transformáciu platia vzťahy (8),(19).

$$x_0 = a_0 + a_1 x + a_2 y \tag{8}$$

$$y_0 = b_0 + b_1 x + b_2 y \tag{9}$$

Použitím iba niektorých koeficientov dostaneme základné, najpoužívanejšie typy transformácií [19],[21]:

- **Translácia** (10) (koeficienty a_0 a $b_0$určujú posun)

$$x_0 = x + a_0 \quad y_0 = x + b_0 \tag{10}$$

- **Zmena škály** (11) (koeficienty a_1a $b_2$určujú koeficienty stlačenia)

$$x_0 = a_1 \qquad y_0 = b_2 y \tag{11}$$

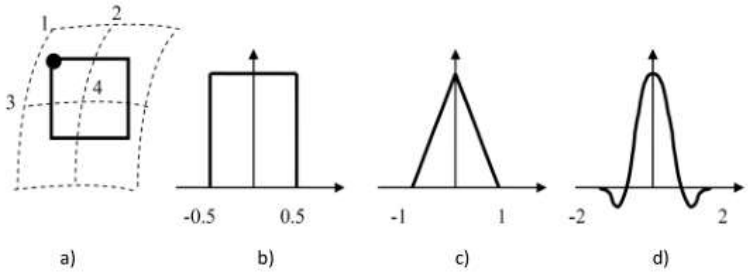
- **Rotácia** (19) (koeficienty $a_2 =$ sin φ a $b_1 =$ −cos φ, kde φ je uhol rotácie[20].

$$x_0 = x + a_2 y \qquad y_0 = y + b_1 x \tag{12}$$

Aproximácia jasovej funkcie.

Hľadáme hodnotu jasu v celočíselných pozíciách a to interpoláciou na základe hodnôt transformovaných súradníc (reálnych čísel), ktoré sú najbližšie.

Geometrickou transformáciou sme vypočítali hodnoty (x_0, y_0), ktoré zodpovedajú bodom (x, y) originálneho obrazu. Celočíselné koordináty vstupného obrazu však transformáciou nadobudnú hodnoty z oboru reálnych čísel. To znamená, že poznáme úroveň jasu v bodoch s reálnymi súradnicami (priesečníky prerušovaných čiar, obrázok Obr. 22)a). My však potrebujeme poznať jas v bodoch predstavujúcich celočíselný raster. (štvorec nakreslený plnou čiarou na obrázku Obr. 22)b).



Obr. 22) Ukážka aproximácie jasovej funkcie [19].

Je zrejmé, že hodnoty jasu v bodoch predstavujúcich celočíselný raster získame interpoláciou susedných bodov, ktorých hodnoty poznáme. Je viacero spôsobov, ako túto hodnotu vypočítať.

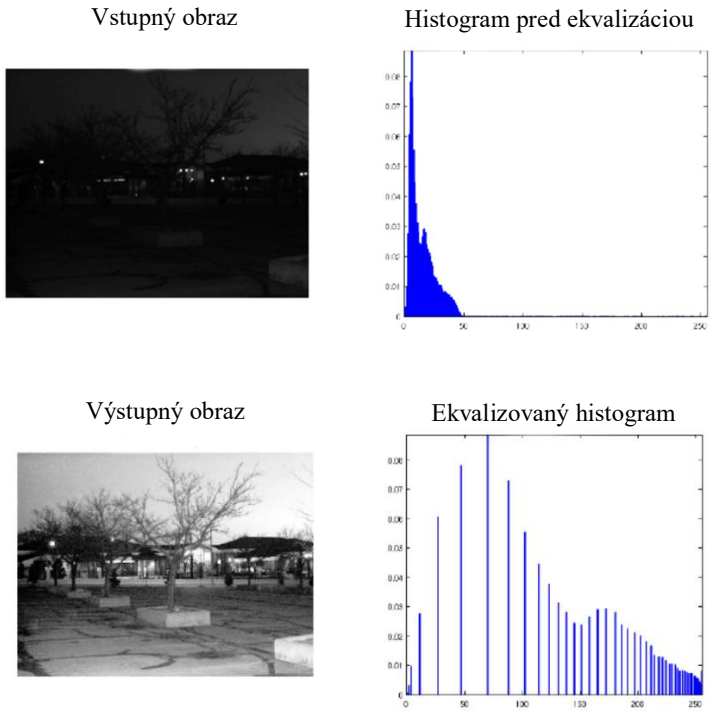
Obrázok Obr. 22)b) zobrazuje interpoláciu metódou najbližšieho suseda, keď jas v hľadanom bode sa určí ako jas najbližšieho transformovaného suseda, ktorého jas je známy. Pri lineárnej interpolácii sa berie do úvahy lineárna kombinácia jasov štyroch najbližších transformovaných susedných bodov (1,2,3,4 pre zvýraznený bod na obrázku Obr. 22)a) Bližší susedné body majú väčší vplyv. Pri bikubickej interpolácii je táto hodnota získaná aproximáciou 16 najbližších susedov, ich vplyv určuje interpolačné jadro v tvare „Mexického klobúka“ (obrázok Obr. 22)d) [20],[21].

4.3.2 Transformácia jasu

Transformácia hodnôt jasu možno rozdeliť na jasové korekcie a transformácia jasovej stupnice.

Pri jasovej korekcii závisí jas v bode výstupného obrazu na jasu bodu zodpovedajúceho vo vstupnom obraze a na jeho blízkom okolí. Používa sa v prípade, keď snímacie zariadenie má odlišnú citlivosť na rôznych miestach obrazu. K tomuto skreslenie môže dochádzať napríklad vplyvom prachových častíc na objektíve, ktoré zabraňujú priechodu svetla alebo slabnutie svetla v optickej sústave vplyvom vinetácie. Podobný efekt má i nerovnomerné osvetlenie snímaného obrazu. Pokiaľ je táto porucha systematická a poznáme odchýlku citlivosti každého bodu od ideálnej charakteristiky, môžeme poruchu korigovať jasovou korekciou. [20].

Pri transformácii jasovej stupnice je len transformovaná určitá hodnota jasu vo vstupnom obraze na inú výstupnú hodnotu bez ohľadu na polohu v obraze. Transformácia jasovej stupnice sa oproti jasovej korekciu netýka jednotlivých pixelov, ale upravuje obraz ako celok a umožňuje upraviť jas obrazu napríklad pre lepšiu zreteľnosť detailov pre človeka. Vtedy pixely s veľmi podobným jasom sú ťažko rozlíšiteľné pre ľudské oko. Ak je však obraz určený k automatickej analýze iba počítačom, neprináša jasová korekcia žiadnu novú informáciu. Pre zvýšenie kontrastu monochromatického obrazu sa veľmi často používa metóda ekvalizácia (vyrovnanie) histogramu. Ekvalizácia zvýši kontrast pre úrovne jasu blízko maxím histogramu a zníži kontrast blízko miním histogramu. Vďaka tomu bude histogram vyrovnaný, čo znamená, že sú jednotlivé jasové úrovne histogramu zastúpené zhruba rovnako početné. Príklad využitia metódy ekvalizácie histogramu je vidieť na obrázku Obr. 23) [21].



Obr. 23) Ukážka metódy ekvalizácie histogramu [23]

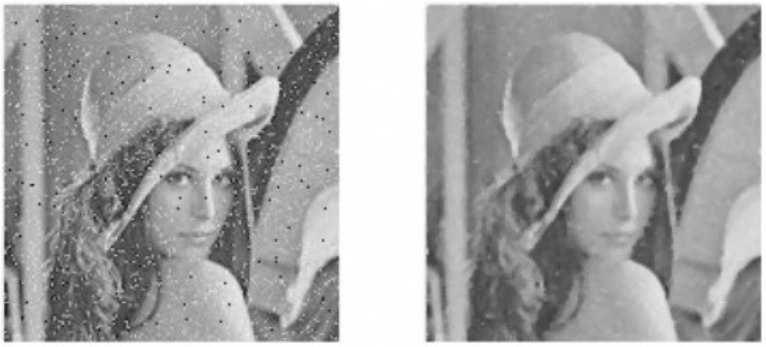
4.3.3 Vyhladzovanie a ostrenie obrazu

Lokálne operátory pracujú tak, že algoritmus systematicky (zvyčajne po riadkoch) prechádza celý obraz a upraví hodnotu každého pixelu na základe hodnoty pixelov v jeho bezprostrednom okolí.

Vyhladzovanie (filtrácia).

Účelom vyhladzovania je odstrániť z obrazu šum a ostré hrany, čiže vysokofrekvenčné zložky v zmysle Fourierovského spektra. Neželaným sprievodným javom tohto procesu je aj rozmazanie hrán, ktoré boli na pôvodnom obraze ostré. Medzi používané filtre patrí napríklad spriemerovanie, filtrovanie pomocou Gaussovej masky, mediánový filter, filtrácia metódou rotujúcej masky. Pri realizácii úloh diplomovej práce je v využitý mediánový filter preto sa bližšie vysvetlí [19],[21].

Mediánový filter je výhodné použiť vtedy ak je potrebné eliminovať vplyv extrémov v obraze. Výpočet spočíva v usporiadaní jasovej úrovne z lokálneho okolia vzostupne a ako mediánovú hodnotu vyberieme jas v strede tejto postupnosti. To zabráni, aby extrémy na krajoch postupnosti ovplyvnili vybranú hodnotu, tak ako je tomu u spriemerovaní. Touto hodnotou nahradíme jas filtrovaného pixelu. Metóda filtrácie veľmi dobre redukuje impulzný šum a je pomerne šetrná k hranám. Naopak filtrácia sa nie celkom dobre hodí v prípade tenkých čiar a ostrých rohov v obraze, ktoré poruší [20].



Obr. 24) Ukážka použitia mediánového filtru [24].

Ostrenie (detekcia hrán).

Jeho účelom je, naopak, zvýrazniť vysokofrekvenčné časti spektra (hrany) a odfiltrovať tie časti obrazu, kde dochádza iba k pomalým zmenám. Neželaným sprievodným javom detekcie hrán je aj zvýraznenie šumu, ktoré má tiež vysokofrekvenčný charakter. Hrana v obraze je vlastnosť obrazového elementu a jeho okolia. Hrana je vektorová veličina, ktorá je určená veľkosťou a smerom. Je treba odlíšiť detekciu hrán od hľadania hraníc. Detekcia hrán je jednoduchou operáciou založenou zvyčajne na lokálnych operátoroch. Naproti tomu hľadanie hraníc je sofistikovanejšou činnosťou (patriacou do oblasti segmentácie obrazov), ktorá môže byť založená na detekcii hrán, avšak využíva znalosti objektov na to, aby jednotlivé úseky hrán správne spojila do hranice [20],[21].

4.4 Segmentácia obrazu

Segmentáciu môžeme chápať ako rozdelenie obrazu na časti, ktoré korelujú s objektmi reálneho sveta. Existujú dva základné spôsoby popisu súborov objektov získaných segmentáciou. Jeden je založený na kvantitatívnom prístupe, čo znamená opis objektov pomocou súboru číselných charakteristík. Týmito charakteristikami môže byť napríklad veľkosť objektu, kompaktnosť, farebný rozptyl [20].Druhou možnosťou ako opísať dané objekty je kvalitatívny prístup. V tomto prístupe sú popisované relácie medzi objektmi a ich tvarové vlastnosti. Spôsob popisu je zvolený vždy podľa toho, na čo bude ďalej využitý. Vo väčšine prípadov je tento opis vstupné informácií pre klasifikáciu (rozpoznávanie) objektov. Výstup je závislý na postupu pri klasifikácie [21].

4.5 Klasifikácia

Bezprostredne po segmentácii nasleduje klasifikácia (rozpoznávanie obrazu). Úlohou klasifikácie je zaradenie objektov nájdených v obraze do skupiny vopred známych tried. Podstatným krokom je porozumenie, do ktorej triedy zaradiť určitý región tak, aby koreloval s objektami vonkajšieho sveta. Rozpoznanie vyžaduje exaktný popis oblastí tak, aby popis mohol byť predložený klasifikátoru, ktorý ich zatriedi do určitých vzájomne disjunktných podmnožín – tried. Základom úspešného rozpoznávania je dobrá reprezentácia znalostí v celej svojej podstate, tak ako je základom umelej inteligencie. Existujú dve základné formy opisu objektu, podľa nich delíme aj metódy rozpoznávania na dve základné skupiny [20],[21]:

Príznaková klasifikácia je spojená s kvantitatívnym popisom objektov. Príznaky popisujúce objekt majú zvyčajne charakter vektora, ktoré sú vstupnými dátami príznakových (štatistických) algoritmov. Klasifikácia prebieha na základe učenia klasifikátora a to s trénovaciou množinou ale aj bez nej. Príkladom často používanej metódy príznakovej klasifikácie je zhluková analýza, ktorá slúži na triedenie jednotiek do skupín (zhlukov) tak, aby jednotky patriace do rovnakej skupiny mali určité podobné vlastnosti oproti objektom z iných skupín [21].

Štrukturálna klasifikácia je spojená s kvalitatívnym popisom objektov. Využíva primitív objektu, tj. objektu sú priradené základné vlastnosti, ktoré objekt charakterizujú. Na tieto vlastnosti sú potom aplikované algoritmy rozboru slová popisujúceho objekt a kontroly syntaxe pre definovanú gramatiku, jazyk a abecedu [21].

5 ÚLOHA DETEKCIA A TRACK-OVANIE FAREBNÉHO OBJEKTU

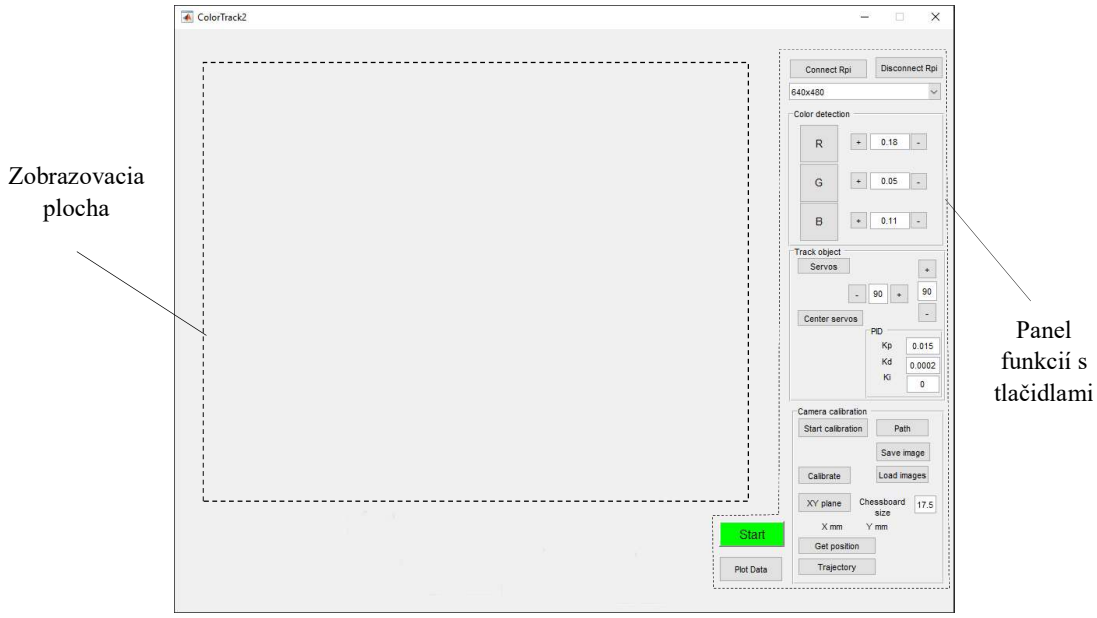
Detekciou farby sa rozumie proces hľadania farby v obraze z kamery. Farba je viditeľná oblasť elektromagnetického žiarenia s vlnovými dĺžkami od 380 do 780 nm. Obrazový snímač kamery je schopný tieto jednotlivé farebné zložky zachytiť, a vytvoriť dáta obrazu, ktorého každý pixel je nositeľ informácie o intenzite farby. Použitím správnych postupov spracovania obrazu, sme schopný v obraze tieto farby detegovať a lokalizovať tak farebný objekt.

Track-ovanie farebného objektu sa realizuje ako zaznamenávanie jeho pohybu počítačom pomocou kamery. Na základe obrazu z kamery vyhodnotíme presnosť určovania polohy objektu. Aby sme mohli kameru použiť na presne určovanie polohy, je potrebné zistiť, jej vnútorné a vonkajšie parametre, k čomu poslúži kalibrácia kamery. Kalibrácia kamery sa vyžaduje najmä pri metódach triangulácie. Triangulácia je metóda určovania relatívnej polohy objektov pomocou známej polohy iného objektu v zosnímanom obraze využívajúc zákonitosti geometrie trojuholníkov. Vstupom je obraz scény s kalibračným objektom a cieľom kalibrácie kamery je získať vzťah medzi súradnicami bodu scény a priestorovými súradnicami bodu scény, k čomu slúži spomínaná triangulácia. Keďže šošovka kamery nie je dokonalá, potrebujeme vypočítať aj koeficienty skreslenia kamery a na základe týchto koeficientov upraviť obraz tak, akoby vyzeral, keby šošovka bola dokonalá. Metódy track-ovania sa využívajú pri navigácii robotov v priestore [28].

Nasledujúce kapitoly obsahujú popis používania grafického užívateľského prostredia, v ktorom sú implementované spomínané funkcie. Použité algoritmy budú popísane spolu s kódom Matlab. Nakoniec experimentom overíme presnosť pri určovaní polohy.

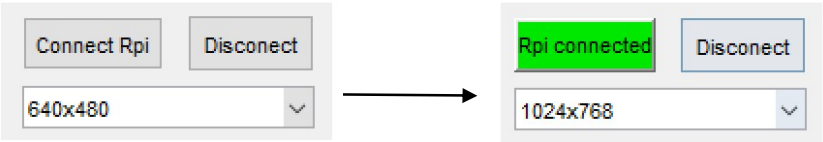
5.1 Popis grafického užívateľského prostredia

Spustením aplikácie sa zobrazí užívateľské okno s názvom *Color track*, ktoré obsahuje panely funkcií s tlačidlami a zobrazovaciu plochu. Panely sú pomenované podľa príslušných funkcií ako *Color detection,* detekcia farieb, *Camera rotation*-natáčanie kamery a *Camera calibration* -kalibrácia kamery pre presné meranie pozície farebného objektu. Ukážka grafického rozhrania aplikácie je na obrázku Obr. 25).



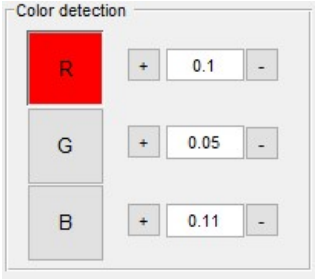
Obr. 25) Ukážka grafického užívateľského prostredia úlohy detekcia a track-ovanie farebného objektu

Ako prvé je nutné zvoliť požadované rozlíšenie snímky a vytvoriť spojenie s Raspberry Pi. To sa realizuje spoločne stlačením tlačidla *Connect Rpi.* Úspešnú komunikáciu a spustenie kamery, signalizuje červená kontrolná dióda na module kamery. Taktiež aplikácia skontroluje pripojenie a tlačidlo *Connect Rpi* zmení vzhľad, na *Rpi Connected* zafarbené na zeleno. Tlačidlom *Disconnect* sa ukončí pripojenie Rpi a vypne kamera. Pri zmene rozlíšenia je potrebné opakovať kroky odpojenia a pripojenia Rpi. Tlačidlo *Start,* spustí prenos snímaného obrazu kamerou do zobrazovacej plochy aplikácie. Ukážka interakcie na obrázku Obr. 26).



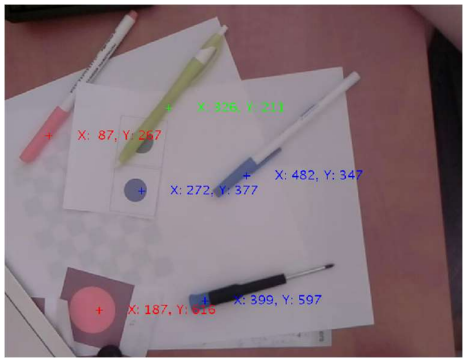
Obr. 26) Interakcia pripájania Raspberry Pi

Panel *Color detection* obsahuje tlačidlá *R, G, B* spolu s nastavením prahovania *(treshhold)*.Ukážka panela je na obrázku Obr. 27) . Tlačidlami R (červená), G (zelená), B (modrá) nastavujeme akú farbu chceme detegovať.



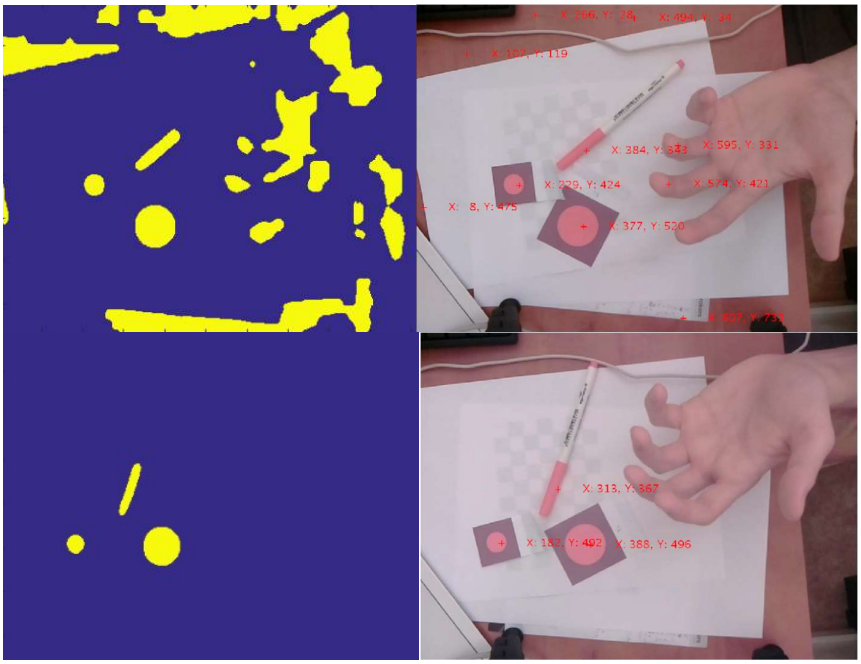
Obr. 27) Ukážka panela Color detection

Na základe hodnoty prahovania a voľby farby sa deteguje požadovaná farba a označí v obraze súradnicami pixelov. Tieto súradnice sú určené ako centroid zhluku rovnakej farby. Preto určovanie centroidu objektu najlepšie zodpovedá kruhu. Ukážka detekcie farieb je na obrázku Obr. 28)



Obr. 28) Ukážka detekcie farieb zelenej, modrej a červenej

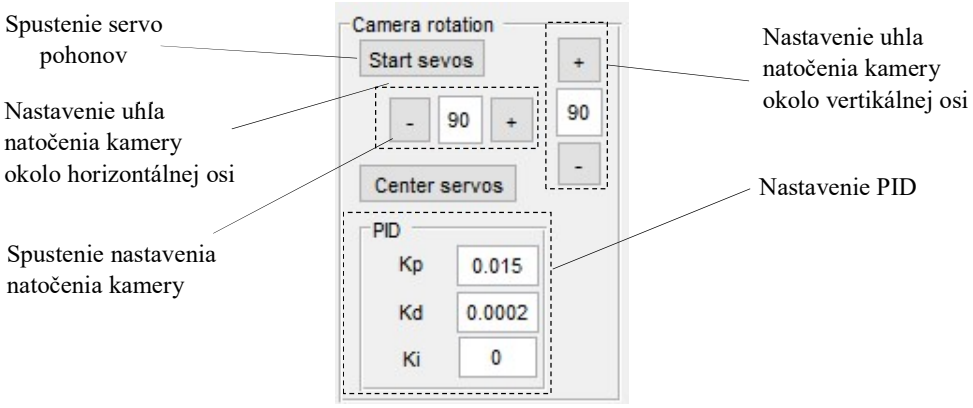
Hodnoty prahovania je možné zadávať priamo do kolónky alebo zvyšovať/znižovať o stotinu bodu tlačidlami „+“ alebo „-“. Je to praktické pri prahovaní farieb, pri rôznych svetelných podmienkach, kedy je treba eliminovať šum. Pri zmene hodnoty prahovania sa eliminujú úrovne jasu červenej farby. Efekt je možné pozorovať v externom okne kde je zobrazený binárny obraz farby, pre ktorú nastavujeme prahovanie. Ukážka je na obrázku Obr. 29). Cenroidy objektov sú označené symbolom„+“ a výpisom súradníc v pixeloch.



Obr. 29) Ukážka detekcie červenej farby pri hodnote prahovania 0,1(hore) a 0,16(dole)

V paneli Panel *Camera rotation* obsahuje funkcie spustenia servo pohonov, natáčanie kamery a nastavenia parametrov pre PID reguláciu servopohonov. Signálové vodiče serva sú pripojené na GPIO piny platformy číslo 12 a 11. Panel je popísaný na obrázku Obr. 30).

Tlačidlom *Start servos* sa aktivuje a opakovaným stlačením deaktivuje natáčanie kamery. Aktivované natáčanie signalizuje šedá farba tlačidla. Tlačidlo *Center servos* umožňuje ľubovoľné natočenie kamery alebo vrátenie kamery do nastavenej polohy. Umožňuje tak nastaviť požadovaný pozorovaný priestor kamery. Ten je možné sledovať na zobrazovacej ploche kde stred obrazu je označený bielym symbolom (+). Natočenie kamery okolo vertikálnej a horizontálnej osi sa nastavuje priamo napísaním požadovaného uhla do príslušnej kolónky alebo klikaním na tlačidla *mínus* (-) alebo *plus* (+). Ovládanie natočenia kamery je možné len v kombinácii súčasne stlačených tlačidiel *Start servos* a *Center servos*.



Obr. 30) Ukážka panela Camera Rotation

Pri aktivovaných tlačidlách *Start, Camera rotation* a tlačidla farby, ktorú chceme trackovať, sa spustí sledovanie polohy objektu. Servo pohony otáčajú kamerou okolo vertikálnej a horizontálnej osi tak, aby optická os kamery prechádzala centroidom objektu. Optickú os kamery reprezentuje symbol *„X“* označený bielou farbou a centroid objektu symbol *„O“,* ktorý je označený fialovou farbou. Regulácia otáčania seriev je realizovaná pomocou PID algoritmu. Hodnoty proporcionálnej zložky Kp, derivačnej Kd a integrálnej Ki, je možné nastavovať v príslušných kolónkach panelu. Nastavenie regulácie polohy, ktoré fungovalo dostatočne dobre, je v tabuľke Tab 1) **.** Hodnoty sa nastavovali ručne tak aby rotovanie kamery príliš nekmitalo pri dosiahnutí polohy objektu. Diskrétny algoritmus PID je popísaný v Príloha E - PID regulácia servopohonu.

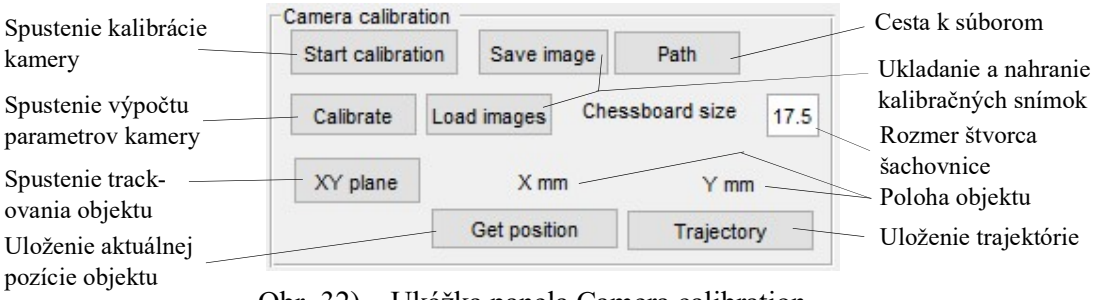
Ukážka sledovania polohy objektu je na obrázku Obr. 31)

Obr. 31) Ukážka serie snímok pri trackovani objektu

PID parametre	
Kp	0.015
Kd	0.0002
Ki	0

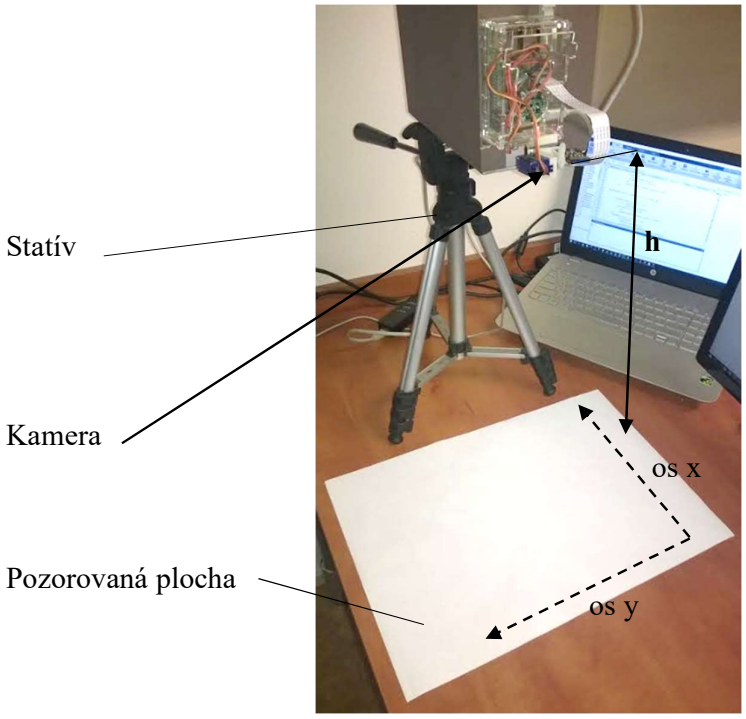
Tab 1) Parametre PID

Panel *Camera calibration* združuje funkcie kalibrácie kamery a následného track-ovania objektu v priestore pozorovanom kamerou. Ukážka panelu je na obrázku Obr. 32). Funkcie v panely sú aktívne pokiaľ je zapnuté tlačidlo *Start calibration*. Týmto sa spustí obrazový prenos a môžeme začať kalibrovať kameru.



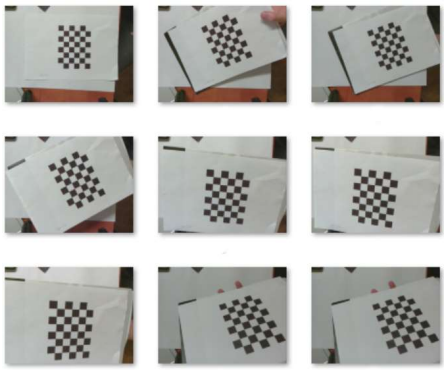
Obr. 32) Ukážka panela Camera calibration

Pre presné track-ovanie je dôležité kameru kalibrovať. Kamera sa umiestni do statívu v určitej vzdialenosti *h* od vodorovnej plochy. Ukážka zostavy je na obrázku Obr. 33).



Obr. 33) Zostava pri kalibrácii kamery

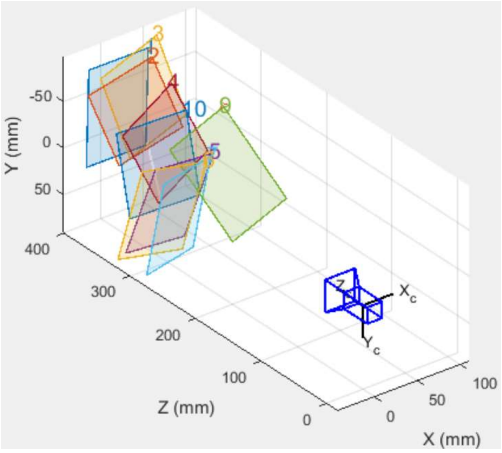
Kamera sa kalibruje sériou snímok šachovnice, ktorá sa umiestňuje v rôznych uhloch a vzdialenostiach od objektívu kamery. Na základne tejto sérií snímok, sa vypočíta poloha kamery v priestore a jej skreslenie. Ukážka série snímok je na obrázku Obr. 34). Platí, že čím viac obrázkov tým presnejšia je kalibrácia.



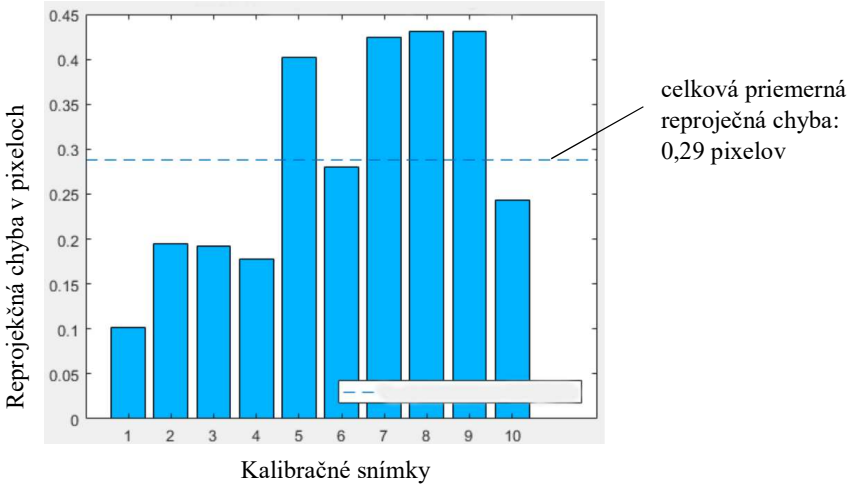
Obr. 34) Ukážka série kalibračných snímkov

Tieto snímky sa ukladajú na disk tlačidlom *Save image.* Miesto a názov priečinku, kam sa majú snímky uložiť sa definuje tlačidlom *Path.* Kalibračné snímky nie je potrebné zhotovovať zakaždým, keď spustíme aplikáciu za predpokladu, že sa s kamerou nemanipulovalo. Tlačidlom *Load images* nahráme uložené kalibračne snímky na základe, ktorých znova môžeme nakalibrovať kameru. Tlačidlom *Calibrate* sa realizuje samotná kalibrácia parametrov. Pretým však je dôležité v kolónke *Chessboard size* zadefinovať stranu štvorca šachovnice v milimetroch. Po úspešnej kalibrácii kamery sa pre kontrolu zobrazí rekonštrukcia scény a výpočet reprojekčnej chyby. Ukážky sú na obrázkoch Obr. 35) a Obr. 36).Reprojekčná chyba

označuje vzdialenosť medzi bodmi detegovanými v kalibračných snímkach a bodmi reprojektovanými naspäť do snímky, na základe vypočítaných parametrov kamery. Výrazne rozdiely chýb medzi kalibračnými snímkami nás upozornia nato, ktoré snímky vylúčiť pretože negatívne ovplyvňujú presnosť kalibrácie kamery. Po selekcii snímkou je treba kalibráciu opakovať. Problematika kalibrácie kamery je detailne rozobraná v Príloha D - Kalibrácia kamery.



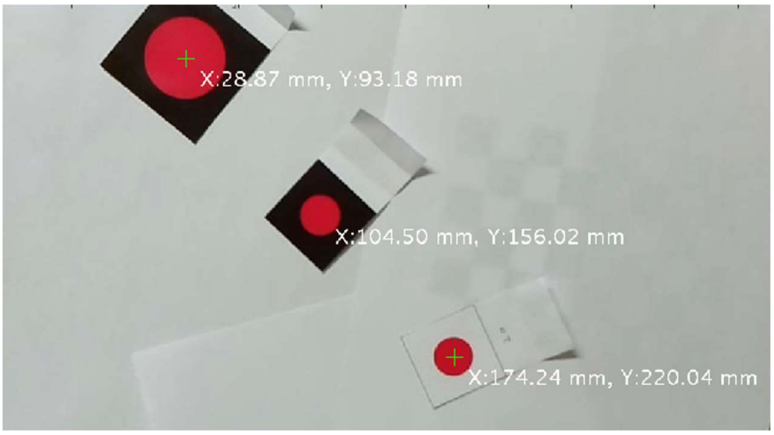
Obr. 35) Ukážka rekonštrukcia scény



Obr. 36) Graf priemerných reprojekčných chýb kalibračných snímkov

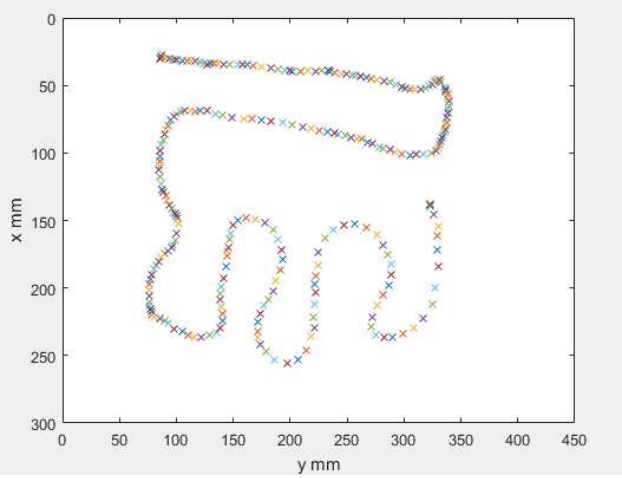
Predtým ako začneme track-ovať objekt pred kamerou je potrebné zorientovať osy *x* a *y*, znova pomocou šachovnice. Týmto docielime, že sa vypočíta rotácia a translácia kamery a budeme schopný pre každý pixel v obraze, čo najpresnejšie odhadnúť polohu v reálnom svete. Ak máme šachovnicu pred objektívom kamery stlačíme tlačidlo *XY plane.* Na obrázku Obr. 37) môžeme vidieť červené objekty, ktorých súradnice centroidu*(x,y)* vyjadrujú vzdialenosť v milimetroch od počiatku súradnicového systému. Aktuálne súradnice body sa zakresľujú do obrazu a do koloniek *Xmm* a Ymm v paneli. Počiatok súradnicového systému je ľaví horný roh

snímky. Takto máme nakalibrovanú kameru a môžeme presne určovať polohu objektu, vzdialenosť od objektívu alebo merať šírku objektov.



Obr. 37) Ukážka určenia súradníc objektov

Tlačidla *Get Position* a *Trajectory* majú za úlohu uložiť súradnice track-ovaného bodu. Tlačidlom *Trajectory* sa spustí funkcia, ktorá ukladá každú zmenu súradníc objektu v obraze pokiaľ dôjde k pohybu. Tlačidlom *Get Position* uložíme pozíciu objektu len v okamihu stlačenia tlačidla. Dáta sa ukladajú do textových súborov na disk. Tlačidlom *Plot Data* je možné vykresliť údaje súradníc polohy objektu do grafu. Ukážka vykreslenia sledovanej trajektórie objektu je na obrázku Obr. 38).



Obr. 38) Ukážka vykreslenia trajektórie pohybu objektu

5.2 Popis použitých algoritmov

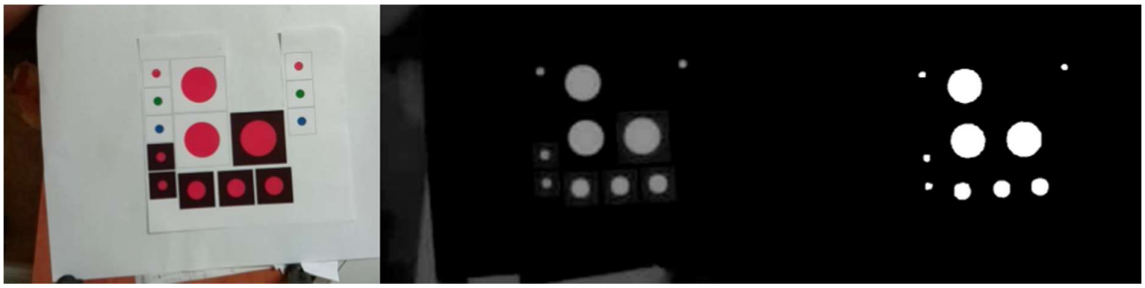
V nasledujúcich podkapitolách budú popísane dôležité časti algoritmov zaujímavých z pohľadu spracovania obrazu a počítačového videnia.

5.2.1 Detekcia farebného objektu

Ako je z teórie známe tak obraz RGB obsahuje maticu [*m n z*], kde *m* a *n* sú jeho rozmery (rozlíšenie) a *z* je vrstva intenzity farebnej zložky zelenej, červenej a modrej.

Na detekciu farby uvedieme príklad kódu detekcie červenej farby v obraze. Pre detekciu červenej farby musíme získať len jej najintenzívnejšie zložky teda pixely kde jednoznačne dominuje červená zložka. To zistíme tak že RGB obraz *(frame)*,najprv prevedieme a na odtieň šedej *(rgb2gray)* a potom odčítame od červenej zložky obrazu *frame(:,:,1)*. Prevod RGB na prevod šedej*(rgb2gray)*, matlab uskutočňuje podľa algoritmu Y=(0.2989)R+(0.5870)G + (0.1140)B, kde Y je výsledná hodnota. Transformačné pomery používa podľa normy *Nonlinear luma*. Ďalším krokom je odstránenie šumu pomocou mediánového filtra*(medfilt2)* a nakoniec prevod na binárny obraz pomocou *imbinarize*. Funkcia nahradí všetky hodnoty v *dframe* podľa histogramu Otsu's metódy, ktorá hodnoty pixelov prevedie na 1 alebo 0 podľa zadaného prahovania *(trashold)*. Hodnoty 1 reprezentujú červenú farby. Ukážka krokov detekcie červenej farby na obrazku Obr. 39).

```
%Získanie komponenty farby
dframe= imsubtract(frame(:,:,1), rgb2gray(frame));
%Odstránenie šumu
dframe = medfilt2(dframe, [3 3]);
%Prevod na binárny obraz
binframeRed = imbinarize(dframe, treshold);
```



Obr. 39) Ukážka detekcie červenej farby(zľava originál obraz, extrakcia červenej komponenty, binárny obraz)

5.2.2 Track-ovanie farbného objektu

Track-ovaním farebného objektu budeme rozumieť track-ovanie bodu, ktorý reprezentuje jeho ťažisko. Pre zjednodušenie používame kruh.

```
%Nájdenie ťažiska a ohraničenia
[centroidRed, bboxRed] = step(hblob, binFrameRed);
```

Z binárneho obrazu *(binframe)* pomocou operátora blob analýzy*(hblob)* nájdeme ťažisko*(centroidRed)* a ohraničenie (*bboxRed)* kruhu. Ohraničenie použijeme pri určovaní rozmerov objektu. Blob analýza v Príloha C - Detektor Shi Tomasi, KLT tracker, Blob analýza a binarizácia obrazu

Pre pochopenie nasledujúcej časti je potrebne preštudovať kapitolu kalibrácie kamery, ktorá je v Príloha D - Kalibrácia kamery.

Po kalibrácii kamery poznáme vnútorné parametry kamery *(cameraParams).* Použijeme funkciu *unistortImage* aby sme obraz, na ktorej je šachovnica *(frame),* upravili pomocou vnútorných parametrov kamery a tak sa korigovalo skreslenie. To je mimoriadne dôležite pre presnosť. Takto dostaneme pozíciu počiatku súradného systém (*newOrigin)* a neskreslený obraz(*im)*. Následne pokračuje detekcia šachovnice*(detectCheckerboardPoints),* ktorá vyhľadá na neskreslenom obraze*(im),* rohy štvorcov šachovnice*(imagePoints)*.

Nakoniec sa prepočítajú vonkajšie parametre kamery, pomocou funkcie *extrinsics,* kde argumenty sú body šachovnice *imagePoints,* body šachovníc z kalibračných snímkov *worldPoints* a vnútorne parametre kamery *cameraParams*. Výstupom je rotácia *R* a translácia kamery *t.*

```
%Zorientovanie kamery v priestore
[im, newOrigin] = undistortImage(frame, cameraParams, 'OutputView', 'full');
%Extrakcia bodov šachovnice
[imagePoints, boardSize] = detectCheckerboardPoints(im);
%Výpočet vonkajších parametrov
[R, t] = extrinsics(imagePoints, worldPoints, cameraParams);
```

Uskutočnením týchto príkazov sú nám známe všetky parametre, ktoré je nutné poznať pre určenie pozície bodu v reálnom svete, tak aby zodpovedala skutočnosti čo najviac. To dosiahneme pomocou funkcie *pointsToWorld.* Takže *worldPoints* sú súradnice hľadaného bodu*(centroidRed)* v pozorovanej scéne kamery. Súradnice zodpovedajú bodu v obraze s zohľadneným skreslenia.

```
%Prevod pixelu na bod vo svete
worldPoints = pointsToWorld(cameraParams, R, t, centroidRed);
```

Pomocou euklidovskej vzdialenosti bodov sme schopný dopočítať súradnice x a y v milimetroch. Súradnica x bodu, ktorý považujeme za počiatok súradnicového systému v realite je *(origin(1)).*

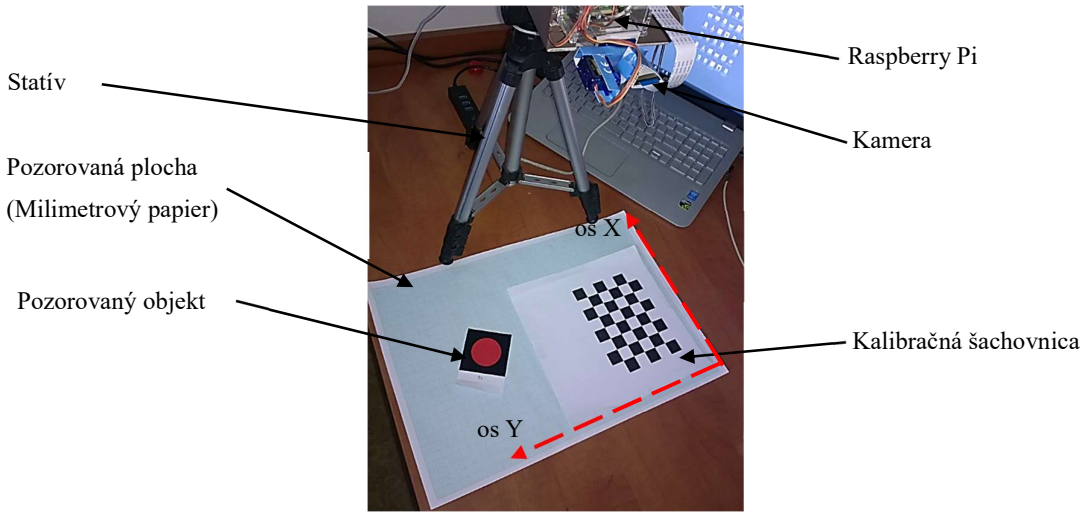
```
%Vypočet x súradnice
x_position=sqrt(power(origin(1)-worldPoints(1),2));
```

5.3 Experiment

V tejto kapitole experimentom overíme presnosť merania polohy objektu. Kameru umiestnime do statívu v určitej vzdialenosti od vodorovnej plochy a skalibrujeme. Na vodorovnú plochu umiestnime milimetrový papier a budeme posúvať farebným objektom, o dĺžku 10 milimetrov po osi X a Y v oblastiach okraju snímky, kde sa predpokladá najväčšie skreslenie obrazu. Súčasne stým, budeme zaznamenávať hodnoty súradníc objektu, pri posunutí pomocou vytvorenej aplikácie.

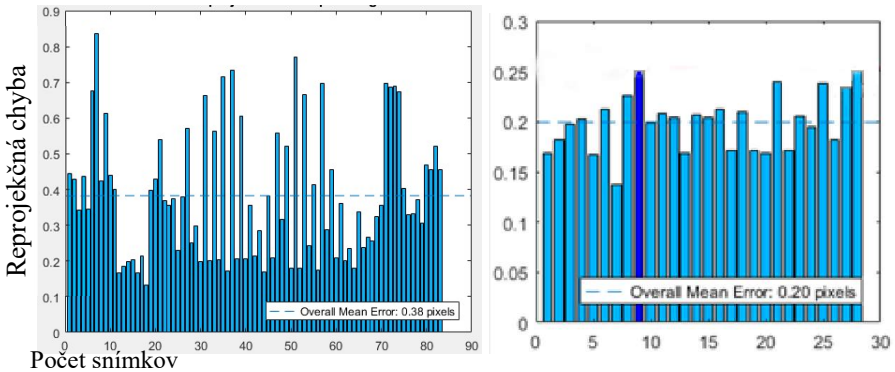
5.3.1 Priebeh experimentu

Do statívu sme umiestnili kameru vo výške 40 centimetrov od vodorovnej plochy stola. Na stôl sa umiestnil milimetrový papier. Ako farebný objekt sa použil červený kruh, ktorého priemer je 40 milimetrov. Obrázok zostavy a orientácia osí je na obrázku Obr. 40).



Obr. 40) Obrázok zostavy a orientácie osy pri track-ovaní polohy objektu

Celý experiment sa uskutočnil pri rozlíšení kamery 1280x720. Na začiatok sa skalibrovala kamera pomocou kalibračných snímok, ktorých sa vyhotovilo 85. Následne sa vyselektovali snímky, ktoré majú príliš veľkú reprojekčnú chybu. Týmto sme sa zbavili snímkou, ktoré negatívne ovplyvňujú presnosť kalibrácie. Ukážka konsolidácie reprojekčnej chyby na obrázku Obr. 41).



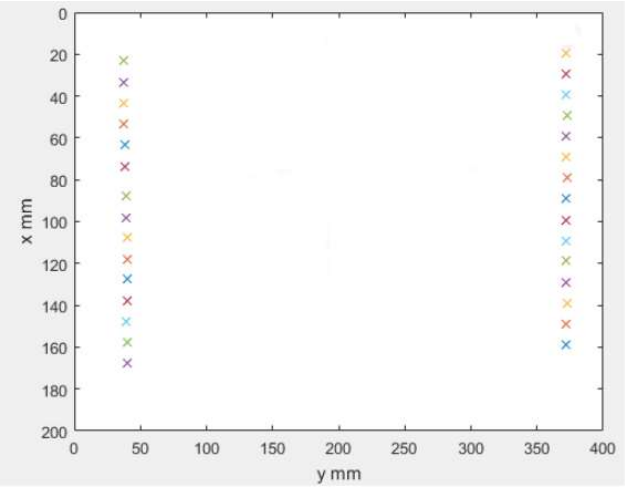
Obr. 41) Konsolidácia reprojekčnej chyby

Po úspešnej kalibrácii sa začalo s meraním polohy, kde sa zaznamenávali súradnice zakaždým ako sa pohlo objektom o 10 milimetrov, najprv ose X a potom Y v oblastiach okolo okrajov snímky. Namerané hodnoty pri meraní X súradníc sú v tabuľke Tab 2) Ukážka vykreslenia x súradníc je na obrázku Obr. 42). Tabuľka hodnôt pri meraní Y súradníc je v tabuľke Tab 6) V tabuľkách je vypočítaný aritmetický priemer vzdialeností medzi krokmi a štandardná odchýlka.

X [mm]	**Y [mm]**	$x = X_n - X_{n-1}$	$x - \bar{x}$
159.52	42.93		
149.36	42.87	10.16	-0.1292
139.55	43.13	9.81	0.2207
129.43	43.06	10.12	-0.0892
119.66	43.00	9.77	0.2607
109.58	42.93	10.08	-0.0492
99.84	42.86	9.74	0.2907
89.48	42.79	10.36	-0.3292
79.46	42.73	10.02	0.0107
69.46	42.98	10.00	0.0307
59.47	42.92	9.99	0.0407
49.51	42.85	9.96	0.0707
39.25	42.46	10.26	-0.2292
29.32	42.72	9.93	0.1007
19.09	42.65	10.23	-0.1992
Aritmetický priemer($\bar{x}$)		10.030 mm	
Štandardná odchýlka			0.17866

X [mm]	**Y [mm]**	$x = X_n - X_{n-1}$	$x' - \bar{x}$
158.92	372.22		
149.14	372.33	9.78	0.1814
139.04	372.43	10.10	-0.1386
129.30	372.22	9.74	0.2214
118.92	372.32	10.38	-0.4186
109.21	372.1	9.71	0.2514
99.20	372.21	10.01	-0.0486
89.20	372.31	10.00	-0.0386
79.23	372.41	9.97	-0.0086
69.27	372.19	9.96	0.0014
59.34	372.29	9.93	0.0314
49.42	372.39	9.92	0.0414
39.52	372.18	9.90	0.0614
29.32	371.95	10.20	-0.2386
19.46	371.74	9.86	0.1014
Aritmetický priemer($\bar{x}$)		9.961 mm	
Štandardná odchýlka			0.17262

Tab 2) Tabuľky hodnôt súradníc X pozdĺž ľavého a pravého okraja.



Obr. 42) Ukážka vykreslenia grafu súradníc pri pohybe po osi x

V tabuľke sú zapísané hodnoty aritmetického priemeru kroku a štandardnej odchýlky pri pohybe po osiach. Ako je patrné z tabuľky aritmetické priemery pri pohybe po osi X a Y sú veľmi podobné.

Pohyb objektu po osi	Aritmetický priemer kroku[mm]	Štandardná odchýlka
X	9.996	0.179
Y	9.995	0.148

Tab 3) Súhrnná tabuľka merania pohybu po ose X a Y

Ako ďalšie meranie, sa objekt upevnil na pravítko a tak pomocou dvojice pravítok sa merali súradnice X a Y pri pohybe objektu po diagonále. Na zaznamenávanie pohybu sa využili funkciu aplikácie a to ukladanie trajektórie. Pomocou dvojice prvých(x_0, y_0) a posledných súradníc(x_1, y_1) sa vypočítala smernica priamky, ktorej rovnica je:

$$y = kx + q \tag{13}$$

kde,

$$k = \frac{(y_1 - y_0)}{(x_1 - x_0)} = \frac{(372{,}29 - 113{,}97)}{(19{,}83 - 169{,}5)} = -1{,}7259 \tag{14}$$

z toho vyplýva že $q = 405{,}95$.

Výsledná smernica priamky je rovná, $y = -1{,}7259x + 405{,}95$.

	Aritmetický priemer [mm]		Štandardná odchýlka	
Pohyb po diagonále	X	Y	X	Y
	0.183	0.315	0.220	0.378

Tab 4) Tabuľka výsledok merania pohybu o diagonále

V tabuľke v prílohe Tab 7) sú zapísane hodnoty súradníc X a Y pri pohybe po diagonále. Na základe rovnice sa prepočítali hodnoty súradníc Y´ podľa súradnice X a súradnica X´ podľa Y. Z takto dopočítaných dvojíc ($[X, X']$ a $[Y, Y']$) sa určila odchýlka nameranej od dopočítanej súradnice. V tabuľke Tab 4) sú výsledné hodnoty aritmetického priemeru a štandardnej odchýlky odchýlok od súradníc dopočítaných pomocou smernice priamky.

5.3.2 Zhrnutie experimentu

Z výsledkov experimentu môžeme usúdiť, že presnosť track-ovania objektu je dostatočne dobrá nato aby bola použitá na presne navigovanie. Pri pohybe objektom po osiach x po y o krok veľkosti 10 mm, sa dosiahol aritmetický priemer 9,996 mm. Šírka pásma kde sa pohybovali súradnice bola pre X ±0,179 mm a Y±0,148 mm. Pri pohybe po diagonále je šírka pásma kde sa pohybujú súradnice, pre X ±0,22 mm a Y±0,38 mm. Vzhľadom nato, že sa pohybovalo objektom ručne výsledok je uspokojivý, a dalo by sa povedať, že presnosť je priam zaručená pri zameriavaní polohy na celý 1 mm. Určovanie polohy sa uskutočňovalo na okrajoch snímky, kde je naviac prítomne skreslenie. Takže môžeme povedať, že kalibráciou kamery sa korigovali radiálne skreslenia natoľko dobre, že sa dosiahla dostatočná presnosť pri určovaní polohy objektu.

6 ÚLOHA DETEKCIA, TRACKOVANIE A ROZPOZNÁVANIE TVÁRE

V tejto kapitole je popísaná samotná realizácia úlohy. Pri jej realizácii boli implementované rôzne algoritmy, ktoré ponúka Matlab, ale aj vlastné naprogramované algoritmy. Vzhľadom nato, že pri preklade z anglického jazyka pojmi detekcia *(detection)* a rozpoznávanie *(recognition)* zamieňajú, je nutné pojmi vymedziť. Vymedzí sa teda detekcia tváre, ako proces hľadania tváre v obraze a pojem rozpoznanie tváre, ako proces identifikácie tváre s tvárou konkrétnej osoby.

Detekcia a track-ovanie objektov sú dôležité prvky v mnohých aplikáciách ako rozpoznávanie pohybu objektu, autonómne riadenie automobilu či biometrický bezpečnostný systém. Na detekciu a track-ovanie objektov, sú vhodne použité funkcie z toolboxov, ktoré ponúka Matlab. Ide o špecializované detektory alebo hotové algoritmy, ktoré majú univerzálne využitie. Je nimi možné detegovať rôzne geometrické tvary, rohy, auta, dopravné značky atď.

Algoritmy detekcie tváre zvyčajne extrahujú príznaky tváre a porovnávajú ich s klasifikátorom. Klasifikátor obsahuje databázu príznakov tvári, ktoré sa trénujú pomocou strojového učenia alebo neurónových sieti z trénovacej množiny tvári. Na základe zhody sa vyhodnotí či ide o tvár alebo nie. V tejto úlohe použijeme na detekciu tváre detektory, ktoré používajú modely klasifikácie schopné rozoznať celú tvár, jej časti (oči, ústa, nos) alebo vrchnú časť tela (hlavu a ramená) [33].

Pri track-ovaní tváre sa hľadajú charakteristické body v oblastí tváre, ktorých sa pozoruje pohyb na základe pohybu tváre.

Pri rozpoznávaní tváre sa používajú rôzne metódy biometriky. V tomto prípade použijeme metódu Eigenface. Túto metódu sa zvolila pre jej jednoduchosť a nenáročnosť na výpočetný výkon. Experimentom overíme jej funkčnosť a úspešnosť verifikácie v databáze tvári osôb.

V nasledujúcich podkapitolách si popíšeme navrhnuté grafické užívateľské prostredie, použité algoritmy a experiment.

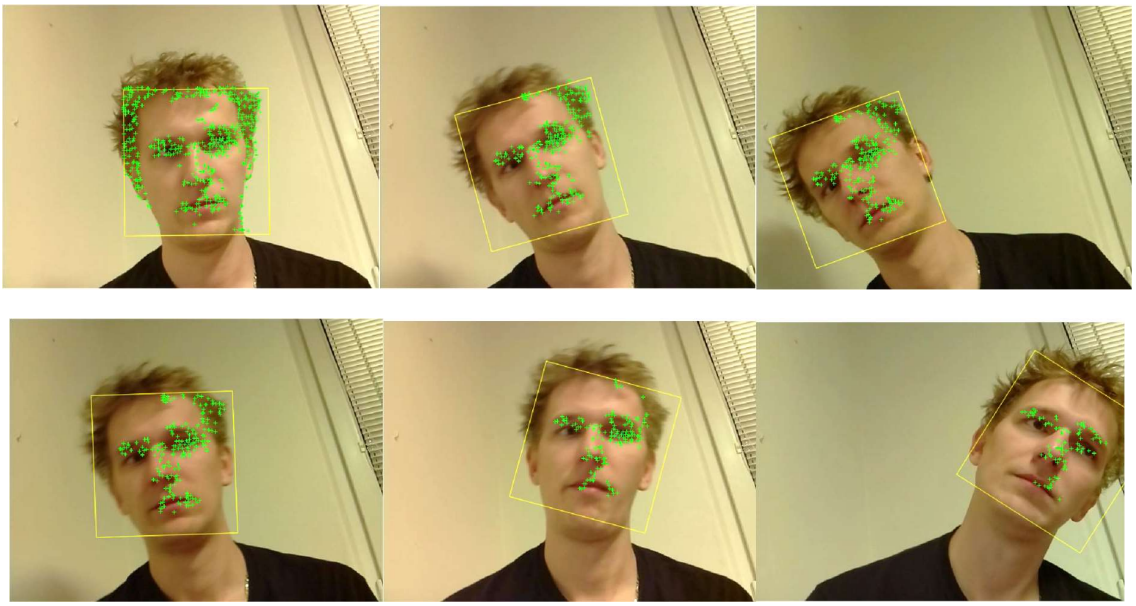
6.1 Popis grafického užívateľského rozhranie

Po spustení aplikácie sa zobrazí okno s názvom *Face Recognition,* ktoré obsahuje panely funkcií s tlačidlami a zobrazovaciu plochu. Panely sú pomenované podľa príslušných funkcií ako *Face Tracking*- track-ovanie tváre, *Camera rotation*-natáčanie kamery a *Face recognition*-rozpoznávanie tváre. Ukážka grafického rozhrania aplikácie je na obrázku Obr. 43).

Obr. 43) Grafické užívateľské rozhranie úlohy detekcia, tracking a rozpoznávanie tváre

Ako prvé je nutné zvoliť požadované rozlíšenie obrazu a vytvoriť spojenie s Raspberry Pi. Tak ako v úlohe č.1 sa pripojenie realizuje stlačením tlačidla *Connect Rpi.* Úspešnú komunikáciu a spustenie kamery, signalizuje červená kontrolná dióda na module kamery. Taktiež aplikácia skontroluje pripojenie a tlačidlo *Connect Rpi* zmení vzhľad na *Rpi Connected* zafarbené na zeleno. Tlačidlom *Disconnect* sa ukončí pripojenie Rpi a vypína kamera. Pri zmene rozlíšenia je potrebné opakovať kroky odpojenia a pripojenia Rpi. Tlačidlo *Start,* spustí prenos snímaného obrazu kamerou do zobrazovacej plochy aplikácie. Ukážka interakcie je na obrázku Obr. 26).

Hlavnou funkciou panelu *FaceTracking* je track-ovanie pohybu tváre. Panel obsahuje iba tlačidlo *Start face tracking,* ktorým sa aktivuje algoritmus detekcie tváre, ktorú ohraničí žltým obdĺžnikom. Vo vnútri ohraničenia tváre sa detegujú rysy tváre, ktoré reprezentujú zelené body. Pre každý z bodov s predchádzajúceho obrazu, sa vyhľadávajú korešpondujúce body v aktuálnom obraze. Pomocou funkcie geometrickej transformácie sa odhadne rotácia a translácia, medzi predchádzajúcimi a novými bodmi. Track-ovanie bodov pokračuje až kým všetky body nezaniknú. K zániku dochádza pri nadmernej zmene obrazu napríklad pri náhlom otáčaní hlavy, kedy niektoré rysy tváre už nie je viac vidieť. Ukážka track-ovanie tváre je na obrázku Obr. 44).



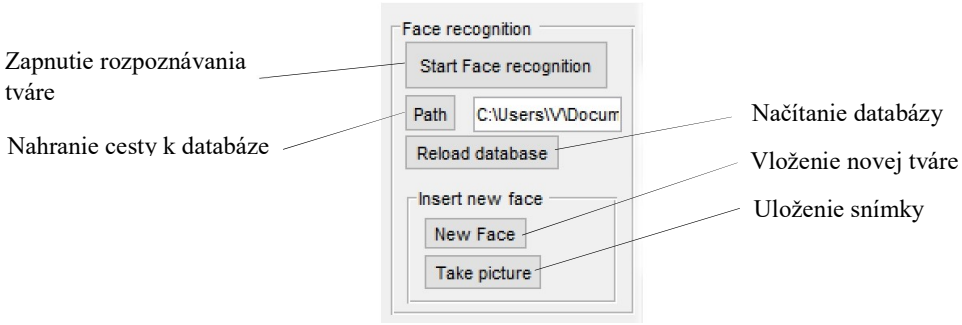
Obr. 44) Ukážka track-ovania tváre

Panel *Camera rotation* obsahuje funkcie spustenia servo pohonov, natáčanie kamery a nastavenia parametrov pre PID reguláciu servopohonov. Panel ma rovnaké ovládanie a podobné funkcie ako v úlohe detekcia a track-ovanie farebného objeku. Panel je zobrazený na obrázku.

Pri aktivovaných tlačidlách *Start, Start face tracking* a tlačidla *Start servos* sa spustí sledovanie polohy tváre. Pri pohybe tváre servo pohony otáčajú kamerou okolo vertikálnej a horizontálnej osi, tak aby optická os kamery prechádzala ťažiskom obdĺžnika, ktorý ohraničuje tvár človeka. Optickú os kamery reprezentuje symbol *„X“* označený fialovou farbou a ťažisko obdĺžniku tváre symbol *„O“,* ktorý je bielej farby. Regulácia otáčania seriev je realizovaná pomocou PID algoritmu. Hodnoty proporcionálnej zložky Kp, derivačnej Kd a integrálnej Ki, je možné nastavovať v príslušných kolónkach panelu. Nastavenie regulácie polohy, ktoré fungovalo dostatočne dobre, je v tabuľke Tab 1) Algoritmus PID je popísaný v prílohe. Ukážka sledovania polohy tváre je na obrázku Obr. 45)

Obr. 45) Sledovanie tváre natáčaním kamery

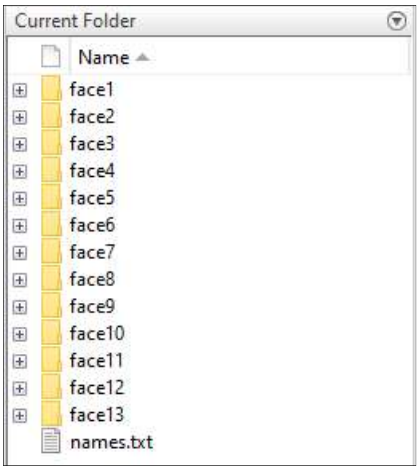
V paneli *Face recognition* je možné spustiť algoritmus detekcie a rozpoznávania tváre a možnosť nahrať vytvorenú databázu alebo pridať novú tvár do databázy.Vzhľad panela je na obrázku Obr. 46).



Obr. 46) Popis funkcii panelu Face recognition

Ako prvé je potrebné nastaviť cestu k databáze tvári. Tlačidlom *Path* sa otvorí dialógové okno výberu cesty k priečinku. Túto cestu bude aplikácia používať na nahrávanie alebo ukladanie tvári do databázy spolu s menami osôb v databáze. Nahrávanie databázy spustíme tlačidlom *Reload database*. Vkladanie tváre sa realizuje stlačením tlačidla *New face*. Tlačidlo vyvolá dialógové okno. Do okna vložíme meno osoby, ktorej tvár ideme vkladať do databázy. Mená osôb sú ukladané do textového súboru s názvom *names.txt*. Z obrazového prenosu algoritmus detekcie tváre, lokalizuje tvár pred kamerou. Následne sa vystrihne miniatúra snímky, ktorá sa

zobrazí do ľavého horného rohu zobrazovacej plochy. Snímky tváre sa obnovujú každým cyklom a tak je možné následne uložiť vhodné snímky do databázy tlačidlom *Take picture*. Snímky sa ukládajú do priečiku s názvom *faceX,* kde *X* je poradové číslo v databáze. Obsah priečinku databázy je na obrázku Obr. 47)



Obr. 47) Štruktúra a obsah priečika databázy tváry

Aktivovaním tlačidiel *Start* a *Start face recognition* sa spustí algoritmus detekcie a rozpoznávania tváre. Po úspešnej detekcii sa tvár ohraničí žltým obdĺžnikom. Tvár sa vystrihne a porovná s tvárami v databáze. Hľadaná a nájdená tvár z databázy, sa spolu s menom osoby zobrazí v externom okne. Ukážky okien s výsledkami rozpoznávania tváre je na obrázku Obr. 48)

Obr. 48) Ukážka okien výsledku rozpoznania tváre

Pred ukončením aplikácie je nutné odpojiť Rpi tlačidlom *Disconnect.*

6.2 Popis použitých algoritmov

V tejto kapitole sú popísané dôležité detaily použitých algoritmov spolu s funkciami, ktoré boli použite pri realizácii úlohy detekcia, track-ovania a rozpoznávania tváre.

6.2.1 Detekcia tváre

Na detekciu tváre je použitý detektor kaskádových objektov, *vision.CascadeObjectDetector*. Detektor používa detekčný algoritmus Viola-Jones a trénovaný klasifikačný model na detekciu. Je štandardne konfigurovaný na detekciu tvárí, ale môže byť natrénovaný pre iné typy objektov. Fungovanie detektoru je popísané v Príloha B - Kaskádový detektor Viola Jones.

Tvár sa deteguje spomínaným detektorom *(faceDetector)* vo vyšetrovanom obraze (*frame*) použitím metódy *step*. Ohraničenie tváre sa uloží do premennej *bbox* .

```
%Detektor tváre
faceDetector=vision.CascadeObjectDetector('MinSize',[150 150] );
%Detekcia a ohraničenie oblasti tváre
bbox=step(faceDetector,frame);
```

6.2.2 Track-ovanie tváre

Problém track-ovania tváre rozdelíme na tri samostatné problémy:

- **Detekcia tváre** (popísaná v kapitole 6.2.1)
- **Identifikácia tvárových prvkov**
- **Track-ovanie prvkov tváre**

Identifikácia tvárových prvkov

Keď detektor lokalizuje tvár tak v nasledujúcom kroku sa identifikujú body, ktoré je možné spoľahlivo sledovať. K tomu je využitý *detektor Shi Tomasi,* ktorý pomocou algoritmu minimálnych vlastných čísel, nájde obrysové body tváre. Popis princípu je v Príloha C - Detektor Shi Tomasi, KLT tracker, Blob analýza a binarizácia obrazu. Nato je použitá metóda *detectMinEigenFeatures*, ktorá v snímke v odtieni sivej, nájde obrysové body vo vnútry oblasti ohraničenia tváre *(bbox)*. Vytvorený objekt *points* obsahuje informácie o detegovaných bodoch. Na prevod farebnej snímky *(frame)* do odtieňa sivej sa použije metóda *rgb2gray(frame).*

```
%Detekcia bodov na tváry
points = detectMinEigenFeatures(rgb2gray(frame), 'ROI', bbox);
```

Track-ovanie prvkov tváre

Na sledovanie prvkov tváre v priebehu času je použitý algoritmus Kanade-Lucas-Tomasi (KLT). Použitie detektoru kaskádových objektov na každom obraze, je výpočtovo náročne. Tak isto môže zlyhať pri detekcii, keď osoba otočí alebo nakloní hlavu. Toto obmedzenie pochádza z typu trénovaného klasifikačného modelu používaného na detekciu. Tvár je rozpozná len raz pomocou detektoru a potom KLT algoritmus sleduje tvár počas

pohybu.Na track-ovnie bodov je použitý tracker s názvom *vision.PointTracker,* ktorý využíva spomínaný KLT algoritmus s dvojsmernou chybou. Princíp je vysvetlený v Príloha C - Detektor Shi Tomasi, KLT tracker, Blob analýza a binarizácia obrazu.

```
%Vytvorenie trackeru
pointTracker = vision.PointTracker('MaxBidirectionalError', 2);
% Inicializácia trakeru s pozíciou počiatočných bodov
points = points.Location;
initialize(pointTracker, points,frame);
```

V novom obraze *(frame)*, pomocou tracker-u *(pointTracker),* sú odhadnuté nové body a ich validita je zapísaná v premennej *(isFound).* Následne z množiny všetkých bodov *(points)* sa vyberú len body *(visiblePoints),* ktoré sú validné a teda vhodné track-ovať.
Bodom z predošlého obrazu *(oldPoints)* sa skontroluje validita a body, ktoré už nevyhovujú sa zmažú. Selekcia bodov je zapísaná do premennej *(oldInliers).*

```
%Lokalizácia nových bodov porovnanie so starými
[points, isFound] = step(pointTracker,frame);
visiblePoints = points(isFound, :);
oldInliers = oldPoints(isFound, :);
```

Pomocou funkcie *estimateGeometricTransform* sa dopočíta odhad translácie, rotácie, a škálovania *(xform)* medzi bodmi z predchádzajúcej snímky a bodmi na aktuálnej snímke. Princíp je bližšie popísaný v kapitole 4.3.1. Výstupom je odhad transormácie *(xform)* a uloženie bodov, ktoré budú použité v nasledujúcom obraze.

```
%Odhad geometrickej transformacie medzi starými a aktuálnymi bodmi
[xform, oldInliers, visiblePoints] = estimateGeometricTransform(oldInliers,
visiblePoints);
```

Odhad transformácie poslúži pri prepočítaní súradníc *(transformPointsForward)* ohraničenia tváre*(bbox).*To zaručí že sa pôvodné ohraničenie tváre rotuje podľa otočenia hlavy osoby.

```
%Tranformácia ohraničenia tváre
bboxPoints = transformPointsForward(xform, bboxPoints);
```

Ako posledné je reset bodov, ktoré budú použité v ďalšom obraze.

```
%reset bodov pre ďalší frame
oldPoints = visiblePoints;
setPoints(pointTracker, oldPoints);
```

Track-ovanie funguje až dovtedy, kým nezaniknú všetky body. Minimálny počet bodov je dva, čo je nutná podmienka geometrickej transformácie.

Track-ovanie tváre otáčaním kamery
Cieľom otáčania kamery pri track-ovaní tváre je pozorovať polohu tváre v pozorovanej scéne, tak aby tvár bola v optickej osi kamery. Výsledný efekt pripomína osobou, ktorá sa snaží pozorovaný objekt nespustiť z očí. Otáčaním kamery okolo vertikálnej a horizontálnej osi dosiahneme prienik optickej osi kamery a ťažiska oblasti tváre *(bbox)*. Otáčanie servo pohonov je reguluje PID algoritmom popísaným v Príloha E - PID regulácia servopohonu.

6.2.3 Rozpoznávanie tváre

Súčasti problému rozpoznávania tváre sú:

- **Detekcia tváre** (popísaná v kapitole 6.2.1)
- **Spracovanie oblasti tváre**
- **Porovnanie tváre v databáze**

Spracovanie oblasti tváre

Snímka *(frame)* na ktorej sa úspešne detegovala tvár sa spracováva len v oblasti tváre. Oblasť tváre vyznačujú súradnice premennej *bbox.* Podľa nich funkciou *imcrop* vystrihneme oblasť tváre.

```
%Vystrihnutie tvare
face = imcrop(frame,bbox);
```

Tvár osoby môže byť vzdialená od objektívu kamery v rôznej vzdialenosti a preto aj vystrihnutá snímka tváre má rôzne rozmery. Nasleduje úprava rozmerov tváre tak aby rozlíšenie snímky súhlasilo so snímkami tvári v databáze. Snímka tváre v databáze má štvorcové rozmery *(picSize)*. Preto vypočítame pomer *(scaleFactor)* medzi rozmerom snímky v databáze*(picSize)* a vystrihnutej snímky (*size(face,1)*). Funkcia *size(face,1)* vráti rozmer vystrihnutej snímky tváre. Takto môžeme vystrihnutú snímku bez straty pomerov strán, zmenšiť alebo zväčšiť podľa pomeru *scaleFactor* do podobných rozmerov snímky*(picSize).*Pri procese je nevyhnutný aby nedošlo ku skresleniu obrysov tváre pri zmenšovaní/zväčšovaní. Nakoniec sú snímky orezané na presný rozmer. Nakoniec snímku tváre *(crpface),* ktorá je v RGB farbách, previesť do odtieňu šedej *(gface)*. To dosiahneme funkciou *rgb2gray*. Má to súvis práve s algoritmom rozpoznávania tváre.

```
%Uprava snímky tvare na požadovanu veľkost
scaleFactor = picSize/size(face,1);
face = imresize( face,scaleFactor);
crpface = imresize(face,picSize);
%Prevod obrazu z RGB do grayscale
gface=rgb2gray(crpface);
```

Porovnanie tváre v databáze

Pre porovnanie tváre je vytvorená funkcia *recognizeface,* ktorá snímku tváre porovná v databáze. Pre rozpoznávania tváre používa metódu Eigenface.

Funkcia *recognizeface* má vstupy *fcdata* a *gface* a výstupy *faceNumber* a *matchedFace.* Vstup *fcdata* je spracovaná databáza tvári a *gface* je hľadaná tvár .Výstupom z metódy je poradové číslo *(faceNumber)* a snímka tváre z databázy, ktorá je najviac podobná hľadanej tváre*(gface).*

Databáza(*fcdata)* je spracovaná podľa algoritmu Eigenface.

```
%Vyhľadanie tváre v databáze
[faceNumber,matchedFace]=recognizeface(fcdata,gface);
```

Algoritmus rozpoznávania Eigenface vo svojej podstate pozostáva z maticových operácii. Pre pracnosť popisu nebude rozpísaný krok po kroku. Jeho princíp je vysvetlený v Príloha A - Princíp Eigenface metódy.

Obr. 49) Ukážka vstupnej tváre a „eigenface“

6.3 Experiment

Experiment pozostáva v testovaní algoritmu rozpoznávania tváre. Test sa uskutoční pri opakovanom rozpoznávaní istej tváre pri rôznych uhloch natočeniach hlavy vzdialenostiach od objektívu kamery. Ako databáza bude vytvorená vlastná databáza tvári osôb. Hodnotiť sa bude úspešnosť správneho rozpoznania tváre. Bude vytvorená štatistika

6.3.1 Priebeh experimentu

Test metódy rozpoznávania tvári Eigenface prebehne na vzorke piatich ľudí. Do databázy sa vloží 10 snímkou tváre na tmavom pozadí. Úspešnosť budeme zaznamenávať pri 20-tich pokusoch, kedy sa tvár ukáže pred kamerou.

Potom ako sa vytvorila vstupná databáza tvári pomocou vytvorenej aplikácie, začal test metódy Eigenface. Osoba pred kamerou každým pokusom mierne otáčala hlavou a objektív kamery sa zakrýval bielou clonou aby algoritmus detekcie znova detegoval tvár, ktorá sa bude rozpoznávať. Výsledky jednotlivých pokusov sa zaznamenávali do tabuľky kde úspech sa značil ako „1“ a neúspech „0“. Výsledky testu sú zaznamenané v tabuľke Tab 5) .

	Vincent	Martin	Vladimír	Artur	Michal
1	0	1	1	1	1
2	1	1	1	0	1
3	1	1	1	0	1
4	1	1	1	1	1
5	0	1	0	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	0	1	1
9	1	1	0	1	1
10	0	0	0	1	1
11	1	0	1	1	1
12	0	0	1	1	0
13	1	1	1	1	0
14	1	1	1	1	1
15	1	1	1	1	0
16	0	0	1	1	1
17	1	0	1	1	0
18	1	1	0	1	0
19	1	1	0	1	1
20	0	1	1	0	1
Úspešnosť	70%	75%	70%	85%	75%
Celková úspešnosť		75%			

Tab 5) Tabuľka výsledkov rozpoznávania tváre

6.3.2 Zhrnutie experimentu

Výsledok testu ukázal celkovú úspešnosť 75% percent. Vzhľadom na svetelné podmienky v miestnosti a relatívnosti natočení hlavy a vzdialenosti od kamery je výsledok uspokojujúci. Pre zvýšenie úspešnosti by pomohlo zvýšenie počtu vzoriek snímkou jednotlivých tvári pri rôznych uhloch a tak zvýšiť šance na správnu identifikáciu. Ďalej je možné implementovať ďalšie metódy rozpoznávania tváre, ako rozpoznanie farebného tónu pleti, a tak priblížiť úspešnosť k 99.9 % [33].

7 ZÁVER

Témou tejto diplomovej práce bolo v prostredí Matlab, naprogramovať dve úlohy pre Raspberry Pi s využitím kamery. Ako demonštračné úlohy, boli navrhnuté detekcia a trackovania farebných objektov a detekcia a rozpoznávanie tváre. Začiatok práce bol zameraný na popis hardware a software vybavenia použitého pri realizácii úloh.

V kapitole hardware výbavy sú spomenuté použité prvky a ich technické parametre. Najprv je popísaná platforma Raspberry Pi, ktorá patrí k populárnym vstavaným systémom vďaka výkonu, kompaktnosti a hlavne cene. Ďalej bola popísaná kamera, ktorá zabezpečuje snímanie obrazu. Pre jej stabilitu a zjednodušenie manipulácie bol navrhnutý držiak. Ten umožňuje kamerový modul namontovať na servo pohony tak, aby bolo možné natáčať kameru okolo jej horizontálnej a vertikálnej osi.

Ako softwarová výbava bol použitý program Matlab, ktorý je populárny ako v inžinierskej tak aj vo vedeckej sfére. S množstvom funkcii a svojimi prídavnými toolboxami preukazuje svoju robustnosť v mnohých oblastiach pri realizácii úloh tejto diplomovej práce. V úlohách sa implementovali mnohé funkcie, obsiahnuté v tooboxch pre Raspberry Pi a počítačové videnie. S ich pomocou, je program Matlab schopný používať platformu Raspberry Pi ako vstupno-výstupné zariadenie, spolu s jeho perifémymi zariadeniami.

V samotnej teórii sú popísané základné kroky, ktoré sú spomínané v literatúrach venovaných počítačovému videniu. Krokmi je popísaný celý proces, ako sa získava obraz pomocou kamery, až po metódy získavania hľadaných prvkov v obraze. Popísaný je proces digitalizácie, ktorá je nutná pre ďalšie spracovávanie číslicovou technikou. Spomenuté sú dôležité techniky predspracovania obrazu, a princíp klasifikácie prvkov v obraze. Taktiež v prílohe diplomovej práce sú spomenuté princípy fungovania použitých algoritmov a detektorov.

V kapitolách realizovaných úloh, bola popísaná funkcionalita vytvorených aplikácii, detaily použitých algoritmov zaujímavých z pohľadu počítačového videnia, a experimenty na overenie funkčnosti použitých metód.

V prvej úlohe je realizovaná detekcia a track-ovanie farebných objektov. Detekcia farieb využíva vhodné techniky spracovania obrazu, ktoré sú založené na extrakcii farebnej zložky z RGB obrazu. Pomocou blob analýzy sa lokalizoval centroid farebného objektu, ktorý bol následne využitý ako bod na track-ovanie v pozorovanom priestore kamery. V experimente sa overila presnosť pri určovaní polohy objektu, ktorú v celom rozsahu pozorovanej plochy bolo možné určiť s presnosťou na jeden milimeter. Táto presnosť sa dosiahla vďaka kalibrácii kamery, ktorá eliminovala skreslenia, ku ktorým dochádza v obraze pri snímaní scény kamerou.

Druhá úloha sa zameriavala na detegovanie a rozpoznanie tváre. Na detegovanie tváre bol použitý kaskádoví detektor Viola-Johnes, obsiahnutý v toolboxoch Matlab-u, ktorý bol predtrénovaný a tak umožnil rýchlu implementáciu. Následne boli na tvári, detektorom detegovali body, ktoré KLT algoritmus track-oval pri pohybe hlavy. Taktiež pomocou rotácie kamery servami, bolo možné track-ovať tvár človeka aj v rozšírenom pozorovanom priestore. Servopohony pomocou diskrétneho PID algoritmu centrovali optickú os kamery a ťažisko tváre, a týmto sa dosiahlo „stalker efektu“. Na rozpoznanie tváre sa použila známa metóda s názvom Eigenface. Experimentom sa vyskúšala jej úspešnosť rozpoznania tváre, ktorá bola

75%. Keď že išlo o základnú metódu bez vylepšení, využitím viacerých techník biometriky by bolo možné, zvýšiť presnosť rozpoznávania tváre a priblížila by sa úspešnosti 99.9% [33].

Vytvorené aplikácie, ktoré združujú implementované funkcie každej z úloh, môže užívateľ voľne použiť, pokiaľ má spomínaný hardware, vo vlastných projektoch. Aplikácie alebo ich časti, môžu byť predlohou pre navigáciu robotov v priestore, kamerové a bezpečnostné systémy alebo systémy skúmajúce pohyb objektov.

V priebehu práce som nahliadol do mnohých technických oblastí, z ktorých som nadobudol nové vedomosti a tak rozšíril svoje obzory. Zdokonalil som sa v programovaní v Matblab-e a zoznámil s platformou Raspberry Pi, ktorú som si veľmi obľúbil a mám v pláne s ňou naďalej pracovať. Platforma potvrdila svoje vlastnosti a silné stránky, aj pri realizácii úloh tejto diplomovej práce, ktorá je bodkou za mojím snažením počas vysokoškolského štúdia.

ZOZNAM POUŽITÝCH ZDROJOV

[1] Embedded systémy. muni.cz [online]. [cit. 2017-05-25]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2005/xmrstik.htm

[2] Úvod do embedded systémů. DPS Elektronika od A do Z [online]. [cit. 2017-05-25]. Dostupné z: http://www.dps-az.cz/vyvoj/id:6173/uvod-do-embedded-systemu

[3] História Raspberry Pi. Nova Digital Media.com [online]. Nova Digital Media, 2015 [cit. 2017-05-14]. Dostupné z: http://novadigitalmedia.com/history-raspberry-pi/

[4] UPTON, Eben a Jakub GORNER. Raspberry Pi uživatelská příručka. Brno: Computer Press, 2013. ISBN 978-80-251-4116-8.

[5] Raspberry pi camera modul. Raspberrypi.org [online]. UK: RASPBERRY PI FOUNDATION, 2016 [cit. 2017-05-13]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/camera/

[6] RASPBERRY PI 3 MODEL B. RASPBERRY PI FOUNDATION [online]. RASPBERRY PI FOUNDATION, 2015 [cit. 2017-05-14]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[7] GPIO: MODELS A+, B+, RASPBERRY PI 2 B AND RASPBERRY PI 3 B. RASPBERRY PI FOUNDATION [online]. RASPBERRY PI FOUNDATION, 2015 [cit. 2017-05-14]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[8] SG90 9 g Micro Servo. ServoDatabase.com [online]. ServoDatabase, 2009 [cit. 2017-05-14]. Dostupné z: http://www.servodatabase.com/servo/towerpro/sg90

[9] E-BOOK Umíte si dobře vybrat servo? ServoDatabase.com [online]. JPMODELS, 2009 [cit. 2017-05-14]. Dostupné z: http://www.modelorlicko.com/phocadownloadpap/03-rady-navody/prislusenstvi/Umite-vybrat-servo.pdf

[10] Matlab – ako to začalo Foltin Martin [online]. posterus.sk, 2009 [cit. 2017-05-14]. Dostupné z: http://www.posterus.sk/?p=44

[11] MATLAB & SIMULINK [online]. www.humusoft.cz [cit. 2017-05-14]. Dostupné z: http://www.humusoft.cz/matlab/

[12] 7 dôvodov prečo MATLAB je najjednoduchšie a najproduktívnejšie prostredie pre inžinierov a vedcov [online]. mathworks [cit. 2017-05-14]. Dostupné z: https://www.mathworks.com/products/matlab/why-matlab.html?s_tid=hp_brand_whymatlab

[13] Výhody matlabu [online]. myartve.net [cit. 2017-05-14]. Dostupné z: http://www.myartve.net/vyhody-matlabu/

[14] *Hardware Support* [online]. mathworks [cit. 2017-05-14]. Dostupné z: https://www.mathworks.com/hardware-support/raspberry-pi-matlab.html

[15] *Porovnanie Raspberry Pi 3 a Raspberry Pi 2* [online]. rs-online.com [cit. 2017-05-14]. Dostupné z: https://www.rs-online.com/designspark/raspberry-pi-3-to-raspberry-pi-2-comparison?_locale=eng

[16] Cleave Moler. *Computer Hope* [online]. [cit. 2017-05-25]. Dostupné z: https://www.computerhope.com/people/cleve_moler.htm

[17] Computer Vision System Toolbox. Mathworks [online]. [cit. 2017-05-25]. Dostupné z: https://www.mathworks.com/products/computer-vision.html

[18] Build standalone applications from MATLAB programs. Mathworks [online]. [cit. 2017-05-25]. Dostupné z: https://www.mathworks.com/products/compiler.html

[19] RNDr. Elena Šikudová, PhD, RNDr. Zuzana ˇ Cerneková, PhD a Ing. Vanda Benešová, CSc. Počitačové videnie. Detekcia a rozpoznávanie objektov. Wikina, Livornská445, 109 00 Praha 10, 2011. ISBN 978-80-87925-06-5.

[20] Počítačové videnie v praxi. Saske.sk [online]. Ústav experimentálnej fyziky SAV, 2016 [cit. 2017-05- 25].
Dostupné z: http://home.saske.sk/~tomori/Downloads/Poc_videnie/PV_2016.pdf

[21] Václav Hlavač a Miloš Sedláček. *Zpracováni signálu a obrazu*. 1. Praha: ČVUT, 2002. ISBN 8001021149.

[22] Fotomobily: snímací čipy CMOS vs. CCD. DIGIMANIA [online]. [cit. 2017-05-25]. Dostupné z: http://www.digimanie.cz/fotomobily-snimaci-cipy-cmos-vs-ccd/2885

[23] University of Utah: CS6640 Image Processing Report. Utah university [online]. [cit. 2017-05-25]. Dostupné z: http://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html

[24] Minimum, maximum and median filters. Graphics Mill [online]. [cit. 2017-05-25]. Dostupné z: https://www.graphicsmill.com/docs/gm/minimum-maximum-median-filters.htm

[25] Matlab Code for Image Retrieval. Pantech Solutions [online]. [cit. 2017-05-25]. Dostupné z: https://www.pantechsolutions.net/image-processing-projects/matlab-code-for-image-retrieval

[26] Obrazové snímače CCD vs. CMOS. Netcam [online]. [cit. 2017-05-25]. Dostupné z: http://www.netcam.cz/encyklopedie-ip-zabezpeceni/obrazove-snimace-ccd-cmos.php

[27] Colors – RGB and CMYK. *Offset printing technology | Offset lithography* [online]. [cit. 2017-05-25]. Dostupné z: http://www.offsetprintingtechnology.com/2015/colors-rgb-and-cmyk/

[28] Kalibrácia vnútorných parametrov kamery [online]. Bratislava: STUBA, 2014 [cit. 2017-05-29]. Dostupné z: http://www.fei.stuba.sk/docs//2014/autoreferaty/autoreferat_Letanovska.pdf

[29] Face Recognition using eigenfaces technique. Medium.com [online]. [cit. 2017-05-25]. Dostupné z: https://medium.com/@devalshah1619/face-recognition-using-eigenfaces-technique-f221d505d4f7

[30] Eigenfaces for Face Detection/Recognition. The Vision, Dynamics and Learning Lab [online]. [cit. 2017-05-25]. Dostupné z: http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf

[31] *Comparison Of Viola-Jones and Kanade-Lucas-Tomasi Face Detection Algorithms* [online]. ORIENTAL JOURNAL OF COMPUTER SCIENCE & TECHNOLOGY, 2017, (1) [cit. 2017-05-29]. ISSN 0974-6471. Dostupné z: http://www.computerscijournal.org/vol10no1/comparison-of-viola-jones-and-kanade-lucas-tomasi-face-detection-algorithms/

[32] Eigenfaces. JMC Spot .com [online]. [cit. 2017-05-29]. Dostupné z: http://jmcspot.com/Eigenface/

[33] Využití detektoru Viola-Jones pro lokalizaci obličeje a očí v barevných obrazech. Elektrorevue [online]. [cit. 2017-05-29]. Dostupné z: http://elektrorevue.cz/cz/download/vyuziti-detektoru-viola-jones-pro-lokalizaci-obliceje-a-oci-v-barevnych-obrazech/.

[34] Shi-Tomasi Corner Detector & Good Features to Track. OpenCV [online]. [cit. 2017-05-25]. Dostupné z: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html

[35] Detection and Tracking of Point Features. Clemson [online]. [cit. 2017-05-25]. Dostupné z: https://cecas.clemson.edu/~stb/klt/tomasi-kanade-techreport-1991.pdf

[36] Blob Analysis. Adaptive-vision [online]. [cit. 2017-05-25]. Dostupné z: http://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/BlobAnalysis.html

[37] Blob Analysis. *Viscotech* [online]. [cit. 2017-05-25]. Dostupné z: http://www.visco-tech.com/english/technology/existence/

[38] What Is Camera Calibration? MathWorks [online]. [cit. 2017-05-25]. Dostupné z: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[39] PID regulátory – jejich vlastnosti, modifikace a číslicová implementace. Tul.cz [online]. [cit. 2017-05-25]. Dostupné z: http://www.fm.tul.cz/esf0247/index.php?download=822.

ZOZNAM, OBRÁZKOV

ZOZNAM, TABULIEK

ZOZNAM PRÍLOH

PRÍLOHA A - PRINCÍP EIGENFACE METÓDY

Eigenface metóda používa PCA (Principal Component Analysis-Eigenface) čo je štatistická metóda redukcie príznakov. PCA používa vektory tvári odvodených z kovariačnej matice pravdepodobnostnej distribučnej funkcie k vytvoreniu šablóny pre vhodne porovnanie.

Uvažujeme trenováciu množinu obrázkov *Z,* ktoré majú rovnaké rozmery *a x b*. Z týchto obrázkov zostavíme maticu o počte riadkov rovnému počtu obrázkov *Z* a počet stĺpcov rovný *axb*. Obrázky tvári sú si podobné v mnohých rysoch a preto nie sú náhodné distribuované (existuje medzi nimi nenulová korelácia). Hlavnou myšlienkou PCA je nájsť vektory, ktoré najlepšie popisujú distribúciu obrázkov tvári v celom priestore obrázkov. Tieto nájdene vektory dimenzii *n* definujú podpriestor, ktorý nazývame priestor obrázkov tvári (eigenspace). Sú to lineárne kombinácie pôvodných vektorov a tvoria bázu eigespace. Tieto vektory sú vlastnými vektormi kovariačnej matice korešpondujúcej k pôvodnému obrázku a pretože sú podobné obrázkom tvári, nazývame ich eigenfaces [29],[32].

Metóda je zložená z niekoľkých krokov.

Obrázok rozmeroch a x b sa prevedie na riadkový vektor $x = a * b$

Na tomto vektory spočítame priemer podľa vzorca.(19):

$$u = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{15}$$

Spočíta sa kovariačná matica, ktorá ma tvar podľa rovnice (16):

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - u)\,(x_i - u)^T \tag{16}$$

Nájdenie vlastných vektorov a vlastných čísel kde plati rovnosť(19):

$$\lambda v_i = S v_i \tag{17}$$

Takto získame charakteristický vektor V=$\{v_1, v_2, \ldots \ldots v_n\}^T$, nazývaný tiež eigen vektor.

Rozpoznávanie tváre je potom hľadanie podpriestoru zodpovedajúceho ľudskej tvári. V reálnej aplikácii je táto metóda reprezentovaná Karhumen-Loeve transformáciou, ktorá dala základ pre metódu detekcie a normalizácie tváre zvanú "eigenface“. Výhodou metódy je rýchlosť výpočtu a jednoduchosť [29],[30]. Ukážka Eigenface metódy je na obrázku Obr. 50).

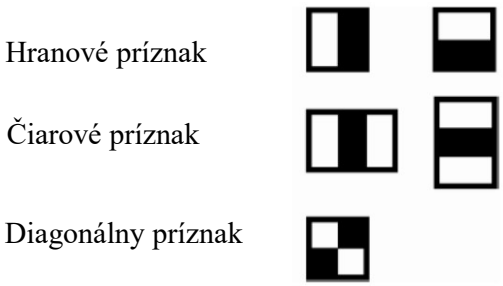
Obr. 50) Ukážka metódy Eigenface, vlastný vstupný obrázok (vľavo) a Eigenface(vpravo)

PRÍLOHA B - KASKÁDOVÝ DETEKTOR VIOLA JONES

Objektový detektor Viola-Jones bol prvý krát predstavený P. Violou a M. Jonesom v roku 2001. Jedná sa o detektor objektov pracujúcich so šedo-tónovými obrazmi, ktorý pozostáva z troch základných častí: integrálneho obrazu, Haarovy vlnky a klasifikačný algoritmus AdaBoost. Výhodou tohto detektora je rýchlosť, dostatočná spoľahlivosť a značná nezávislosť na osvetlení a veľkosti sledovaného objektu [31],[33]. Činnosť detektora je znázornená na obrázku Obr. 52)

AdaBoost je klasifikačný algoritmus vychádzajúci zo strojového učenia. Cieľom metódy je zlepšenie klasifikačnej presnosti ľubovoľného algoritmu strojového učenia. Základom je vytvorenie viac klasifikátorov označovaných ako slabí žiaci. Tieto klasifikátory vzniknú pomocou výberu vzoriek zo základného trénovacieho súboru. Prvý klasifikátor má len o málo väčšiu presnosť než je presnosť odhadu (tj viac ako 50% v prípade dvojstavového klasifikátora). Postupne sú ďalšie klasifikátory pridávané s podobnou mierou presnosti, čím sa vygeneruje súbor klasifikátorov označených jako silný žiaci. Jeho celková klasifikačná presnosť je ľubovoľne vysoká vzhľadom na vzorky v trénovancej množine. Takto je klasifikácia zosilnená (boosted) [33].

Snahou detektoru Viola-Jones je získať veľké množstvo jednoduchých príznakov s minimálnymi výpočtovými nárokmi. Takýto typ príznakov sú príznaky založené na princípe Haarových vlniek (Haar-like features). Ukážka na obrázku Obr. 51). Hodnota príznaku sa vypočíta ako súčet obrazových bodov zodpovedajúcich svetlým časti, z ktorých je odpočítaná suma pixelov zodpovedajúcich tmavých častí. Tieto vlnky môžu byť vytvorené dvomi (hranový príznak), tromi (čiarový príznak) alebo štyri (diagonálny príznak) obdĺžnikovými oblasťami. Jednotlivé príznaky sa použijú na celý vstupný obraz, pričom súčasne dochádza k zmeny veľkosti jednotlivých príznakov

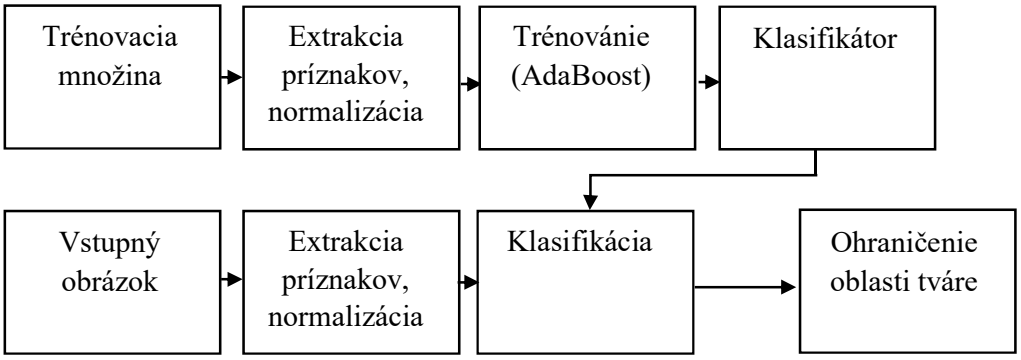


Obr. 51) Príznaky podobné Haarovej vlnke [33]

Integrálny obraz slúži na rýchlu a efektívny výpočet hodnôt jednotlivých príznakov vstupného obrazu. Aby pre každú hodnotu príznaku vstupného obrazu nemusela byť počítaná suma hodnôt pixelov zodpovedajúcich danému príznaku, obraz je preložený do reprezentácie integrálneho obrazu, kde každý bod tohto obrázku zodpovedá súčtu hodnôt všetkých predchádzajúcich bodov. Výpočet hodnôt jednotlivých príznakov sa podstatne zjednoduší, pretože výpočet súm ľubovoľného obdĺžnika v obraze postačí dve operácie sčítania a jedna operácia odčítania [33].

Aby bolo možné detegovať určitý objekt v obraze je najprv nevyhnutné vytvoriť trénovaciu množinu obrázkov obsahujúcich pozitívne vzory objektu a negatívne vzorové pozadia(všetko čo nie je sledovaným objektom). Všetky vzory by mali mať rovnakú veľkosť a zároveň by mala dosiahnuť čo najmenšie hodnoty (samozrejme za predpokladu dostatočného optického rozlíšenia sledovaného objektu), a to z dôvodu extrémnej výpočtovej náročnosti v procese trénovania. Pomocou týchto vzorov je trénovaný klasifikačný algoritmus AdaBoost. Pri vlastnej detekcii nie je spracovaný vstupný obraz ako celok, ale po častiach. Tieto časti sú vybrané pomocou posuvného pod-okna, ktoré mení svoju pozíciu v obraze a súčasne mení aj svoju veľkosť [33].

Snaha o redukciu výpočtovej doby celkovej detekcie je riešená znížením priemernej doby, ktorú detektor venuje pri vyhľadávaní každého pod-okna. To je dosiahnuté kaskádovým zapojením klasifikátorov. Myšlienka tohto riešenia je založená na pozorovaní, že k vytvoreniu klasifikátora, ktorý dokáže vyberať takmer všetky pozitívne prípady a súčasne množstvo (20 - 50%) negatívnych prípadov, stačí iba niekoľko príznakov. Významným postrehom na vytvorenie kaskády je skutočnosť, že väčšina pod-okien v obraze je negatívna, teda neobsahuje sledovaný objekt. Preto sa v každom stupni kaskády snažia odmietnuť čo najviac negatívnych pod-okien, kým do ďalších stupňov kaskády prechádzajú iba pod-okna označená za pozitívne [33].



Obr. 52) Diagram štádií detektoru a použitie [33]

PRÍLOHA C - DETEKTOR SHI TOMASI, KLT TRACKER, BLOB ANALÝZA A BINARIZÁCIA OBRAZU

Detektor Shi Tomasi

Detekor vychádza z Harisovho detektora rohov kde malou modifikáciou páni Shi a Tomas docielili lepšie výsledky. Takto detegované body sa nazývajú ako „*good features to track*“.

Harrisov rohový detektor má kritériá výberu rohov, ktorých skóre sa vypočíta pre každý pixel a ak je skóre nad určitou hodnotou, pixel je označený ako roh. Skóre vypočíta funkcia pomocou dvoch vlastných čísel. Shi a Tomasi navrhli, že by sa táto funkcia mala odstrániť a iba vlastné čísla by sa mali použiť na kontrolu či je pixel bol roh alebo nie [34].

KLT Tracker

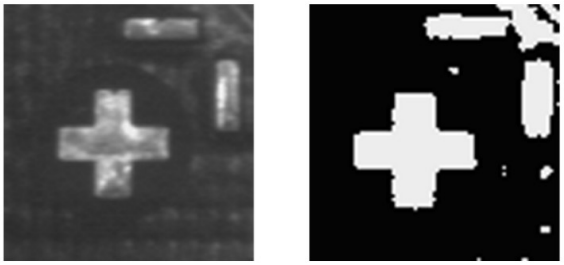
KLT tracker využíva Lucas-Kanade algoritmus, ktorý dokáže určiť pohyb významných bodov v obraze. Pritom používa pohybový vektor kde pre každý významný bod v po sebe nasledujúcich snímkach. Predpokladom je aby sledované body medzi snímkami sa jasovo nemenili. Taktiež sa predpokladá že bod sa pohybuje podobne ako jeho okolie a zároveň nedôjde k výraznej zmene vzdialenosti medzi snímkami.

Takto tracker v slede nasledujúcich dvoch snímok hľadá blízke vzory zložené z významných bodov. Na ich základe nadviaže na nové vypočítané body, ktoré sa porovnajú s predošlými, a zhodné body sa uchovávajú ako vzor pre nasledujúcu snímku. Tracker trackuje body dovtedy kým sa zhoda bodov nezanikne. Potom je nutné inicializovať nové body a zopakovať odznova [31],[35].

Blob analýza a binarizácia obrazu

Blob analýza je základná technika strojového videnia založená na analýze konzistentných oblastí obrazu. Ako taká je nástrojom výberu pre aplikácie, v ktorých sú kontrolované predmety jasne rozpoznateľné z pozadia. Existujú rôznorodé metódy Blob analýzy, ktoré umožňuje vytvoriť na mieru riešenia, pre širokú škálu problémov vizuálnej kontroly. Hlavnými výhodami tejto techniky sú vysoká flexibilita a vynikajúci výkon.

Proces binarizácie predchádza blob analýza. Je to proces kedy na základe nastavenie požadovanej prahovej hodnoty ako štandardu a prevodom „grayscale“ obrazu na hodnoty 0 a 1. Týmto procesom je možné zamerať sa len na záujmový objekt [36]. Výsledok je možné vidieť na obrázku Obr. 53)



Obr. 53) Ukážka binarizácie obrazu pomocou blob analýzy [37]

PRÍLOHA D - KALIBRÁCIA KAMERY

Aby sa kamera mohla použiť na presne meranie, je dôležite ju kalibrovať. To znamená zistiť jej vnútorné a vonkajšie parametre. Šošovka kamery nieje dokonalá a tak je potrebné, vypočítať koeficienty skreslenia kamery na základe, ktorých, sa upraví obraz tak ako má vyzerať ak by šošovka bola dokonalá. Medzi najčastejšie typy skreslení patrí radiálne a tangenciálne skreslenie. Pri kalibrácii kamery sa uvažuje len s radiálnym skreslením šošovky. To sa dá dobre matematicky popísať a tým aj eliminovať z výsledného obrazu. Existuje množstvo kalibračných metód a klasifikujú sa podľa rôznych hľadísk, najviac však podľa informácii o kamere a scéne [28].

Väčšina konvenčných metód kalibrácie kamier sa zakladá na tom, že poznáme referenčné súradnice stabilného bodového poľa. Na to sa používajú kalibračné objekty rôznych tvarov a veľkostí. Ukážka kalibračného objektu je na obrázku Obr. 54). Metódy používajúce na kalibráciu objekty známych rozmerov sa nazývajú fotogrametrické metódy [38].

Obr. 54) Ukážka kalibračného objektu [28]

Kalibráciu kamery možno popísať nasledujúcimi krokmi. Ako prvé potrebujeme nájsť maticu kamery, ktorá popisuje vnútorné parametre kamery. Následne potrebujeme vypočítať vektory otočenia a posunutia kamery od počiatku súradnicového systému. Takto určíme vzájomnú polohu kamery k svetovému súradnicovému systému [28],[38].

V práci je použitý algoritmus kalibrácie od Matlabu, ktorý zodpovedá modelu s dierkovou kamerou. Algoritmus vypočíta maticu kamery *P* pomocou vonkajších a vnútorných parametrov. Vonkajšie parametre predstavujú transformáciu z 3-D svetového súradnicového systému na 3-D súradnicový systém kamery. Vnútorné parametre predstavujú priafektívnu transformáciu z súradníc 3-D kamery do súradníc 2-D obrazu [38].

Transformáciu popisujú rovnice (19)

$$W[x\ y\ 1] = [\mathrm{X\ Y\ Z\ 1}]\ \mathrm{P} \tag{18}$$

kde [X Y Z 1] je vektor bodov vo svete, $[x\ y\ 1]$body premietnuté do obrazu a W škálovací faktor. Matica P, ktorá reprezentuje parametre kamery popisuje rovnica (19):

$$P = \begin{bmatrix} R \\ t \end{bmatrix} \mathrm{K} \tag{19}$$

kde, K- matica vnútorných koeficientov, $\begin{bmatrix} R \\ t \end{bmatrix}$ vonkajšie parametry

Vonkajšie parametre pozostávajú z rotácie a translácie .Tieto parametre sa dopočítajú podľa kalibračného objektu, takže počiatočný súradnicový systém fotoaparátu je v jeho optickom centre a jeho osi x a y definuje rovina obrazu(kalibračný objekt) [38].

Vnútorne parametre zahŕňajú ohniskovú vzdialenosť, optické centrum, tiež známe ako hlavný bod a koeficient skreslenia. Vnútorná matica K, je definovaná ako:

$$K = \begin{pmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{pmatrix}$$

kde,

c_x, c_y-optické centrum kamery

f_x, f_y-ohnisková vzdialenosť v pixeloch vypočítaná z rovníc (20) ako

$$f_x = \frac{F}{px}; f_y = \frac{F}{py} \tag{20}$$

F-ohnisková vzdialenosť v milimetroch

p_x, p_y- veľkosť pixelu v milimetroch

s -skreslenie pixelu je definované podľa rovnice (19) ako

$$s = f_y \tan \alpha \tag{21}$$

kde, α je uhol skreslenia pixelu (podľa kalibračného obrazu)

Matica kamery nezohľadňuje skreslenie šošovky, pretože ideálna dierková kamera nemá objektív. Pre presné reprezentovanie skutočnej kamery, je nutné stanoviť radiálne a tangenciálne skreslenie šošoviek [38].

Radiálne skreslenie sa vyskytuje, keď sa pri prechode šošovky svetelné lúče ohnú viac v blízkosti okrajov objektívu ako v optickom centre. Čím je objektív menší, tým väčšie je skreslenie.

Obr. 55) Ukážka radiálnych skreslení,(negatívne, bez skreslenia, pozitívne) [38]

Koeficienty radiálneho skreslenia modelujú tento typ skreslenia. Deformované body sú definvané rovnicami (22) a (19) ako

$$x_{skreslené} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (22)$$

$$y_{skreslené} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (23)$$

kde, x, y-nenormalizované umiestnenie pixelov. x a y sú v normalizovaných súradniciach obrázka. Normalizované súradnice sú vypočítané zo súradníc pixelov preložením do optického stredu a rozdeľovaním ohniskovej vzdialenosti v pixeloch. Takže x a y sú bezrozmerné,

k_1, k_2 a k_3 – sú koeficienty radiálneho skreslenia šošovky a r^2=x^2+y^2

Tangenciálne skreslenie sa vyskytuje vtedy, keď objektív a rovina obrazu nie sú rovnobežné. Koeficienty tangenciálneho skreslenia modelujú tento typ skreslenia. Čo v prípade kamery použitej v tejto diplomovej práci neplatí.

Obecne pre tangenciálne skreslenie platia rovnice (24)a (19):

$$x_{skreslené} = x(2p_1 xy + p_2(r^2 + 2x^2)) \quad (24)$$

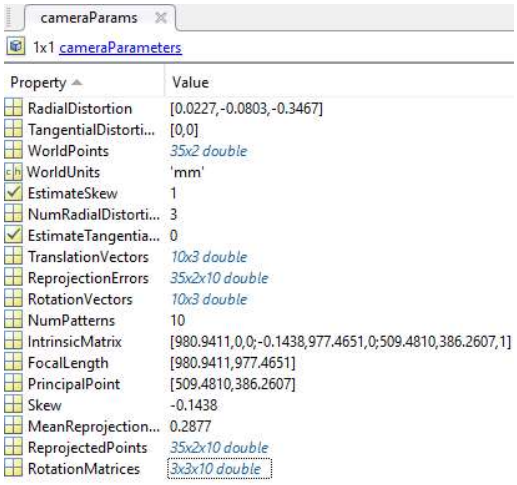
$$y_{skreslené} = y(2p_2 xy + p_1(r^2 + 2y^2)) \quad (25)$$

kde,

x, y-nenormalizované umiestnenie pixelov. x a y sú v normalizovaných súradniciach obrázka. Normalizované súradnice sú vypočítané zo súradníc pixelov preložením do optického stredu a rozdeľovaním ohniskovej vzdialenosti v pixeloch. Takže x a y sú bezrozmerné,

p_1 a p_2– sú koeficienty tangenciálneho skreslenia šošovky, a r^2=x^2+y^2[38].

V programe Matlab kalibráciu fotoaparátu vykonáme pomocou jeho aplikácie *Camera Calibration App.* Z aplikácie sa prevzali skripty kalibrácie a tak bolo možné implementovať algoritmus kalibrácie do grafického užívateľského prostredia pre potreby track-ovanie objektu. Takto je možné vypočítať parametre kamery potrebné pre presné určovanie polohy objektu. Pomocou *cameraParameters* príkazu je možné nahliadnuť do vypočítaných parametrov kamery[38]. Ukážka vypočítaných parametrov je na obrázku Obr. 56)



Obr. 56) Kalibračné parametre kamery

PRÍLOHA E - PID REGULÁCIA SERVOPOHONU

Základnou funkciou regulátora v regulačnom obvode je vytvárať akčnú veličinu u na základe regulačnej odchýlky $e = w - y$. Kde y je požadovaná je Akčná veličina má za úlohu svojím pôsobením na regulovanú sústavu v každom časovom okamihu zabezpečovať to, aby bola regulačná odchýlka u čo najmenšia bez ohľadu na poruchovú veličinu v [39].

Pre spojitý PID regulátor platí rovnica

$$u(t) = kp\left[e(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \tag{26}$$

Keďže počítač nepracuje spojite rovnica sa prevedie na diskrétny tvar tak že integrál nahradíme sumáciou a deriváciu diferenciou:

$$u(t) = r_0[e(k) + \frac{T_V}{2T_i}\sum_{i=0}^{k}(e(i) + e(i-1)) + \frac{T_D}{T_v}(e(k) - e(k-1))] \tag{27}$$

,kde

r_0 -proporcionálny nárast(pásmo),

T_V -perioda vzorkovania,

T_i-integrálny zásah,

T_D-derivačný zásah [39].

PRÍLOHA F - TABUĽKY EXPERIMETU TRACKOVANIA OBJEKTU

X[mm]	Y[mm]	$y = Y_n - Y_{n-1}$	$y - \bar{y}$
28.28	133.46		
28.23	143.48	10.02	-0.0236
28.19	153.49	10.01	-0.0136
28.14	163.51	10.02	-0.0236
28.42	173.52	10.01	-0.0136
28.37	183.85	10.33	-0.3336
28.33	193.86	10.01	-0.0136
28.28	203.86	10.00	-0.0036
28.24	213.86	10.00	-0.0036
28.19	223.85	9.99	0.0064
28.15	233.84	9.99	0.0064
28.42	243.84	10.00	-0.0036
28.05	253.82	9.98	0.0164
28.01	264.12	10.30	-0.3036
27.96	273.78	9.66	0.3364
27.92	283.76	9.98	0.0164
28.19	294.06	10.30	-0.3036
28.15	304.03	9.97	0.0264
28.10	314.00	9.97	0.0264
28.06	323.96	9.96	0.0364
28.01	333.92	9.96	0.0364
27.97	343.88	9.96	0.0364
27.92	353.83	9.95	0.0464
27.88	363.79	9.96	0.0364
27.83	373.73	9.94	0.0564
28.11	383.37	9.64	0.3564
Aritmetický priemeru [mm]		9.995	
Štandardná odchýlka			0.1483

Tab 6) Tabuľka hodnôt súradníc Y pozdĺž horného okraja.

X	**Y**	**X′**	**Y′**	$x = \mathrm{X}' - \mathrm{X}$	$y = \mathrm{Y}' - \mathrm{Y}$	$x - \bar{x}$	$y - \bar{y}$
169.5	113.97	169.50	113.97				
169.49	114.63	169.12	113.99	-0.37	0.64	0.190	-0.327
168.48	116.59	167.98	115.73	-0.50	0.86	0.318	-0.548
167.48	118.24	167.02	117.45	-0.46	0.79	0.276	-0.475
165.81	120.86	165.50	120.33	-0.31	0.53	0.127	-0.218
163.81	123.48	163.98	123.77	0.17	-0.29	-0.352	0.607
163.14	125.12	163.03	124.93	-0.11	0.19	-0.070	0.121
162.13	126.76	162.08	126.67	-0.05	0.09	-0.128	0.221
160.8	128.4	161.12	128.96	0.32	-0.56	-0.506	0.872
160.13	130.36	159.99	130.11	-0.14	0.25	-0.038	0.066
159.13	131.67	159.22	131.83	0.09	-0.16	-0.278	0.479
158.13	133.31	158.27	133.56	0.14	-0.25	-0.326	0.561
157.12	135.27	157.13	135.30	0.01	-0.03	-0.198	0.341
155.79	137.56	155.81	137.59	0.02	-0.03	-0.199	0.342
154.12	140.17	154.29	140.46	0.17	-0.29	-0.353	0.609
152.79	142.79	152.77	142.75	-0.02	0.04	-0.162	0.280
150.79	146.38	150.69	146.20	-0.10	0.18	-0.078	0.135
148.79	149.64	148.79	149.64	0.00	0.00	-0.186	0.320
146.79	153.23	146.71	153.09	-0.08	0.14	-0.102	0.175
144.46	157.14	144.44	157.10	-0.02	0.04	-0.162	0.279
142.47	160.4	142.55	160.53	0.08	-0.13	-0.260	0.447
140.14	164.63	140.09	164.55	-0.05	0.08	-0.134	0.231
137.49	168.86	137.64	169.11	0.15	-0.25	-0.328	0.566
135.5	172.76	135.37	172.54	-0.13	0.22	-0.054	0.093
133.51	176.01	133.48	175.97	-0.03	0.04	-0.158	0.271
130.86	180.55	130.85	180.53	-0.01	0.02	-0.172	0.296
128.55	184.12	128.78	184.51	0.23	-0.39	-0.410	0.705
126.89	187.36	126.90	187.37	0.01	-0.01	-0.189	0.325
123.91	192.22	124.07	192.50	0.16	-0.28	-0.347	0.598
120.94	197.4	121.07	197.62	0.13	-0.22	-0.310	0.534
118.63	201.28	118.81	201.60	0.18	-0.32	-0.368	0.634
116.65	205.16	116.56	205.01	-0.09	0.15	-0.095	0.164
114.67	208.06	114.88	208.42	0.21	-0.36	-0.392	0.675
112.37	212.26	112.44	212.38	0.07	-0.12	-0.254	0.437
110.72	215.49	110.57	215.22	-0.15	0.27	-0.029	0.049
108.42	219.36	108.32	219.19	-0.10	0.17	-0.082	0.141
106.44	222.9	106.26	222.60	-0.18	0.30	-0.007	0.012
104.47	226.12	104.39	225.99	-0.08	0.13	-0.108	0.185
102.83	229.02	102.71	228.82	-0.12	0.20	-0.064	0.110
101.19	231.59	101.22	231.64	0.03	-0.05	-0.212	0.366
99.22	235.13	99.16	235.03	-0.06	0.10	-0.127	0.219
97.25	238.34	97.30	238.43	0.05	-0.09	-0.234	0.403
95.29	241.87	95.25	241.80	-0.04	0.07	-0.144	0.249
92.99	245.72	93.02	245.77	0.03	-0.05	-0.209	0.361
91.36	248.61	91.34	248.57	-0.02	0.04	-0.162	0.279
89.72	251.82	89.48	251.40	-0.24	0.42	0.062	-0.106
88.09	254.38	87.99	254.21	-0.10	0.17	-0.082	0.141
86.45	256.94	86.50	257.03	0.05	-0.09	-0.236	0.407
85.14	259.5	85.02	259.29	-0.12	0.21	-0.060	0.103
83.84	261.74	83.72	261.53	-0.12	0.21	-0.059	0.102
82.54	263.66	82.60	263.77	0.06	-0.11	-0.245	0.422
81.23	266.22	81.12	266.02	-0.11	0.20	-0.069	0.118
79.92	268.45	79.82	268.28	-0.10	0.17	-0.084	0.145
78.29	271.33	78.15	271.09	-0.14	0.24	-0.042	0.073
77.31	273.25	77.03	272.78	-0.28	0.47	0.092	-0.159
76.01	275.48	75.74	275.01	-0.27	0.47	0.087	-0.150
74.39	277.71	74.45	277.81	0.06	-0.10	-0.239	0.411
72.76	280.9	72.59	280.61	-0.17	0.29	-0.017	0.029
71.13	283.77	70.93	283.42	-0.20	0.35	0.019	-0.033
69.18	286.95	69.08	286.78	-0.10	0.17	-0.085	0.146

X[mm]	**Y[mm]**	X′	Y′	$x = \mathrm{X}' - \mathrm{X}$	$y = \mathrm{Y}' - \mathrm{Y}$	$x - \bar{x}$	$y - \bar{y}$
65.28	293.96	65.01	293.50	-0.27	0.46	0.085	-0.146
63.98	296.19	63.72	295.74	-0.26	0.45	0.079	-0.137
62.36	299.05	62.06	298.53	-0.30	0.52	0.120	-0.206
60.09	303.18	59.66	302.44	-0.43	0.74	0.247	-0.426
58.15	306.03	58.01	305.78	-0.14	0.25	-0.038	0.066
55.23	311.43	54.87	310.81	-0.36	0.62	0.177	-0.304
53.3	314.28	53.22	314.13	-0.08	0.15	-0.099	0.170
53.3	314.28	53.22	314.13	-0.08	0.15	-0.099	0.170
52.97	315.23	52.66	314.70	-0.31	0.53	0.123	-0.211
52.65	315.54	52.48	315.25	-0.17	0.29	-0.017	0.030
51.35	318.08	51.01	317.49	-0.34	0.59	0.157	-0.271
49.74	320.93	49.36	320.27	-0.38	0.66	0.202	-0.347
49.41	321.57	48.98	320.84	-0.43	0.73	0.243	-0.419
49.41	321.57	48.98	320.84	-0.43	0.73	0.243	-0.419
47.15	325.04	46.97	324.73	-0.18	0.31	-0.002	0.004
46.51	325.99	46.42	325.83	-0.09	0.16	-0.091	0.157
46.83	325.99	46.42	325.28	-0.41	0.71	0.229	-0.394
46.83	325.36	46.78	325.28	-0.05	0.08	-0.137	0.236
44.89	329.15	44.58	328.62	-0.31	0.53	0.123	-0.213
43.28	332	42.93	331.40	-0.35	0.60	0.168	-0.289
42.31	333.9	41.83	333.07	-0.48	0.83	0.301	-0.518
41.03	335.79	40.73	335.27	-0.30	0.52	0.118	-0.203
39.41	338.63	39.08	338.06	-0.33	0.57	0.147	-0.253
37.8	341.47	37.43	340.84	-0.37	0.63	0.185	-0.319
36.84	343.36	36.33	342.49	-0.51	0.87	0.323	-0.556
35.55	345.25	35.24	344.71	-0.31	0.54	0.130	-0.224
34.59	347.15	34.13	346.36	-0.46	0.79	0.273	-0.470
33.3	349.35	32.86	348.59	-0.44	0.76	0.260	-0.448
32.01	351.87	31.39	350.81	-0.62	1.06	0.433	-0.746
30.73	354.08	30.11	353.01	-0.62	1.07	0.436	-0.751
29.45	355.96	29.02	355.22	-0.43	0.74	0.247	-0.426
27.84	358.79	27.38	357.99	-0.46	0.80	0.280	-0.482
26.24	361.62	25.73	360.75	-0.51	0.87	0.323	-0.556
25.27	363.19	24.82	362.42	-0.45	0.77	0.264	-0.455
24.31	365.08	23.73	364.07	-0.58	1.01	0.402	-0.692
23.35	366.65	22.81	365.73	-0.54	0.92	0.353	-0.608
22.39	367.9	22.09	367.38	-0.30	0.52	0.119	-0.204
22.07	368.85	21.54	367.93	-0.53	0.92	0.350	-0.603
21.43	369.78	21.00	369.03	-0.43	0.75	0.250	-0.431
21.11	370.41	20.63	369.59	-0.48	0.82	0.296	-0.509
20.79	371.04	20.27	370.14	-0.52	0.90	0.341	-0.588
20.15	371.98	19.72	371.24	-0.43	0.74	0.247	-0.426
19.83	372.29	19.54	371.79	-0.29	0.50	0.107	-0.185
Aritmetický priemer[mm]				-0.183	0.315		
Štandardná odchýlka						0.220	0.378

Tab 7) Tabuľka hodnôt X a Y súradníc trajektórie

PRÍLOHA G - DÁTOVÝ NOSIČ

Elektronická verzia práce.

Aplikácia detekcia a track-ovanie farebného objektu

Aplikácia detekcia a rozpoznávanie tváre

3D modely držiakov kamery a servopohonov