

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta elektrotechniky a informatiky

Evidenčné číslo: FEI-5384-72892

Ovládanie UAV gestami ruky

Diplomová práca

Študijný program: aplikovaná informatika

Študijný odbor: 9.2.9. aplikovaná informatika

Školiace pracovisko: Ústav informatiky a matematiky

Vedúci záverečnej práce: Ing. Peter Ťapák, PhD.

Bratislava 2018

Bc. Martin Molnár



ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Martin Molnár**
ID študenta: 72892
Študijný program: aplikovaná informatika
Študijný odbor: 9.2.9. aplikovaná informatika
Vedúci práce: Ing. Peter Ťapák, PhD.
Miesto vypracovania: Ústav informatiky a matematiky FEI STU

Názov práce: **Ovládanie UAV gestami ruky**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je navrhnuť a realizovať na platforme raspberry pi systém rozpoznávanie gest ruky, ktorý prevedie na povel pre ovládanie UAV.

Úlohy:

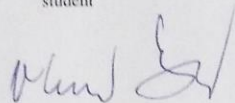
1. Oboznáňte sa s platformou raspberry pi.
2. Zvoľte a nainštalujte si vhodný operačný systém.
3. Využite knižnicu open CV na rozpoznávanie gest ruky.
4. Zvoľte spôsob komunikácie s UAV a realizujte posielanie povelov pre UAV na základe rozpoznaných gest.

Zoznam odbornej literatúry:

1. Shreyamsh Kamate, Nuri Yilmazer, Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles, Procedia Computer Science, Volume 61, 2015, Pages 436-441, ISSN 1877-0509
2. Wang Liang, Liu Guixi, Duan Hongyan, Dynamic and combined gestures recognition based on multi-feature fusion in a complex environment, The Journal of China Universities of Posts and Telecommunications, Volume 22, Issue 2, April 2015, Pages 81-88, ISSN 1005-8885
3. Xiaogang Cheng, Qi Ge, Shipeng Xie, Guijin Tang, Haibo Li, UAV Gesture Interaction Design for Volumetric Surveillance, Procedia Manufacturing, Volume 3, 2015, Pages 6639-6643, ISSN 2351-9789

Riešenie zadania práce od: 18. 09. 2017
Dátum odovzdania práce: 11. 05. 2018

Bc. Martin Molnár
Študent



prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska



prof. Dr. Ing. Miloš Oravec
garant študijného programu

ČESTNÉ VYHLÁSENIE

Podpísaný Bc. Martin Molnár čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne za pomoci môjho školiteľa Ing. Petra Ťapáka, PhD. a s použitím uvedenej dostupnej literatúry.

V Bratislave dňa 10.5.2018

.....

podpis autora

Pod'akovanie

Ďakujem svojmu školiteľovi diplomovej práce Ing. Petrovi Ťapákovi, PhD. za jeho prejavenu ochotu, odborné vedenie a cenné rady, ktoré mi poskytol pri jej vypracovávaní.

ABSTRAKT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program: Aplikovaná informatika

Autor: Martin Molnár

Názov diplomovej práce: Ovládanie UAV gestami ruky

Vedúci diplomovej práce: Ing. Peter Ľapák PhD.

Mesiac a rok odovzdania: Máj, 2018

Hlavným cieľom diplomovej práce bolo vytvoriť jednoduchú aplikáciu, ktorá bude schopná pomocou webovej kamery snímať používateľovu dlaň a rozpoznať gesto, ktoré ukazuje. Na základe rozpoznaného gesta je následne ovládané bezpilotné lietadlo (UAV).

Aplikáciu sme vyvíjali na zariadení Raspberry PI 3 v programovacom jazyku C++ a za pomoci otvorenej multiplatformovej knižnice OpenCv, ktorá poskytuje množstvo nástrojov uľahčujúcich prácu s obrazom.

Práca je rozdelená do jednotlivých logických častí, v ktorých sa postupne zamýšľame nad cieľovými požiadavkami na aplikáciu, vypracovaním návrhu a následne detailne popisujeme ako sme danú aplikáciu implementovali. V závere práce zhrnieme úspešnosť dosiahnutého riešenia na základe reálneho testovania v laboratóriu a zhodnotíme užitočnosť práce.

Kľúčové slová:

Rozpoznávanie, OpenCV, Raspberry, Gestá, UAV

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study programme: Applied informatics

Author: Martin Molnár

Diploma Thesis: Hand gestures UAV control

Supervisor: Ing. Peter Ťapák PhD.

Month and year of submission: May, 2018

The main target of the diploma thesis was to create a simple application that will be able to capture the user's palm by webcam and recognize the gesture they are showing. Based on the identified gesture, the unmanned aerial vehicle (UAV) is then controlled.

We developed the application on the Raspberry PI 3 device in the C++ programming language, mostly using OpenCv open multiplatform library, which provides plenty of tools to process with image.

The thesis is divided into several logical parts in which we are gradually thinking about target application requirements, designing and then describing in detail how we implemented the application. At the end of the thesis we summarize the success of the solution based on real laboratory testing and evaluate the usefulness of thesis.

Key words:

Recognition, OpenCV, Raspberry, Gestures, UAV

Obsah

Úvod.....	1
1. Rozpoznávanie gest.....	3
1.1. Metódy rozpoznávania gest ruky.....	3
1.1.1. Snímanie infračerveného spektra	4
1.1.2. Zariadenie Leap motion	4
1.1.3. Strojové učenie	4
1.1.4. Obrazová detekcia	5
2. UAV	6
2.1. Využitie UAV	7
2.1. Nevýhody UAV	8
3. Návrh riešenia	9
3.1. Cieľové požiadavky.....	9
3.2. Hardvérový návrh	11
3.3. Softvérový návrh	12
3.4. Popis použitých technológií.....	12
3.4.1. Hardvérové technológie	12
3.4.1.2. Raspberry	13
3.4.2. Softvérové technológie.....	18
4. Opis riešenia.....	20
3.1. Príprava hardvéru	20
3.2. Príprava softvéru	22
3.3. štruktúra kódu	22
3.4. Popis kódu.....	24
3.4.1. Nastavenie UART	24
3.4.1. Snímanie videa	26
3.4.2. Aplikácia filtrov	27

3.4.3. Zistenie kontúr.....	30
3.4.4. Nájdenie stredu dlane	32
3.4.5. Nájdenie pozície prstov	34
3.4.6. Komunikácia s Arduino UNO	37
3.4.7. Kompilácia	38
3.4.8. Spustenie	38
5. Testovanie	40
Záver.....	42
Použitá literatúra	44
Prílohy	45
Príloha A: Obsah priloženého CD nosiča	45
Príloha B: Video z testovania aplikácie na reálnom UAV	46
Príloha C: Návod na inštaláciu OpenCv knižnice na Raspbian	47

Zoznam použitých skratiek

UAV - Unmanned Aerial Vehicle

CD - Compact Disc

HD - High Definition

GCC - GNU Compiler Collection

UML - Unified Modeling Language

ARM - Advanced RISC Machines

UART - Universal asynchronous receiver-transmitter

GPIO - General-purpose input/output

OS - Operation System

USB - Universal Serial Bus

RDP – Remote Desktop Protocol

SSH – Security Shell

UNIX - Uniplexed Information and Computing System

LAN – Local Network Area

RAM - Random Access Memory

HDMI – High Definition Multimedia Interface

Zoznam obrázkov, tabuliek a grafov

Obrázok 1 - Príklad obrazovej detekcie gesta	5
Obrázok 2 - Bezpilotné lietadlo (UAV)	7
Obrázok 3 - Towercopter	13
Obrázok 4 - Raspberry PI 3	15
Obrázok 5 - Webkamera A4tech PK-900HA Full HD	16
Obrázok 6 – Arduino UNO	17
Obrázok 7 – RDP klient vo Windows	21
Obrázok 8 – Obrázok použitia kontrastného filtra	27
Obrázok 9 – Obrázok použitia prahového filtra	28
Obrázok 10 – Obrázok použitia funkcie Erode	29
Obrázok 11 – Obrázok použitia funkcie Dilate	30
Obrázok 12 – Obrázok určenia kontúr v obrázku	31
Obrázok 13 – Znázornenie vyhľadávania pozície dlane	33
Obrázok 14 – Konvexné body ruky	34
Obrázok 15 – Náhľad grafického znázornenia aplikácie	39
Obrázok 16 – Náhľad konzolového výpisu aplikácie	39
Obrázok 17 – Testovanie v školskom laboratóriu	40
Graf 1 - UML diagram návrhu aplikácie	10
Graf 2 – Diagram hardvérového návrhu	11
Graf 3 – Sekvenčný UML diagram behu aplikácie	23
Tabuľka 1 – Tabuľka meniacej sa výšky pri testovaní	41

Úvod

Ľudia sa od istého vývojového stupňa začali zaujímať o lietanie. Závideli živočíchom, ktoré to vedeli a zamýšľali sa nad tým, aké by to bolo, a či to nemožno dosiahnuť. Postupne sa tieto ľudské sny začínali aj naplňať s príchodom prvých lietajúcich strojov ako boli vzducholode či prvé lietadlá.

V dnešnej dobe je už pohyb vo výškach pre ľudí úplnou samozrejmou. V poslednom čase ho už ľudia nezvyknú využívať iba na svoju prepravu, ale vytvárajú aj miniatúrne bezpilotné lietadlá, ktoré majú široké spektrum využitia od vytvárania video záznamov až po premiestňovanie predmetov.

Ďalšou vecou, ktorá vstupuje do povedomia ľudí je ovládanie vecí bez dotyku. Už dnes je možné množstvo vecí ovládať nejakým fyzickým pohybom ako napríklad mimikou či gestami. Môže sa stať, že v blízkej budúcnosti veci ako klávesnica a myš nebudeme potrebovať.

V našej práci sa pokúsime tieto dve technologické veci spojiť a vytvoriť jednoduchú aplikáciu, ktorá umožní používateľovi ovládať bezpilotné lietadlo využitím gesta ruky.

V prvých dvoch kapitolách si povieme trochu teórie zo skúmanej oblasti. Najskôr si povieme, akú významnú rolu hrá rozpoznávanie ľudských fyziologických znakov v moderných technológiách a na čo všetko je možné to využiť. Následne sa viac zameriame na rozpoznávanie gest ruky čo je v podstate hlavnou myšlienkou našej práce.

Druhá kapitola približuje svet moderných bezpilotných lietadiel. Popisujeme v nej čo je to bezpilotné lietadlo, načo sa v dnešnej dobe zvyknú používať a akým spôsobom ho budeme používať v našej práci.

Tretia kapitola sa hlbšie venuje cieľu našej práce. Zamýšľame sa v nej nad tým, ako by mal cieľ našej práce vyzeráť, ako ho dosiahnuť a následne si aj navrhujeme konkrétny postup, ktorým by sme sa k cieľu chceli dopracovať. Ďalej si v tejto kapitole aj bližšie popíšeme hardvérové a softvérové technológie, ktoré sme si zvolili na využitie pri vývoji našej cieľovej aplikácie.

Vo štvrtej kapitole si už rozoberieme konkrétne implementačné riešenie našej aplikácie. Najskôr si povieme niečo o tom, čo je potrebné spraviť pred samotným začiatkom implementácie a ako si pripraviť pohodlne vývojové prostredie na zariadení Raspberry PI 3. Následne si prejdeme najdôležitejšie časti zdrojového kódu, kde popíšeme jednotlivé algoritmy na detekciu gesta ruky a komunikáciu s UAV. V závere kapitoly môžeme vidieť ako výsledná aplikácia vyzerá v grafickom náhľade.

Kapitola päť sumarizuje výsledok testovania aplikácie, ktoré prebiehalo v školskom laboratóriu na reálnom, zjednodušenom UAV.

V závere diplomovej práce sumarizujeme celkový pohľad na našu prácu či už je to jej praktická alebo teoretická časť a zhŕňame všetky zistené poznatky. Tak isto v nej nájdeme zoznam oblastí, kde by informácie z našej práce mohli byť nápomocné a užitočné. Pre zaujímavosť prikladáme aj video z reálneho testovania, kde môžeme vidieť ako celá aplikácia funguje.

1. Rozpoznávanie gest

Rozpoznávanie gest je významnou témou v oblasti informatiky a v dnešnej dobe predstavuje nielen pomerne širokú oblasť výskumu ale aj samotného využitia. Cieľom tejto oblasti je interpretovať ľudské gestá prostredníctvom matematických algoritmov do počítačovo spracovateľnej podoby. Gestá môžu pochádzať v podstate z akéhokoľvek telesného pohybu alebo stavu, ale najčastejšie pochádzajú z tváre alebo ruky.

Súčasný zameranie v oblasti sa špecializuje najmä na rozpoznávanie emócií z výrazu tváre a rozpoznávanie gesta dlane. Používatelia môžu následne používať jednoduché gestá na komunikáciu so zariadeniami, alebo na ich ovládanie bez priameho fyzického kontaktu. Rozpoznávanie gest môže byť vnímané ako spôsob, aby počítač začal chápať ľudský jazyk tela, čím vytvára bohatšie prepojenie medzi ľuďmi a strojmi. Príkladov využitia rozpoznávania gest je mnoho. Môžu to byť napríklad:

- Interpretácia znakovej reči,
- Práca so smartfónmi
- Domáca automatizácia zariadení
- Rozlišovanie ľudských emócií
- Práca vo virtuálnej realite
- Ovládanie niektorých prístrojov a iné

V našej práci sa venujeme konkrétne rozpoznávaniu gesta ruky a následným ovládaním malého bezpilotného lietadla - UAV.

1.1. Metódy rozpoznávania gest ruky

Rozpoznávať jednoduché ľudské fyziologické črty ako napríklad výraz tváre alebo gesto ruky je možné v informatike robiť viacerými spôsobmi. Keďže sa v našej práci budeme venovať konkrétne rozpoznávaniu gesta ruky z pohľadu obrazového rozpoznávania, popíšeme si aj niekoľko iných metód, ktorými sa tento cieľ dá dosiahnuť a porovnáme v čom sú ich výhody a nevýhody.

1.1.1. Snímanie infračerveného spektra

Pri tomto spôsobe rozpoznávania je využívané infračervené žiarenie vyžarované ľudským telom. Pri takomto spôsobe je potrebné mať k dispozícii kameru schopnú snímať infračervené žiarenie. Tento prístup sa využíva pomerne často vďaka vysokej efektívnosti a zároveň nebýva príliš rušený ani nadmernou alebo nedostatočnou svetlosťou v miestnosti.

1.1.2. Zariadenie Leap motion

„Ide o malé periférne USB zariadenie od Americkej spoločnosti Leap Motion, Inc., ktoré je navrhnuté tak, aby bolo umiestnené na fyzickej ploche smerujúcej nahor. Môže byť tiež namontovaný na slúchadlá s virtuálnou realitou. softvérom Leap Motion pomocou komplexných matematických vzorcov vytvára 3D pozície rúk porovnaním 2D rámcov vytvorených dvoma kamerami. V štúdii z roku 2013 bola celková priemerná presnosť zariadenia 0,7 milimetra.“ (1) Nevýhodou zariadenia je zatiaľ nedostatok kvalitného softvéru pre jeho použitie.

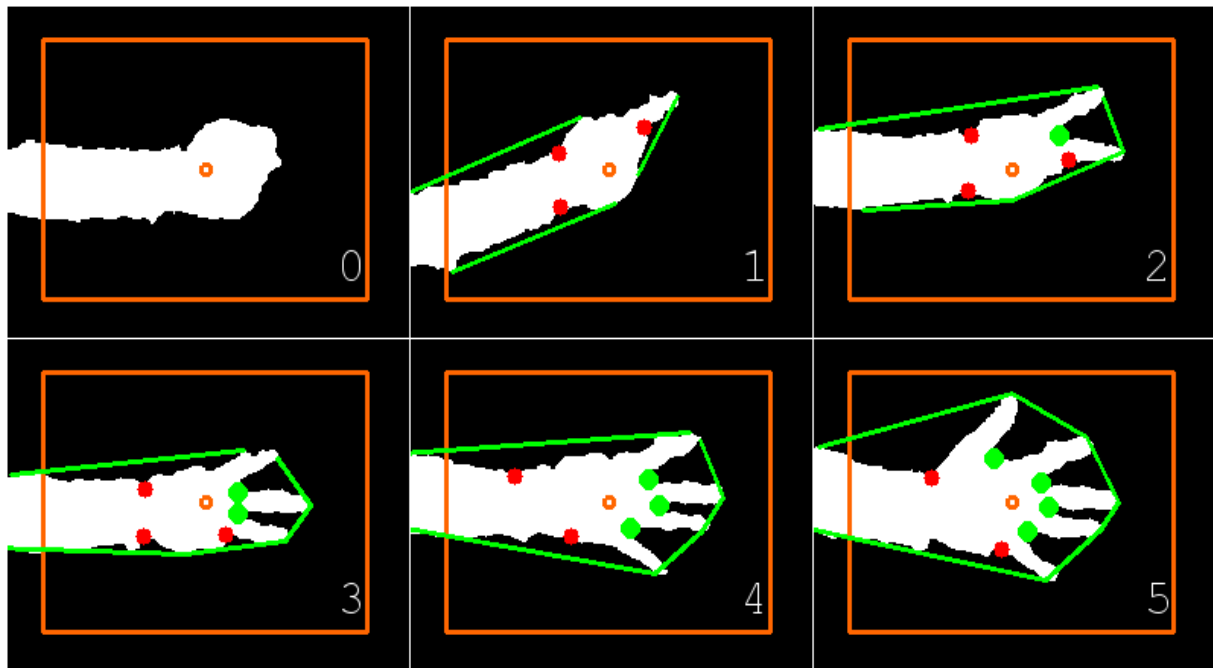
1.1.3. Strojové učenie

Strojové učenie (angl. *Machine learning*) je podoblasť umelej inteligencie, ktorá sa zaoberá metódami a algoritmami, umožňujúcimi programu učiť sa a následne adekvátne reagovať na rôzne vstupné hodnoty bez toho, aby bol na ne explicitne naprogramovaný, iba na základe informácií, ktoré sa naučil tzv. tréningových dát. Tieto dáta sú väčšinou úmyselne vytvorené práve za účelom čo najlepšieho výsledku natrénovania.

Pri rozpoznávaní gest pomocou takéhoto strojového učenia je využívané snímanie dlane kamerou. Na rozpoznávanie je použitá tzv. neurónová sieť, ktorú je potrebné natrénovať tréningovými dátami. Tie v tomto prípade znamenajú početné obrázky rôznych gest, pričom každé gesto je obsiahnuté viackrát na rôznych obrázkoch. To zabezpečí aby bola sieť natrénovaná čo najoptimálnejšie. Následne je snímané gesto v neurónovej sieti vyhodnocované a je mu pridelené najpravdepodobnejšie gesto. Presnosť tohoto riešenia závisí najmä od kvality a počtosti tréningových dát, kamery a vhodne nastavenej neurónovej siete.

1.1.4. Obrazová detekcia

Táto detekcia je založená na obyčajnom snímaní obrazu pomocou farebnej kamery. K rozpoznávaniu gesta je následne využité rôzne upravovanie snímok, hľadanie kontúr a matematické počítanie vzájomnej pozície prstov a dlane. Podrobnejšie sa touto detekciou budeme zaoberať v ďalších častiach tejto práce, pretože sme si pre detekciu gest vybrali práve tento spôsob.



Obrázok 1 - Príklad obrazovej detekcie gesta

2. UAV

Pod skratkou UAV (Unmanned Aerial Vehicle) môžeme rozumieť bezpilotné lietadlo. Ide o menšie alebo väčšie motorové, rotorové lietadlo bez pilota s otáčavými nosnými plochami schopné vzlietnuť, vznášať sa na mieste, pristáť či pohybovať sa na rôzne smery. Zväčša je skonštruované z ľahkých materiálov, ktoré sú odolné voči nárazom alebo poveternostným podmienkam. UAV je zovšeobecnením názvu pre viaceré typy bezpilotných lietadiel ako napríklad:

- **Autonómne lietadlo** - predprogramované bezpilotné lietadlo neriadené pilotom, ktoré sa pohybuje vopred určeným spôsobom alebo trasou, prípadne je vybavené samostatnými autonómnymi subsystémami pomocou, ktorých si dokáže určovať trasu.
- **Diaľkovo riadené lietadlo** - bezpilotné lietadlo riadené z riadiacej stanice, ktorá nie je na palube lietadla.
- **Model lietadla** - bezpilotné lietadlo používané takmer výhradne len na športové, súťažné alebo rekreačné účely. Zväčša je ovládané diaľkovým ovládačom, ktorý má obmedzený dosah.

Prvé bezpilotné lietadlo zrejme vzniklo v roku 1916 a vytvoril ho profesor Archibald Montgomery Lowe a bolo pomenované ako Aerial Target. Odvtedy sa situácia zmenila a s rozmachom miniaturizácie technológií koncom 90tych rokov nastal výrazný nárast počtu najrôznejších bezpilotných lietadiel, ktoré si už dnes môže dovoliť takmer každý. Ich cena je vcelku prijateľná.

Bezpilotné lietadlá bývajú často vybavené senzormi na zber informácií ako napríklad kamera, vlhkomer, teplomer alebo v prípade vojenských lietadiel zbraňovými a prieskumnými systémami. UAV sú niekedy nazývané aj ako DRON. Názov DRON sa však väčšinou oveľa viac zvykne spájať s malými bezpilotnými lietadlami ovládanými vzdialene, ktoré je možné využiť na mnohé činnosti, niekedy aj nelegálne, zatiaľ čo pod UAV bezpilotným lietadlom si predstavujeme skôr využitie vo vojenskom boji či prieskume. V podstate však možno povedať že oba pojmy sa významovo zhodujú. (2)



Obrázok 2 - Bezpilotné lietadlo (UAV)

2.1. Využitie UAV

O UAV možno povedať, že v dnešnej dobe sú čoraz populárnejšie a majú množstvo oblastí kde môžu byť nápomocné. Malé lietajúce stroje dokážu manévrovať, a keď spadnú, nič strašné väčšinou nestane ani lietadlu ani prípadným osobám. Veľmi často je UAV vybavené snímacou kamerou, ktorá vie zaznamenať veľmi hodnotné zábery z vtácej perspektívy. K najbežnejším využitiam bezpilotných lietadiel patria:

- Prieskum ťažko dostupných miest
- Presun predmetov menších rozmerov
- Monitorovanie zvierat v divočine
- Kamerový záznam pri natáčaní filmov
- Sledovanie osôb
- Monitorovanie počasia
- Reklamná činnosť
- Kuriérske doručovanie

2.1. Nevýhody UAV

Drony si však nájdu aj svojich odporcov, ktorí sa netaja tým, že by ich najradšej zakázali, alebo aspoň prísne obmedzili. Medzi najčastejšie dôvody patrí to, že je veľmi ľahké ich zneužiť na narušenie súkromia. To zvyknú využívať napríklad novinári na odpočúvanie, alebo aj vojaci na preskúmavanie územia iného štátu.

Druhou nebezpečnou vecou je, že drony, napriek svojej malej veľkosti a nízkej váhe, môžu spôsobiť problémy najmä v mestách, kde ich poveternostné podmienky môžu hodiť do drôtov elektrického vedenia alebo poškodiť budovy. Rovnako môžu pri náhlej poruche aj zraniť ľudí v prípade, že na niekoho spadnú z väčšej výšky.

Aj z týchto dôvodov sa čoraz viac prijímajú rôzne obmedzenia na ich používanie a začínajú sa viac chápať ako bežné lietadlá, ktoré by nemali len tak slobodne lietať po vonku.

3. Návrh riešenia

V tejto kapitole sa zamyslíme konkrétnejšie nad našim cieľom a viac sa zameriame na naše cieľové požiadavky na aplikáciu, ktorá by mala byť schopná rozpoznať viacero používateľových gest a popritom komunikovať s UAV. Následne navrhujeme softvérové a hardvérové technológie, ktoré by sme chceli pri vývoji a testovaní aplikácie využívať a zároveň sa s nimi v poslednej podkapitole aj bližšie oboznámime.

3.1. Cieľové požiadavky

Tu si skúsime presnejšie zadefinovať čo od našej aplikácie očakávame. Prvým krokom určite bude zabezpečiť, aby sme boli schopní snímať obraz z kamery a následne ho spracovať takým spôsobom, že budeme vedieť rozpoznať používateľovo gesto. Pre jednoduchosť si definujeme päť základným gest, pričom každé bude reprezentované počtom prstov. Nebudeme teda pre jednoduchosť skúmať, uhol medzi nimi i keď v konečnom dôsledku by to s použitím jednoduchej matematiky nebol žiadny problém.

Po prečítaní rôznej náučnej literatúry sme zistili, že na rozpoznanie objektu na obraze je vhodné obraz nejakým spôsobom upraviť, aby bolo možné v ňom následne vyhľadávať útvary, ktoré chceme. Ide najmä o úpravy svetelných podmienok, farebného spektra a vyhľadávanie obrysov (kontúr). Potom už vieme určiť body kontúr a na základe 2D rozloženia týchto bodov určiť, ktorý bod reprezentuje akú časť dlane. Na základe toho, ktoré body sú pre nás dôležité, s nimi už vieme pracovať podľa ľubovoľného uváženia a v našom prípade určovať gestá.

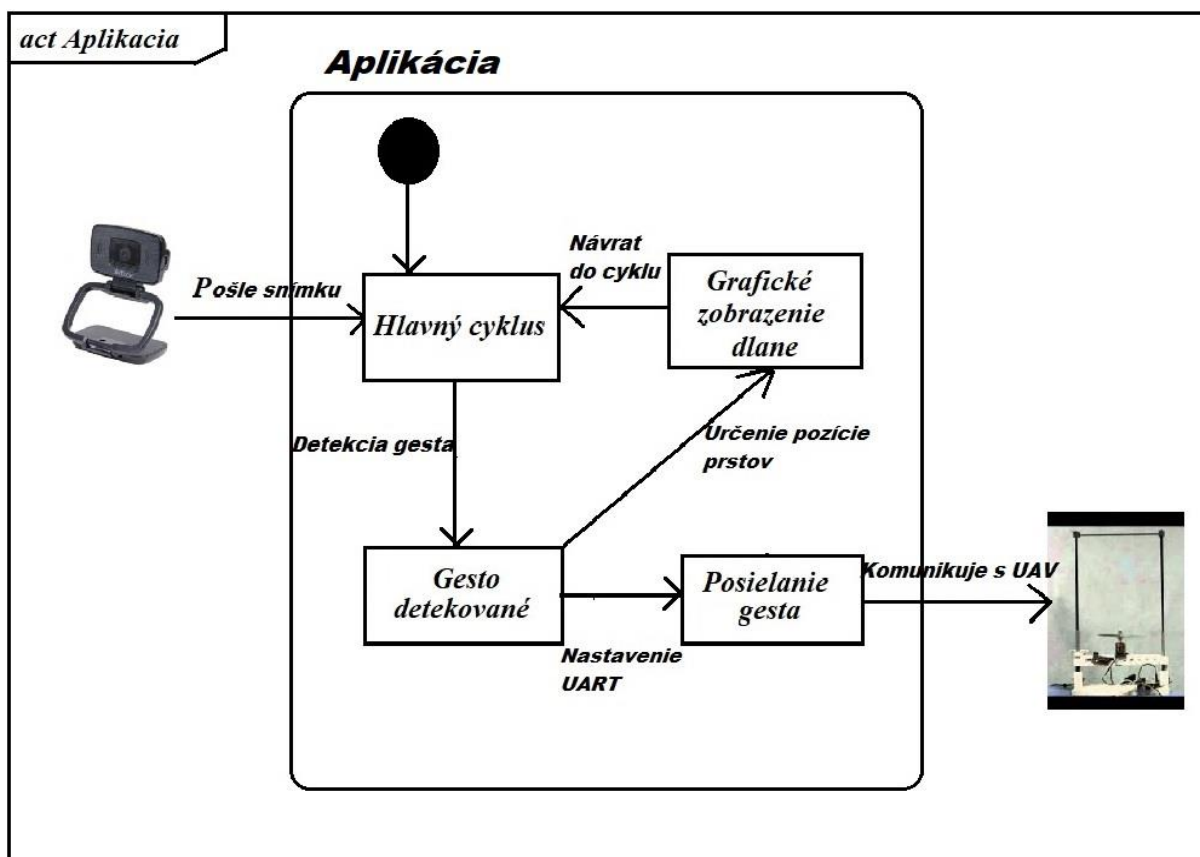
Od aplikácie očakávame že nám bude dávať dva druhy grafických výstupov.

- Prvý výstup bude výstup z kamery zobrazený v náhľade, podľa ktorého budeme vedieť presne umiestniť dlaň do snímачa. Zároveň by sme chceli tento výstup využiť aj na grafické znázornenie našich zistených údajov ako sú dôležité obrysové body a najmä pozície prstov a dlane.
- Druhým výstupom bude textový výstup, ktorý nás bude informovať o aktuálnom priebehu. Priebežne nám bude vypisovať aké gesto je momentálne na kamere. Keďže UAV zvyknú zvyčajne vracať rôzne zamerané dáta, budeme aj tieto získané dáta

vypisovať. V poslednom rade tu budú vypísané aj prípadné chybové hlásenia. Tento výstup budeme riešiť výstupom v štandardnej linuxovej konzole.

Samozrejme bude potrebné aj zabezpečiť komunikáciu s UAV. Každé UAV je iné, z čoho vyplýva, že sa budeme musieť zamerať na naše konkrétne testovacie UAV, ktorého súčasťou je miniatúrna doska Arduino UNO, o ktorej si povieme viac nižšie. Naša komunikácia teda bude prebiehať s najväčšou pravdepodobnosťou cez sériovú USB linku.

Aplikácia by mala bežať v cykle, pričom každá iterácia cyklu bude mať na starosti zosnímať snímku z kamery, pokúsiť sa určiť gesto, odoslať toto gesto UAV a zobrazit náhľad snímky aj s detekovanými údajmi. Plánovaný priebeh činnosti aplikácie si môžeme pozrieť na nasledovnom UML diagrame:



Graf 1 - UML diagram návrhu aplikácie

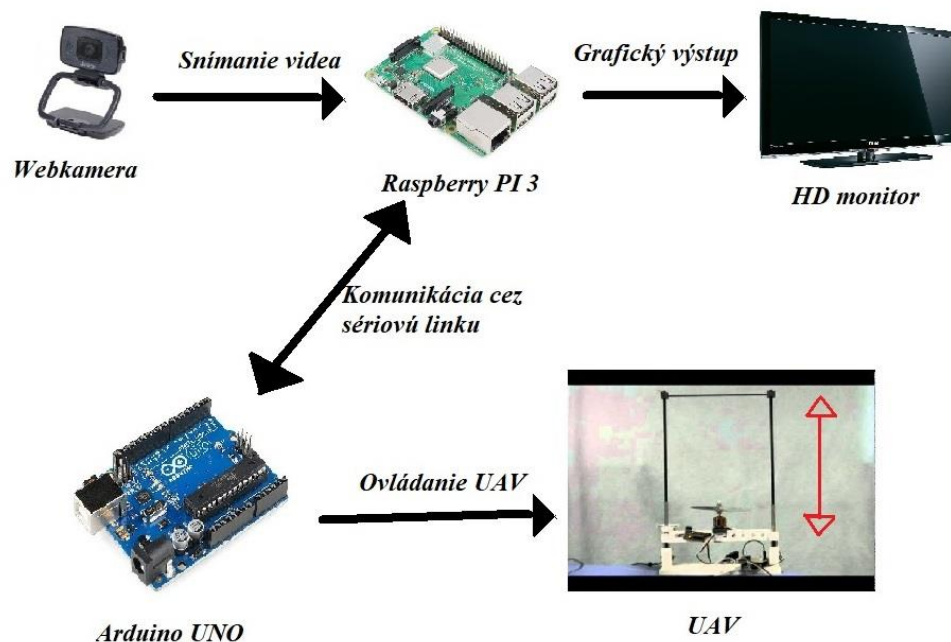
3.2. Hardvérový návrh

Z dôvodu lepšieho oboznámenia sa so systémom Raspbian a aj kvôli zvýšeniu originality práce sme sa rozhodli, že vývoj a aj testovanie aplikácie budú prebiehať na miniatúrnom zariadení veľkosti platobnej karty - Raspberry PI 3, na ktorom bude nainštalovaný operačný systém Raspbian Stretch.

Keďže chceme snímať používateľovu dľaň, a z nej určovať gesto, je jasné že bude nevyhnutné k zariadeniu Raspberry PI pripojiť nejaké zariadenie, ktoré toto snímanie umožní. Ako vhodné zariadenie nám posluží farebná USB webkamera **A4tech PK-900HA Full HD**, ktorá má vyhovujúcu kvalitu a pripojíme ju do jedného zo štyroch USB portov Raspberry PI.

Aby sme mali k dispozícii aj grafický náhľad aplikácie je vhodné zabezpečiť, aby sme mali možnosť vidieť grafický výstup zo zariadenia. Preto ďalším hardvérovým komponentom by mal byť HD displej s HDMI prepojovacím káblom.

Po rozpoznaní gesta nám ešte zostáva zabezpečiť, aby Raspberry PI 3 vedelo toto gesto poslať UAV. Nato nám dobre posluží miniatúrne zariadenie Arduino UNO, ktoré na jednej strane bude cez sériovú USB linku komunikovať s Raspberry PI 3 a zároveň bude pomocou GPIO vstupných a výstupných pinov dávať príkazy UAV. Na zariadení Arduino Uno bude exportovaný kód, ktorý bude čítať informáciu od Raspberry a konvertovať ju na konkrétne povely pre UAV.



Graf 2 – Diagram hardvérového návrhu

3.3. Softvérový návrh

Keďže už máme spracovaný hardvérový návrh diplomovej práce, môžeme sa konkrétnejšie pozrieť aj na návrh softvérových technológií. Ako sme vyššie spomínali vývoj aplikácie bude prebiehať na zariadení Raspberry PI 3 pod operačným systémom Raspbian - operačný systém založený na Linuxovej distribúcii Debian určenej pre ARM procesory.

V prvom rade by bolo veľmi efektívne využiť na spracovanie obrazu nejakú externú knižnicu. Použijeme teda zrejme najznámejšiu a najrozšírenejšiu knižnicu pre počítačové videnie OpenCv. Nepatrí však do základnej výbavy systému Raspbian, a preto ju budeme musieť manuálne stiahnuť, skompilovať a nainštalovať. Bližší návod je v prílohe C.

Ďalším softvérom, ktorý budeme potrebovať je programovací jazyk, v ktorom chceme cieľovú aplikáciu vyvíjať. Ak chceme pracovať s knižnicou OpenCv, výber jazykov sa zúžil na Java, C, C++ a Python. Po zvážení našich súčasných vedomostí a skúseností sme si po dlhšej úvahe zvolili jazyk C++, s ktorým mám bohaté skúsenosti. Naša aplikácia nie je príliš rozsiahla takže nám úplne postačí písať kód v textovom editore a kompilovať ho z príkazového riadku pomocou prekladača gcc (resp. g++).

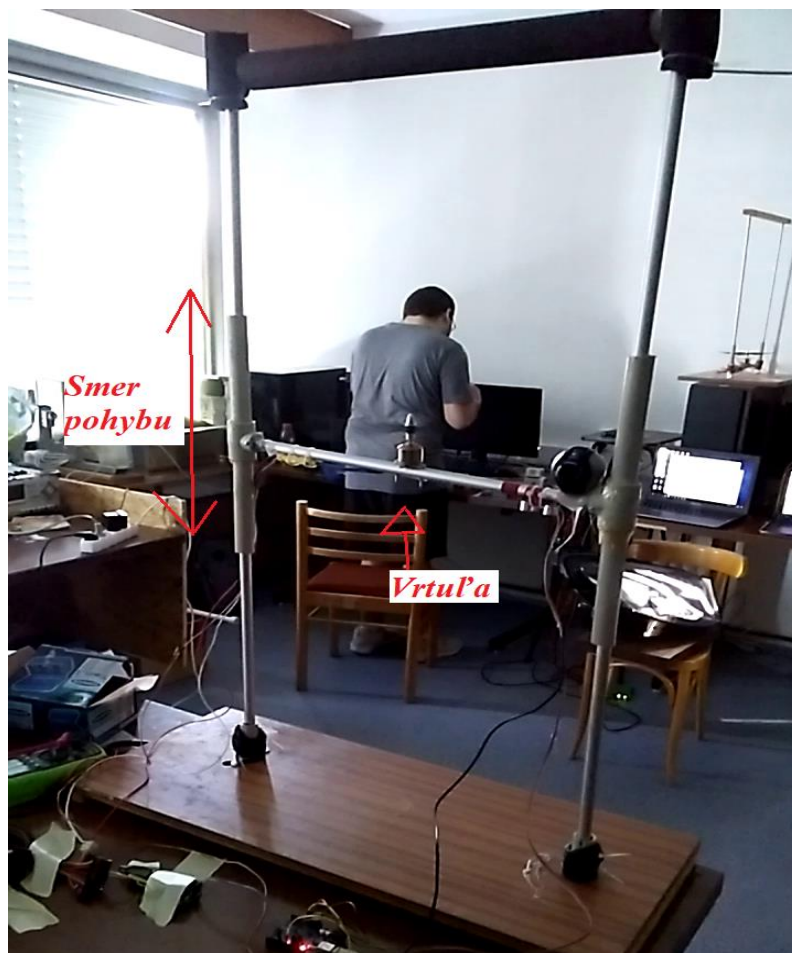
3.4. Popis použitých technológií

Keďže už máme vypracovaný aspoň približný návrh nášho riešenia, môžeme sa v tejto kapitole detailnejšie oboznámiť s hardvérovými a softvérovými technológiami, ktoré sme si zvolili, a ktoré budeme používať pri vývoji a testovaní nášho programu.

3.4.1. Hardvérové technológie

3.4.1.1. UAV – Towercopter

V našej práci sme najmä kvôli bezpečnosti používali zjednodušenú formu bezpilotného lietadla tzv. tower-copter. Ide o UAV pohybujúce sa na dvoch zvislých tyčiach, ktoré umožňujú pohyb hore a dole a zároveň zabráňujú nechcenému pohybu mimo zadaného priestoru či poškodeniu UAV. Tower-copter má teda obmedzenú dráhu pohybu.



Obrázok 3 - Towercopter

3.4.1.2. Raspberry Pi 3

Raspberry Pi je lacný jednodoskový počítač s veľkosťou kreditnej karty, ktorý sa pripája do počítačového monitora alebo televízora, a používa štandardnú klávesnicu a myš. Je to schopné malé zariadenie, ktoré umožňuje ľuďom všetkých vekových kategórií skúmať výpočty a naučiť sa programovať v jazykoch, ako sú Scratch a Python. Je schopný robiť všetko, čo od stolového počítača očakávate. Od prehliadania internetu a prehrávania videa vo vysokom rozlíšení, vytvárania tabuliek, spracovania textu a hrania hier.

Okrem toho má Raspberry Pi schopnosť komunikovať s okolitým svetom, a používa sa v širokej škále projektov digitálnych výrobcov, od hudobných strojov a rodičovských detektorov až po meteorologické stanice a tweeting birdhouses s infračervenými kamerami. Chceme vidieť, že Raspberry Pi používajú aj deti po celom svete, aby sa naučili programovať a chápať, ako fungujú počítače. Aktuálne existuje celý rad zariadení Raspberry PI.

- **Raspberry Pi A+, B+, Zero a Compute module** so základnou doskou SoC BCM2835 firmy Broadcom, ktorý obsahuje centrálny procesor ARM1176JZF-S s taktom 700 MHz a 256 MiB alebo 512 MiB pamäte RWM (RAM).
- **Raspberry Pi 2** so základnou doskou SoC BCM2836 tiež firmy Broadcom, ktorý obsahuje štyri spätne kompatibilné CPU s taktom 900 MHz, posilnenú jednotkou SIMD a 1 GiB pamäte RAM.
- **Raspberry Pi 3** je zatiaľ najaktuálnejší model so základnou doskou Compute Module 3 je SoC BCM2837 firmy Broadcom s taktom 1,2 GHz.

Všetky typy Raspberry Pi obsahujú grafický procesor VideoCore IV kompatibilný s OpenGL ES 2.0. Naopak neobsahujú žiadne rozhranie pre pevný disk alebo SSD – pre zavedenie systému a trvalé uchovanie dát je určený slot na SD kartu.

Samotný výrobca ponúka k počítaču ako operačné systémy ARM verzie linuxových distribúcií Debian (Raspbian) a Arch. Výrobca tiež ohlásil práce na systéme Rasdroid pre Raspberry Pi na báze Androidu 4.0.

Raspberry PI Foundation

„Je nadácia zaoberajúca sa vývojom zariadení Raspberry Pi. Je to registrovaná vzdelávacia charita (registračné číslo 1129409) so sídlom v Spojenom kráľovstve. Cieľom nadácie je podporiť vzdelávanie dospelých a detí, najmä v oblasti počítačov, informatiky a súvisiacich predmetov.“ (1)

ARM Procesor

„ARM je označenie architektúry procesorov v informatike používaných najmä vďaka svojej nízkej spotrebe elektrickej energie v mobilných zariadeniach (mobilné telefóny, tablety). Globálne bola v roku 2013 ARM najpočetnejšie zastúpenou architektúrou mikroprocesorov, pričom 60% mobilných zariadení na svete obsahuje ARM čip. V roku 2013 bolo vyrobených 10 miliárd ARM procesorov, v roku 2014 už 50 miliárd. Vývoj ARM architektúry sa začal v Británii vo firme ARM Holdings v 80. rokoch 20. storočia.“ (2)

Hardvérová špecifikácia Raspberry Pi 3 (rašp3)

SoC: Broadcom BCM2837

Processor: 4× ARM Cortex-A53, 1.2GHz

Grafická karta: Broadcom VideoCore IV

Operačná pamäť RAM: 1GB LPDDR2 (900 MHz)

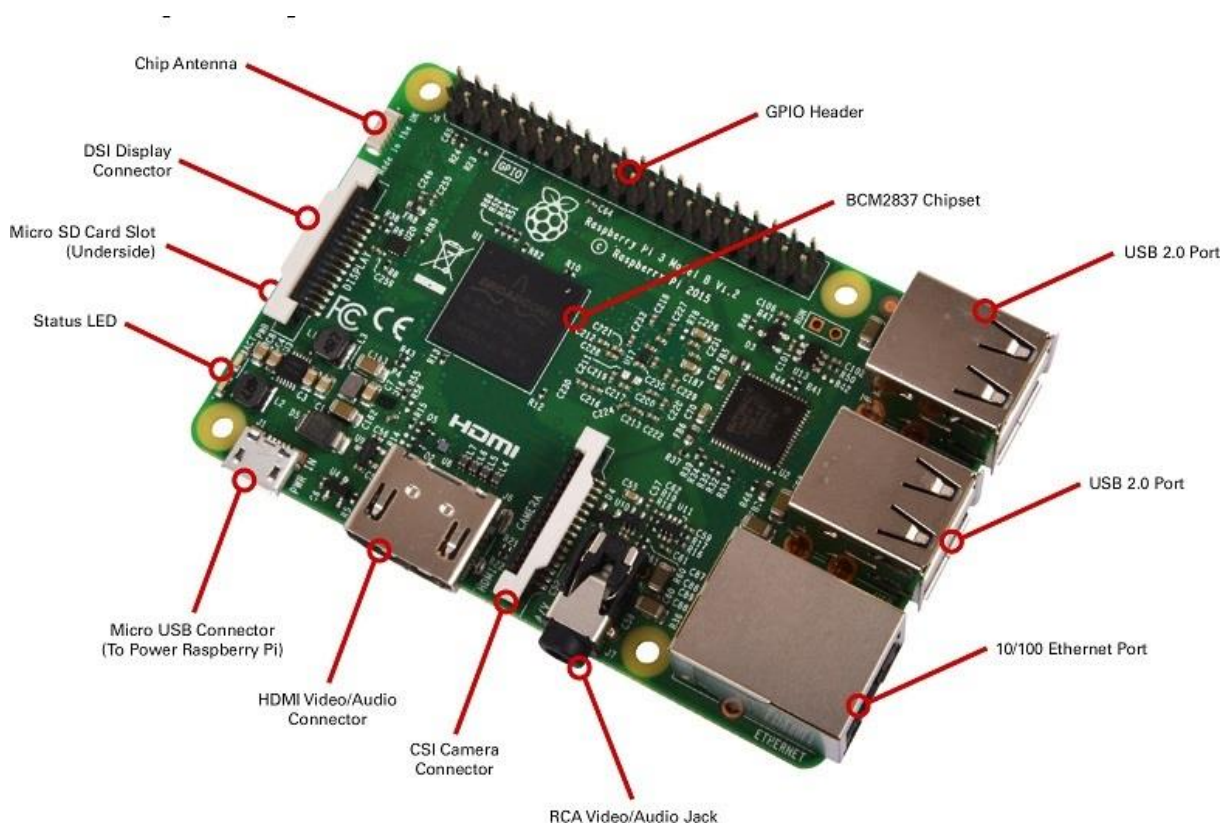
Sieť: 10/100 Ethernet, 2.4GHz 802.11n + WiFi

Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Úložisko: microSD karta

GPIO: 40-pin header, populated

Porty: HDMI, 3.5mm analógový audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



Obrázok 4 – zariadenie Raspberry PI 3

3.4.1.3. Webkamera

„Webkamera alebo webová kamera je vstupné zariadenia počítača slúžiace na snímanie pohyblivého obrazu alebo statických obrázkov, a ich prenos do počítača pre ďalšie spracovanie. Predstavujú konštrukčne jednoduchý spôsob digitalizácie obrazu pre počítačové spracovanie.“ (3)

Hlavným využitím webkamier je internet. Využíva sa najmä na zobrazenie účastníkov videohovoru v reálnom čase, ale my ju budeme používať na snímanie používateľovej dlane, z ktorej budeme určovať gesto.

A4tech PK-900HA Full HD

Konkrétne sme si zvolili webkameru **A4tech PK-900HA Full HD**. Ide o kompaktnú webkameru využívajúcu vysoko citlivý Full HD senzor zaisťujúci dokonale ostré snímanie videa aj pri nižšej hladine okolitého osvetlenia v rýchlosti 30 snímok za sekundu a rozlíšení až 16 Mpx.

Špecifikácia A4tech PK-900HA Full HD

Rozlíšenie videa: 1920 × 1080 Full HD Mpx

Rozlíšenie fotografií až 16 Mpx

Podporovaný operačný systém: Microsoft Windows 7, Microsoft Windows Vista, Microsoft Windows XP

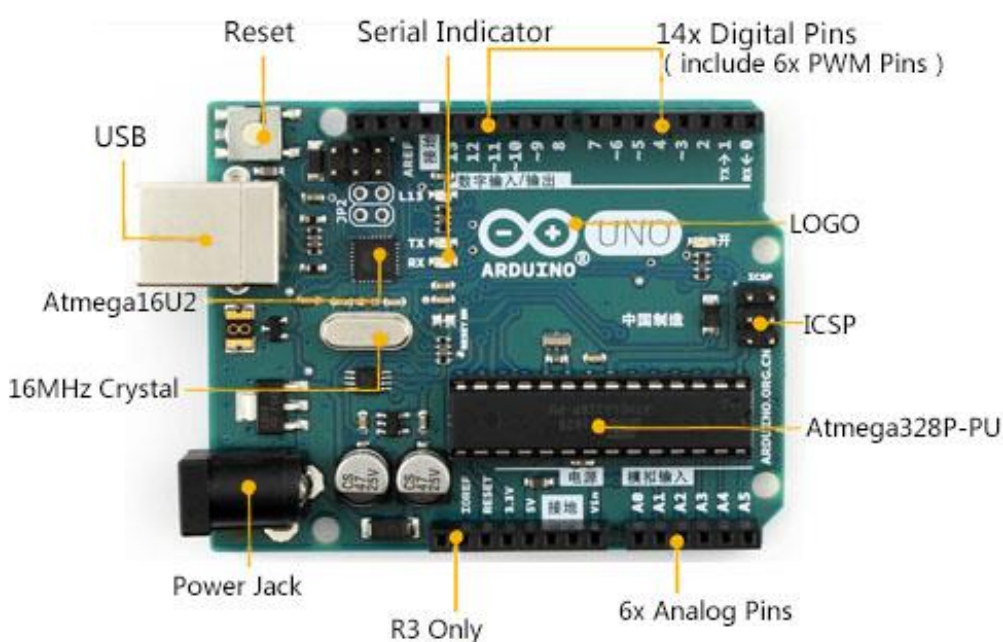


Obrázok 5 - Webkamera A4tech PK-900HA Full HD

3.4.1.4. Arduino UNO

„Arduino UNO je open-source platforma založená na mikrokontroléri ATmega od firmy Atmel a grafickom vývojovom prostredí, ktoré vychádza z prostredia Wiring (podobný projekt ako Arduino, teda doska s mikrokontrolérom a IDE a Processing (prostredie pre výuku programovania). Arduino môže byť použité k vytváraniu samostatných interaktívnych zapojení, alebo môže byť pripojené k softvéru na počítači (napr. Macromedia Flash, Processing, Max/MSP, Pure Data, SuperCollider). Momentálne možno kúpiť verzie, ktoré sú už skompletované; schéma a návrh plošného spoja je dostupný pre tých, ktorí si chcú postaviť Arduino sami. Projekt Arduino získal ocenenie v kategórii digitálnych komúnít na Prix Ars Electronica 2006.“ (4)

Arduino UNO sa od Raspberry PI líši najmä svojou jednoduchšou architektúrou. V podstate na ňom môže bežať súčasne iba jeden proces na rozdiel od Raspberry, na ktorom môže bežať kompletný operačný systém s viacerými procesmi naraz.



Obrázok 6 – Arduino UNO

3.4.2. Softvérové technológie

3.4.2.1. *Raspbian*

Raspbian je bezplatný operačný systém primárne určený najmä pre zariadenia série Raspberry Pi. Je založený na inom Unixovom systéme - Debiane optimalizovanom pre hardware Raspberry a procesory ARM.

Počiatkové vydanie bola uskutočnené v júni 2012 a v roku 2015 ho oficiálne vydala aj Nadácia Raspberry Pi ako hlavný operačný systém pre rodinu počítačov Raspberry Pi. Raspbian však nie je spojený s Nadáciou Raspberry Pi. Vytvorili ho Peter Green a Mike Thompson ako nezávislý projekt.

Existuje viacero verzií Raspbian, ako napríklad Raspbian Jessie či Raspbian Stretch. My využívame v našej práci Raspbian **Stretch**. Operačný systém je stále vysoko frekventovaný i napriek tomu, že operačný systém pre Raspberry vidala aj firma Microsoft pod názvom Windows 10 IoT core. Raspbian aj Windows 10 IoT core sú vysoko optimalizované pre nízkonapäťové ARM procesory radu Raspberry Pi, a sú podporované procesormi so štruktúrou ARM, 86x a 64x.

3.4.2.2. *C++*

C ++ je univerzálny, objektovo orientovaný programovací jazyk (OOP). Je jedným z najpopulárnejších jazykov, ktorý sa primárne používa so systémovým alebo aplikačným softvérom, ovládačmi, aplikáciami klient-server a zabudovaným firmvérom.

Vyvinul ho Bjarne Stroustrup (pôvodne nazvaný "C with Classes") v Bell Labs v roku 1983 ako rozšírenie jazyka C pretože mal všetky vlastnosti jazyka C s ďalším konceptom "tried". Napriek tomu bol v roku 1983 premenovaný na C ++. (1)

Rozširovanie začalo pridaním tried a neskôr pokračovalo pridávaním ďalších vlastností ako sú virtuálne funkcie, prekrývanie operátorov, viacnásobná dedičnosť, šablóny a ošetrovanie výnimiek. Jazyk má statickú typovú kontrolu, podporuje procedurálne programovanie, dátovú abstrakciu, objektovo orientované programovanie, ale aj generické programovanie. Hlavným vrcholom C ++ je kolekcia preddefinovaných tried. Jazyk tiež uľahčuje deklarovanie používateľsky definovaných tried. Triedy môžu ďalej prispôbovať funkcie členov, aby mohli implementovať špecifické funkcie. Mnohé objekty určitej triedy

môžu byť definované tak, aby implementovali funkcie v rámci triedy. Objekty môžu byť definované ako inštancie vytvorené v čase spustenia. Tieto triedy môžu byť zdedené aj inými novými triedami, ktoré štandardne využívajú verejné a chránené funkcie.

Štandard jazyka C++ bol schválený v roku 1998 ako ISO/IEC 14882:1998, aktuálna verzia je z roku 2003 (ISO/IEC 14882:2003). Nová verzia štandardu (známa pod označením C++11 alebo C++0x) bola schválená v auguste 2011. Od 90-tych rokov 20. storočia patrí k najpopulárnejším programovacím jazykom, používa ho až vyše 95% engine-ov počítačových hier. C++ sa považuje za jazyk strednej úrovne, pretože zahŕňa jazykové funkcie vysokej aj nízkej úrovne.

3.4.2.3. *Knižnica OpenCv*

Na úspešné dosiahnutie cieľa diplomovej práce sme sa z dôvodu veľkej efektívnosti rozhodli využiť užitočné funkcie knižnice pre počítačové videnie OpenCv. Ide o knižnicu, ktorá dokáže pracovať s obrazom a kombináciou jej viacerých nástrojov sa vieme dopracovať aj k nájdeniu gesta ruky na snímke z webkamery. Knižnica je celosvetovo veľmi populárna a veľmi často využívaná. Jej najväčšou výhodou je najmä jej multijazyčnosť a multiplatformovosť.

Knižnica OpenCV obsahuje približne 2500 najmodernejších a najoptimálnejších počítačových algoritmov z oblasti počítačového videnia a strojového učenia. Ich využitie je rôzne napríklad rozpoznávanie mimiky či detekcia pohybu. OpenCv sa masívne používa v rôznych vedeckých zoskupeniach či bezpečnostných systémoch. Medzi jej najvýznamnejšími používateľmi môžeme nájsť aj spoločnosti ako IBM, Microsoft, YouTube či Facebook. (2)

V súčasnosti je OpenCv knižnica podporovaná operačnými systémami ako Android, Windows, Mac OS či Linux a má zabezpečené rozhranie pre programovacie jazyky C, C++, Python a Matlab. Samotná OpenCv je primárne napísaná v jazyku C++ a je ju možné efektívne využívať s jeho súčasťami ako napríklad „kontajnery“.

Pri inštalácii knižnice sme sa nezaobišli bez miernych ťažkostí. Keďže je knižnica primárne určená pre procesory typu Intel, bola jej inštalácia trocha zložitejšia. Nemohli sme ju priamo nainštalovať, ale manuálne skompilovať, aby bola funkčná na Raspberry PI s ARM procesorom. Táto kompilácia trvala približne 4 hodiny.

4. Opis riešenia

V tejto kapitole diplomovej práce sa pozrieme na podrobné implementačné riešenie našej aplikácie, ktorá by mala spĺňať ciele definované v návrhu (Kap.2), a mala by byť schopná rozpoznávať gestá dlane zosnímané webkamerou a na základe rozpoznaného gesta odoslať pokyn pre UAV. Na konci môžeme vidieť ako výsledný program vyzerá po spustení.

3.1. Príprava hardvéru

Ešte pred tým než začneme s implementáciou, musíme si nutne zabezpečiť všetok potrebný hardvér. Pripravili sme si teda miniatúrne zariadenie Raspberry pi 3 (ideálne aj s chladiacim ventilátorom kvôli možnému prehriatiu pri náročnejších výpočtoch), pripojili sme k nemu našu USB webovú kameru a cez sériovú USB linku sme pripojili k Raspberry dosku Arduino Uno. Arduino je prepojené s UAV pomocou GPIO pinov, ktorými posiela signály na jeho ovládanie.

Raspberry PI 3 je však minipočítač, ktorý nemá žiadne integrované vstupné a ani výstupné zariadenia. Aby sme mohli aplikáciu vyvíjať, je nutné zabezpečiť nejaké používateľské rozhranie pre prácu s Raspberry. Sú v podstate tri možnosti ako môže používateľ s Raspberry PI 3 komunikovať, ktoré si teraz bližšie popíšeme.

1. **Pomocou SSH klienta** – Prvou možnosťou je komunikácia pomocou SSH klienta cez SSH protokol.

Ide o zabezpečený prístup k príkazovému interpretovaču (angl. secure shell), skr. SSH, je v informatike počítačový program ako aj súvisiaci sieťový protokol určený na prihlasovanie a vykonávanie príkazov na vzdialenom počítači v počítačovej sieti. Návrhári SSH mali v úmysle nahradiť skôr používané málo bezpečné protokoly rlogin, telnet a rsh a výsledný protokol poskytuje bezpečnú kryptovanú komunikáciu medzi dvoma neautentifikovanými strojmi v nezabezpečenej sieti. (3)

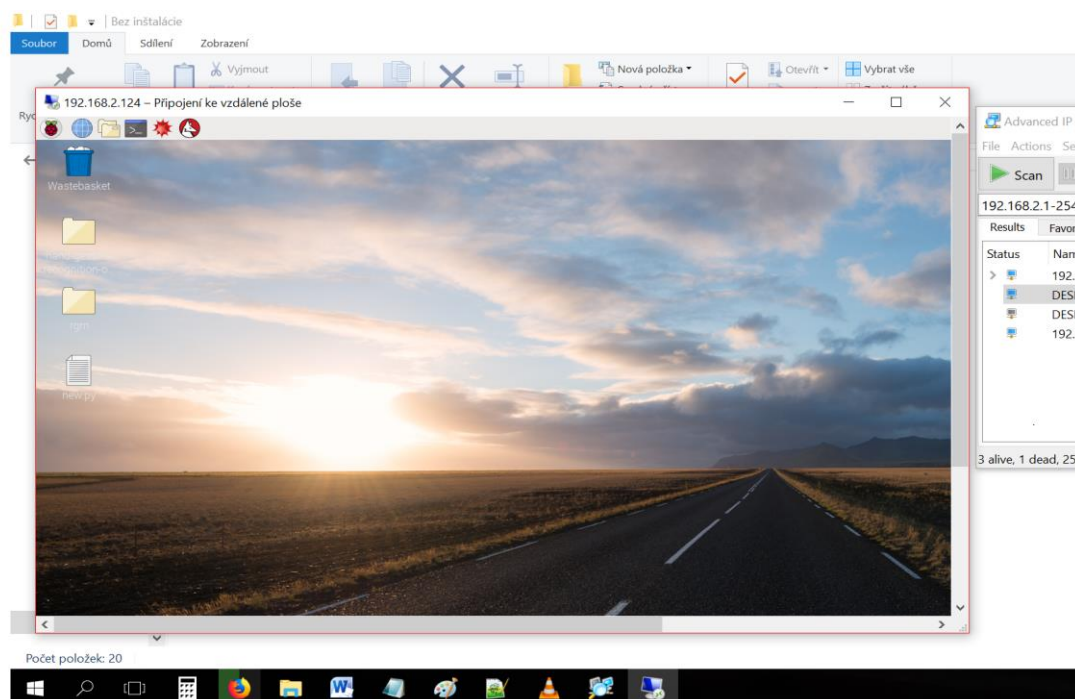
Táto možnosť bola pre nás nevyhovujúca pretože príkazový interpretovač nám neponúka žiadne grafické rozhranie a teda by nebolo možné sledovať vstup z webkamery a testovať správnosť spracovania obrazu.

2. **Pripojiť externé zariadenia** - Druhou možnosťou pre ovládanie Raspberry PI 3 je pochopiteľne pripojenie vlastných zariadenia ako myš, klávesnica a HDMI monitor.

Vzhľadom nato, že sme nemali k dispozícii vlastný externý monitor ani klávesnicu ani tento prístup sme nevyužili.

3. **Nainštalovať XRDP klienta** – Zvolili sme si možnosť XRDP klienta. XRDP je služba určená pre Unix systémy bežiaca na pozadí operačného systému. Umožňuje komunikovať pomocou RDP protokolu s inými zariadeniami. **RDP** je protokol spoločnosti Microsoft určený na uľahčenie zabezpečenia prenosu dát aplikácií a šifrovanie medzi klientmi, zariadeniami a virtuálnym sieťovým serverom. Umožňuje vzdialenému používateľovi pridať grafické rozhranie na pracovnú plochu iného počítača. Na základe súboru protokolov ITU-T.120 je protokol RDP kompatibilný s viacerými typmi protokolov a topológií lokálnej siete (LAN). (8)

Pre používanie XRDP klienta je ho potrebné najskôr na Raspberry PI 3 nainštalovať cez terminál pomocou príkazu „**sudo apt-get install xrdp**“. Po následnom reštartovaní zariadenia a jeho pripojení k lokálnej sieti LAN je možné do zariadenia prístupiť z iného zariadenia. Na Windows zariadeniach stačí napríklad spustiť aplikáciu Pripojenie k vzdialenej ploche. Je potrebné zadať IP adresu a prihlasovacie údaje. Následne môžeme pracovať s Raspberry priamo v desktopovom režime z iného počítača. Na nasledujúcom obrázku môžeme vidieť RDP klient vo Windows komunikujúci s XRDP klientom na Raspberry.



Obrázok 7 – RDP klient vo Windows

3.2. Príprava softvéru

Po úspešnej príprave hardvéru a zabezpečení používateľskej komunikácie so zariadením Raspberry PI bude potrebné nainštalovať aj potrebný softvér. Na raspberry sme ako operačný systém zvolili zrejme najpoužívanejší Raspbian. Systém Raspbian predvolene neobsahuje knižnice pre pokročilú prácu s multimediami OpenCv preto je nutné túto knižnicu manuálne doinštalovať čo môže trvať aj pár hodín – podrobný popis ako nainštalovať OpenCv na Raspbian je popísaný v **prílohe B**.

Ďalším softvérom, ktorý musíme nainštalovať je vývojové prostredie. Ako programovací jazyk sme si zvolili C++. Ponúkajú sa dve možnosti vyvíjať aplikáciu vo vývojovom prostredí alebo ju písať v textovom editore a kompilovať ju cez terminál. Ak by bola aplikácia robustnejšia určite by sme zvolili prvú možnosť ale v našom prípade nám pohodlne postačí kompilácia pomocou gcc kompilátora.

3.3. štruktúra kódu

Na začiatok implementácie bolo potrebné importovať niekoľko OpenCv knižníc, aby sme ďalej v kóde mohli využívať OpenCv nástroje.

```
#include "opencv2/objdetect/objdetect.hpp"

#include "opencv2/highgui/highgui.hpp"

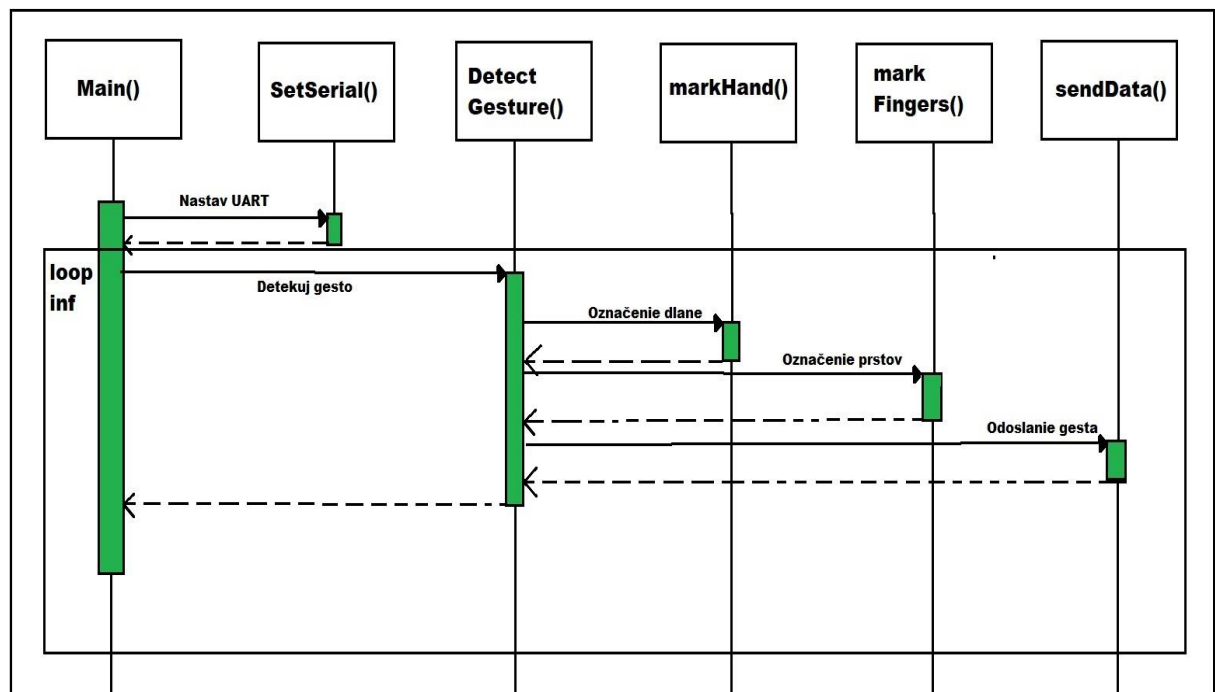
#include "opencv2/imgproc/imgproc.hpp"
```

Ďalej zdrojový kód aplikácie pozostáva z nasledovných funkcií, ktorých význam si teraz stručne popíšeme a následne zobrazíme v UML diagrame

- **int Main()** – Hlavná funkcia programu v ktorej cyklicky prebieha snímanie obrazu z webkamery. Po každom zosnímaní jednej snímky je tento frame odoslaný na spracovanie do funkcie **DetectGesture**. Po stlačení klávesy Esc hlavný cyklus skončí čím sa zároveň ukončí program.
- **void setSerial()** – Funkcia slúži na počiatočné otvorenie a nastavenie sériového portu a jeho presmerovanie na deskriptor.

- **void DetectGesture(cv::Mat frame)** – Funkcia ktorá najskôr aplikuje na snímku rôzne filtre pre zjednodušenie detekcie gesta a následne volá funkcie na zistenie pozície dlane a prstov. Tieto pozície potom graficky znázorní a zavolá funkciu **sendData()**.
- **void sendData(int n)** – Funkcia odošle pomocou prideleného deskriptora cez sériovú linku dáta pre Arduino Uno na základe počtu zosnímaných prstov a vypíše spätné dáta od UAV.
- **bool markHand(cv::Mat frame, vector<Point> cont, Point *rPoint,int *radius)** – Funkcia ktorá sa snaží z upravenej snímky určiť pozíciu a veľkosť dlane v zábere čo je následne nápomocné pri určovaní pozícií prstov.
- **vector<Point> mark_fingers(Mat frame, vector<Point> hull,Point pt,int radius, int *fingerCnt)** – Funkcia, ktorá určí počet prstov a pre každý prst určí aj jeho presnú polohu na snímke. Táto poloha sa následne zobrazí aj graficky.

Nasledujúci UML diagram znázorňuje beh aplikácie:



Graf 3 – Sekvenčný UML diagram behu aplikácie

3.4. Popis kódu

Teraz sa už môžeme priamo pozrieť na náš zdrojový kód, v ktorom si vysvetlíme jeho jednotlivé časti a najmä zložitejšie algoritmy na detekciu pozície dlane a prstov.

3.4.1. Nastavenie UART

Prvým krokom ktorý naša aplikácia urobí je, že otvorí sériový port, presmeruje ho na deskriptor a nastaví jeho parametre.

Sériový port

Sériový port (angl. UART) alebo štandard RS-232 alebo **sériová linka** je komunikačné rozhranie umožňujúce dvom propojeným objektom či komponentom prevádzkovať takzvanú sériovú komunikáciu dvoch zariadení, čiže dáta sa prenášajú po jednotlivých bitoch postupne v sérii za sebou po jednom. Podobne sériovo komunikujú aj Ethernet a USB.

Sériový port patrí medzi tzv. legacy porty (zastarané) a je nahrádzaný rýchlejším USB. Prenos dát prebieha najčastejšie pomocou 8 bitov a reprezentácia logických hodnôt 1/0 je spôsobená presnou hodnotou napätia. Najviac používané sú $\pm 12V$. $-12V$ pre logickú hodnotu 1 a $+12V$ pre nulu. V súčasnosti sa sériový port už veľmi nepoužíva. V niektorých odvetviach priemyslu však má RS-232 stále nenahraditeľné miesto. (9)

Sériový port otvoríme pomocou štandardnej funkcie Open. Funkcia nám vráti číslo systémového deskriptoru, ktorý nám bol pridelený a pomocou ktorého budeme môcť cez sériovú linku posielať dáta alebo ich z nej prijímať.

```
int USB = open( "/dev/ttyUSB0", O_RDWR | O_NOCTTY | O_NONBLOCK);
```

Treba si všimnúť dve veci. Prvý parameter **/dev/ttyUSB0**. Ide o cestu k DEV súboru, ktorý predstavuje pripojené UART zariadenie. V tomto prípade môže nastať problém pri opätovnom odpojení a pripojení. Systém totiž môže pri každom pripojení vybrať iný DEV súbor. Najčastejšie je to jeden zo súborov **/dev/ttyUSB0** alebo **/dev/ttyACM0**. Ak zadáme nesprávny DEV súbor funkcia open skončí neúspechom. Preto je vhodné testovať oba súbory. Ak pri prvom vráti funkcia open hodnotu -1 skúsime druhý súbor.

Druhým parametrom, ktorý funkcia prijíma sú tzv. flags teda možnosti práce so súborom. Prvá možnosť **O_RDWR** nám hovorí že cez sériovú linku chcem dáta čítať aj zapisovať. Možnosť **O_NONBLOCK** nám zasa zaistí, aby sa program počas behu nezablokoval v prípade, že zariadenie na druhom konci sériovej linky neposiela alebo neprijíma dáta.

3.4.1.1. KonfiguráciaUART

Po úspešnom otvorení sériového portu ho ešte treba správne nakonfigurovať. Preto hneď po spustení programu voláme funkciu **setSerial()**, ktorá nastaví niektoré vlastnosti sériovej linky. Prejdeme si aspoň jej najdôležitejši časti.

```
struct termios tty;
struct termios tty_old;
memset (&tty, 0, sizeof tty);

/* Error Handling */
if ( tcgetattr ( USB, &tty ) != 0 ) {
    std::cout << "Error " << errno << " from tcgetattr: " << strerror(errno) << std::endl;
}
```

V tejto časti zisťujeme či je sériová linka funkčná a v prípade, že nie vypíšeme informáciu o chybe.

```
/* Set Baud Rate */
cfsetospeed (&tty, (speed_t)B115200);
cfsetispeed (&tty, (speed_t)B115200);
```

Nastavíme prenosovú rýchlosť odosielania aj prijímania na 115200 bitov za sekundu. Tu je dôležité vedieť, akú prenosovú rýchlosť má druhé zariadenie a nastaviť ju rovnako, pretože inak sa nám nepodarí správne prečítať rovnaké dáta na oboch koncoch sériovej linky.

```
tty.c_cc[VMIN] = 1;           // read doesn't block
tty.c_cc[VTIME] = 0.5;       // 0.5 seconds read timeout
```

Tu nastavujeme, aby sa nezablokovala funkcia `read()` v prípade, že čaká na dáta od druhého zariadenia. Ako môžeme vidieť je potrebné túto vlastnosť nastaviť dvakrát. Prvýkrát vo funkcii `open()`. Hodnota 0.5 hovorí že ak funkcia `read` nedostane dáta do 0.5 sekundy tak bude program pokračovať ďalej.

V tejto chvíli by sme mali byť schopný plne využívať sériovú linku a komunikovať cez ňu s Arduino UNO. Pri testovaní sme na Arduino UNO exportovali jednoduchý program ktorý nám vracal rovnaké dáta ako prijal.

3.4.1. Snímanie videa

Na snímanie obrazu z webkamery sme použili OpenCv triedu `VideoCapture`. `VideoCapture` je trieda pre snímanie videa z kamier, video súborov alebo sekvencií obrázkov.

```
cv::VideoCapture capture(0);
```

Hodnota 0 je identifikačné číslo kamery.

Prvá vec, ktorú sme urobili bolo nastavenie požadovanej snímacej frekvencie pomocou príkazu `capture.set(CAP_PROP_FPS, 30)` na hodnotu **30** snímok za sekundu i keď zariadenie ako je Raspberry niekedy nemusí 30 snímok stíhať.

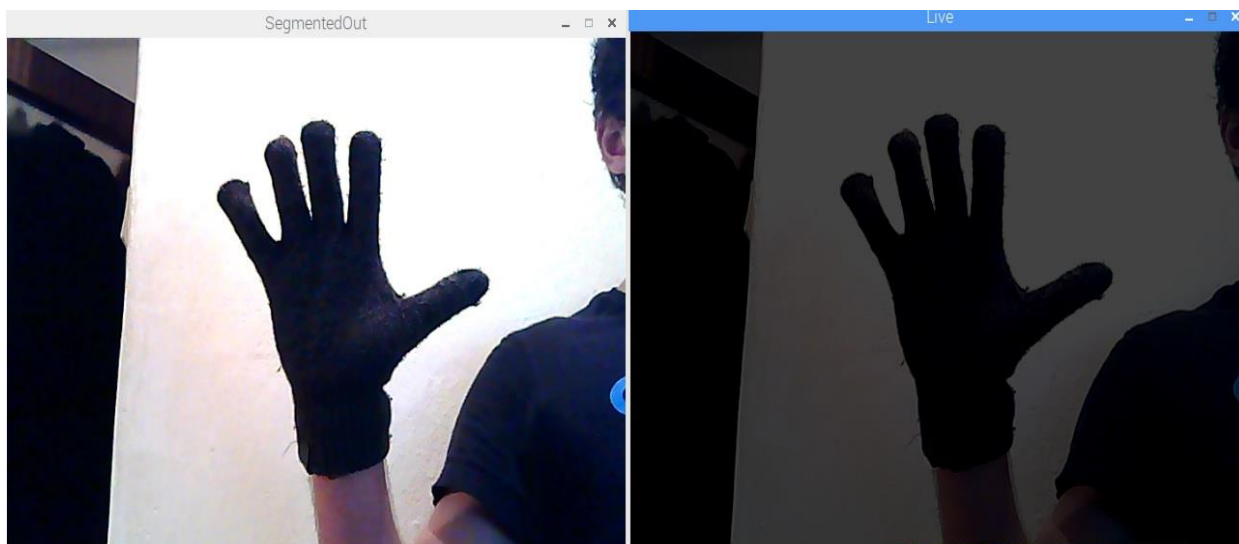
```
while (capture.isOpened() {
    capture >> img;
    if (!img.empty())
        DetectGesture(img);
}
```

Ako môžeme vidieť program beží celú dobu v cykle a vždy keď je z kamery získaná snímka, tak táto je poslaná do funkcie `DetectGesture()`, ktorá sa pokúsi na snímke vyhľadať ruku a gesto.

3.4.2. Aplikácia filtrov

Keďže rozpoznávanie gesta bez úpravy zosnímaného obrázku by nebola možná, je potrebné zosnímaný obrázok nejakým spôsobom upraviť. Našťastie OpenCv ponúka množstvo nástrojov. Prvou úpravou, ktorú sme vykonali bola úprava kontrastu. Je to z dôvodu toho že menšia svetelnosť objektov zvyšuje schopnosť rozpoznávať ich obrysy. V OpenCv nato slúži funkcia `void convertTo(OutputArray m, int rtype, double alpha = 1, double beta = 0)`. Nás zaujímali najmä atribúty **alpha** a **beta**. Alpha má za následok vynásobenie hodnoty každého pixelu a beta má za následok pripočítanie hodnoty ku každému pixelu $\text{color} = \text{color} * \alpha + \beta$. My sme po niekoľkonásobnom testovaní zvolili hodnoty **0.3** a **-10**.

```
frame.convertTo(frame,-1,0.3,-10);
```



Obrázok 8 – Obrázok použitia contrastného filtra

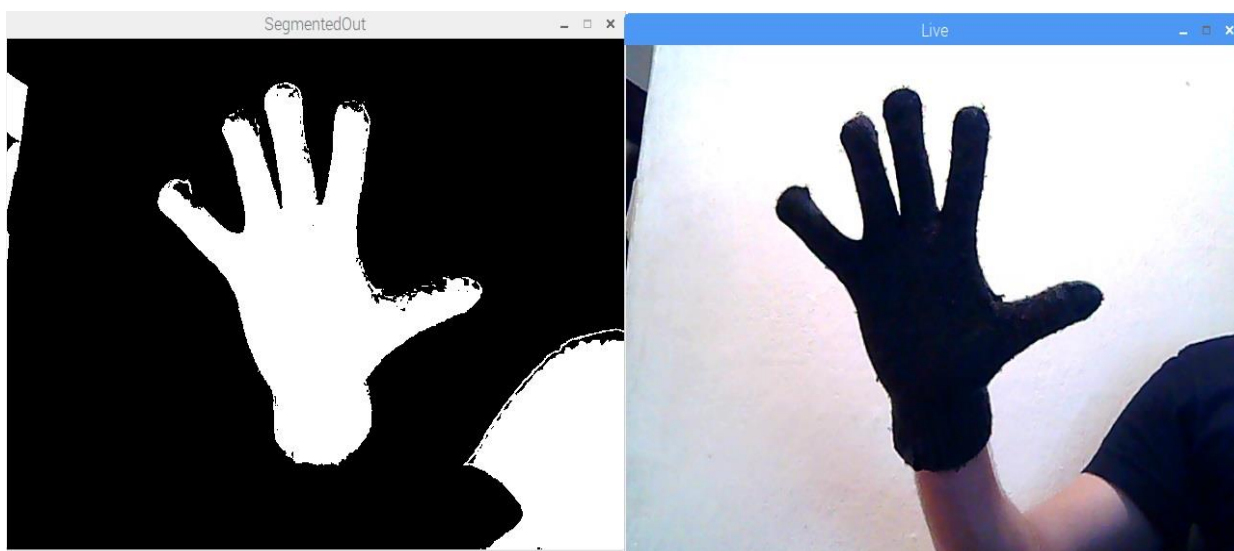
Druhým filtrom v poradí je aplikácia čiernobieleho filtra. Je to preto že farby obrázku pre nás nie sú podstatné a úprava sivých odtieňov je vhodnejšia a jednoduchšia pre ďalší postup. Nasledovná funkcia obrázok zbaví farieb.

```
cvtColor(frame, img, COLOR_BGR2GRAY); //black and white
```

3.4.2.1. Prahový filter

```
cv::threshold(img, binary, 35, 255, cv::THRESH_BINARY_INV);
```

Tento filter je zo všetkých použitých filtrov najdôležitejší. Funkcia sa vzťahuje na prahové hodnoty. Zvyčajne sa používa na získanie dvoj-úrovňového (binárneho) obrazu z obrazu v odtieňoch sivej. Okrem toho sa dá funkcia použiť aj na odstránenie šumu, to znamená filtrovanie pixelov s príliš malými alebo príliš veľkými hodnotami. Najdôležitejší parameter je tretí parameter s hodnotou 35, ktorý hovorí o tom na akú hodnotu má byť nastavený prah citlivosti farby (od akej úrovne nastaví čiernu farbu). Najlepšie to pochopíme podľa nasledujúceho obrázka.



Obrázok 9 – Obrázok použitia prahového filtra

3.4.2.2. Erode a Dilate filtre

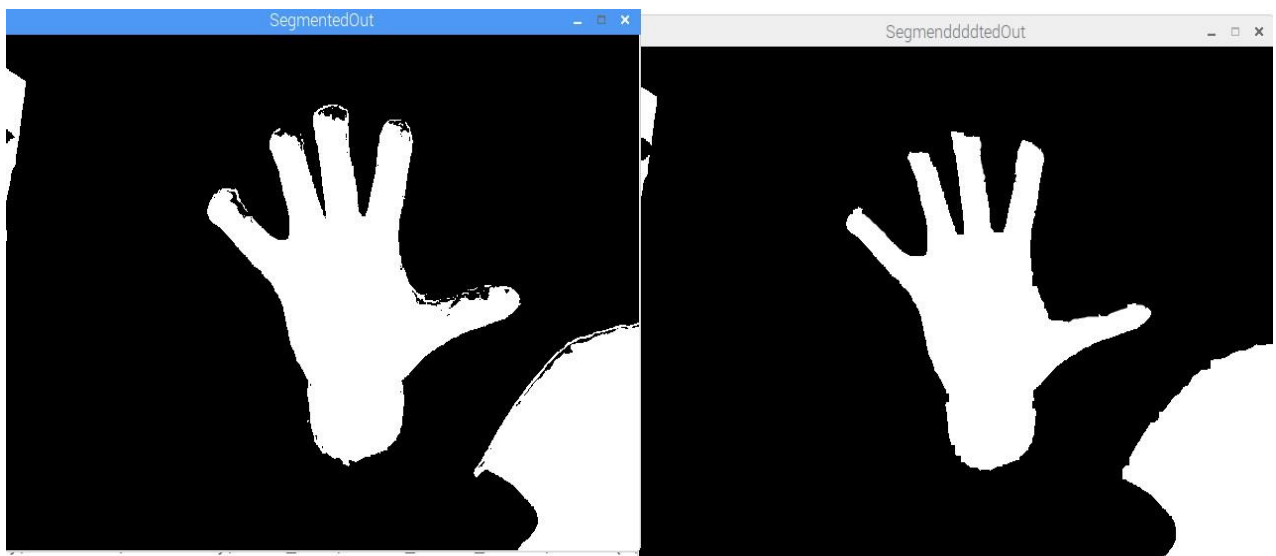
Posledné dva filtre, ktoré použijeme sú takzvané tvarovacie filtre. Tvarovacie filtre sú sada operácií, ktoré spracúvajú obrázky na základe ich tvarov. Majú širokú škálu využití ako napríklad odstránenie šumu alebo izolácia jednotlivých prvkov a spojenie rozdielnych prvkov v obraze. Najpoužívannejšie tvarovacie filtre sú **Erode** a **Dilate**, ktoré sú najčastejšie aplikované na binárne (dvoj-farebné) obrázky.

Erode

```
erode(binary, binary, getStructuringElement(MORPH_RECT, Size(4, 4)));
```

Zmysel funkcie je vypočítať lokálne minimum v určitej oblasti obrázka. Keď sa oblasť naskenuje, vypočíta sa minimálna hodnota pixelov z hľadiska prevahy tmavej alebo bledej farby a nahradíme obrazové body v oblasti touto minimálnou hodnotou. Takýmto spôsobom môžeme napríklad opraviť drobné nedostatky v obrázku. Parameter `getStructuringElement(MORPH_RECT, Size(4, 4))` nám určuje veľkosť prehľadávaných oblastí z obrázka.

Týmto postupom rastú na obrázku vo veľkosti tmavé oblasti a bledé oblasti sa zmenšujú. Napríklad veľkosť objektu v tmavom alebo čiernom odtieni sa zvyšuje, kým klesá veľkosť objektu v bielej farbe. Lepšie to pochopíme z nasledovného obrázka.



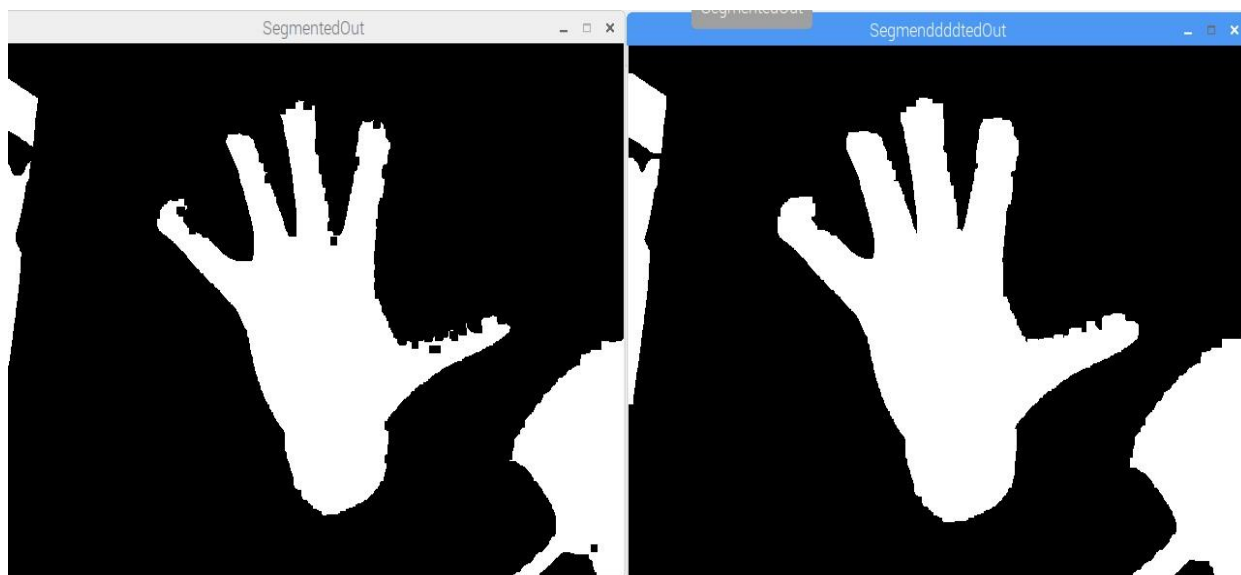
Obrázok 10 – Obrázok použitia funkcie Erode

Dilate

```
dilate(binary, binary, getStructuringElement(MORPH_RECT, Size(9, 9)));
```

Táto operácia je opakom **Erode**. Rozdiel je v tom, že po tom čo sa oblasť naskenuje, vypočíta sa maximálna hodnota pixelov z hľadiska prevahy tmavej alebo bledej farby a obrazové body v tejto oblasti sú nahradené touto maximálnou hodnotou.

Týmto postupom na rozdiel od **Erode** na obrázku rastú vo veľkosti bledé oblasti a tmavé oblasti sa zmenšujú. Opäť to môžeme vidieť na obrázku – vpravo je obrázok po vykonaní funkcie Dilate. Touto funkciou sa taktiež odstránia niektoré nedokonalosti obrysov.



Obrázok 11 – Obrázok použitia funkcie Dilate

V zdrojovom kóde používame najskôr funkciu **erode** s veľkosťou oblasti štyri a to hneď dva krát za sebou čo zlepšuje jej efekt. Za ňou nasleduje funkcia **dilate** s veľkosťou oblasti deväť. Oba tvarovacie filtre nám pomáhajú odstrániť niektoré chyby zosnímanej kontúry dlane a zároveň jej kontúry viac zvýrazňujú.

3.4.3. Nájdenie kontúr

Po aplikovaní rôznych filtrov nám na obrázku lepšie vyniknú kontúry. Kontúru v tomto prípade môžeme chápať ako zoznam bodov tvoriacich súvislú alebo čiastočne prerušovanú líniu, ktorá je zároveň obrysom nejakého objektu na obrázku. V tomto prípade nám prácu veľmi uľahčí OpenCv funkcia findContours.

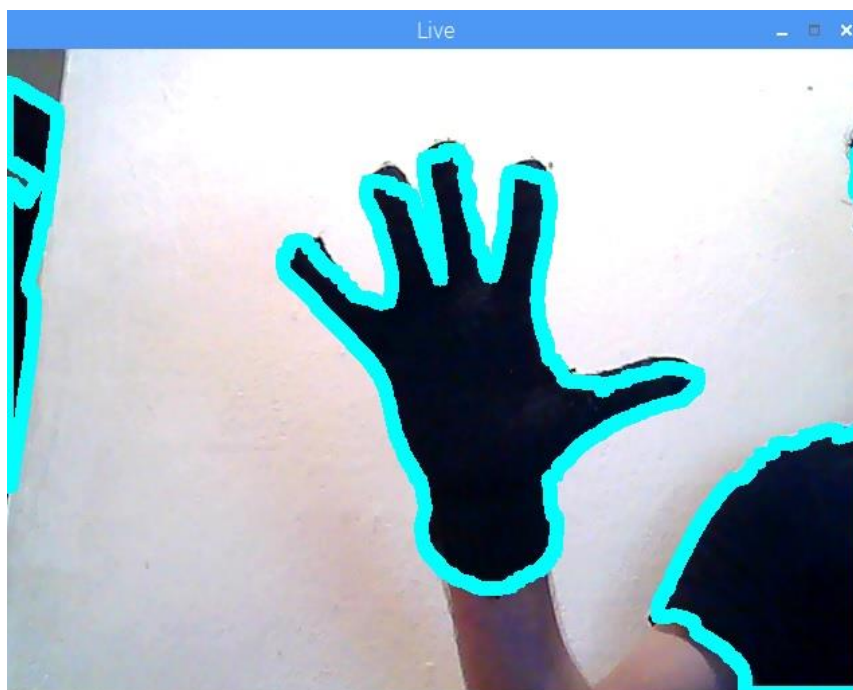
```
findContours(binary, Contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE);
```

Funkcia z nášho čiernobieleho obrázka ľahko určí kontúry všetkých bielych objektov a vráti nám ich ako vektor bodov v premennej Contours.

Na obrázku však môže byť viacero objektov pričom každý z nich má svoju vlastnú kontúru. Funkcia ale nedokáže určiť ktorá kontúra patrí našej ruke a ktorá nie. Preto budeme predpokladať že dlaň bude na obrázku vždy najväčší objekt. Potrebujeme teda určiť ktorá kontúra má najväčšiu plochu.

```
for (int i = 0; i < Contours.size(); i++) {  
    double currentArea = contourArea(Contours[i]);  
    if (currentArea > maxarea)  
    {  
        maxarea = currentArea;  
        biggestContour = i;  
    }  
}
```

Použijeme nato jednoduchý for cyklus, ktorý prehľadáva všetky nájdené kontúry. Opäť nám tu veľmi pomôže OpenCv funkcia **contourArea(Contours[i])**; ktorá je schopná zo vstupnej kontúry vypočítať jej plochu. Takýmto spôsobom získame informáciu o tom, ktorá kontúra je najväčšia a uložíme ju do premennej biggestContour.



Obrázok 12 – Obrázok určenia kontúr v obrázku

3.4.4. Nájdenie stredu dlane

Teraz keď máme s najväčšou pravdepodobnosťou určenú pozíciu najväčšej kontúry, môžeme z nej určiť pozíciu dlane. Nachádzame sa vo funkcii `markHand()`, ktorá na svojom vstupe prijíma štyri premenné:

- **cv::Mat** frame – snímka z kamery, v ktorej hľadáme pozíciu dlane
- **vector<Point>** cont – zoznam bodov najväčšej kontúry
- **Point *rPoint** – pointer na premennú, do ktorej zapíšeme polomer dlane
- **int *radius** – pointer na premennú do ktorej zapíšeme pozíciu dlane

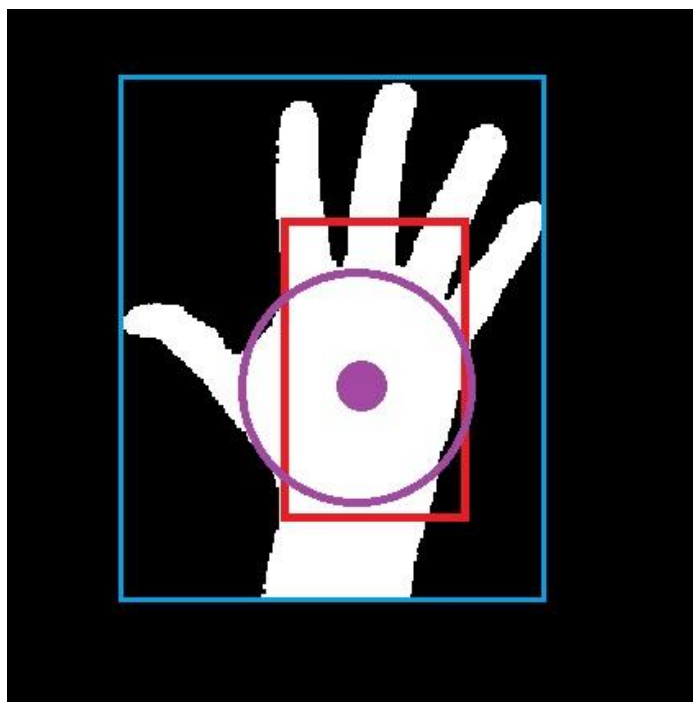
Prvou vecou, ktorú urobíme je, že získame opísaný štvorec našej kontúry pomocou funkcie `Rect rect = boundingRect(cont)`. Vieme teda, že nami hľadaný bod bude určite niekde vo vnútri tohto štvorca. Nemôžeme však robiť výpočet pre každý jeden bod zo štvorca. To by bolo príliš časovo náročné (najmä na zariadení ako je Raspberry PI), preto budeme prechádzať iba niekoľko strategických bodov štvorca čo úplne postačuje.

```
int ind_y;
int ind_x;

for (ind_y=int(y+0.3*h); ind_y <= int(y+0.8*h); ind_y+=5)
    for ( ind_x=int(x+0.3*w); ind_x <= int(x+0.6*w) ; ind_x+=5) {
        int dist= pointPolygonTest(cont,Point2f(ind_x,ind_y),true);
        if (dist>max_d) {
            max_d=dist;
            pt.x=ind_x;
            pt.y=ind_y;
        }
    }
}
```

Ako môžeme vidieť prehľadávame iba 50% plochy štvorca a zároveň iba každý piaty bod. Všimnime si riadok `int dist= pointPolygonTest(cont,Point2f(ind_x,ind_y),true);`

OpenCv funkcia `pointPolygonTest()` nám vracia vzdialenosť od najbližšieho bodu z množiny `cont` (body určujúce kontúru). Ak je táto vzdialenosť väčšia ako aktuálna hodnota premennej `max_d`, tak ju nahradíme a zároveň si zapamätáme aktuálny bod. Po skončení cyklu tak máme v premennej `pt` uložený bod, ktorý je euklidovskou vzdialenosťou najďalej od všetkých bodov kontúry. Tento bod prehlásime za stred dlane a vzdialenosť bodu od najbližšieho bodu kontúry nám zároveň definuje polomerovú veľkosť dlane.



Obrázok 13 – Znázornenie vyhľadávania pozície dlane

V závere funkcie máme ešte dôležitú podmienku, ktorá rozhoduje o tom či náš výpočet pozície dlane bude relevantný.

```
If (max_d > 0.04*frame.cols)
    return true;
else
    return false;
```

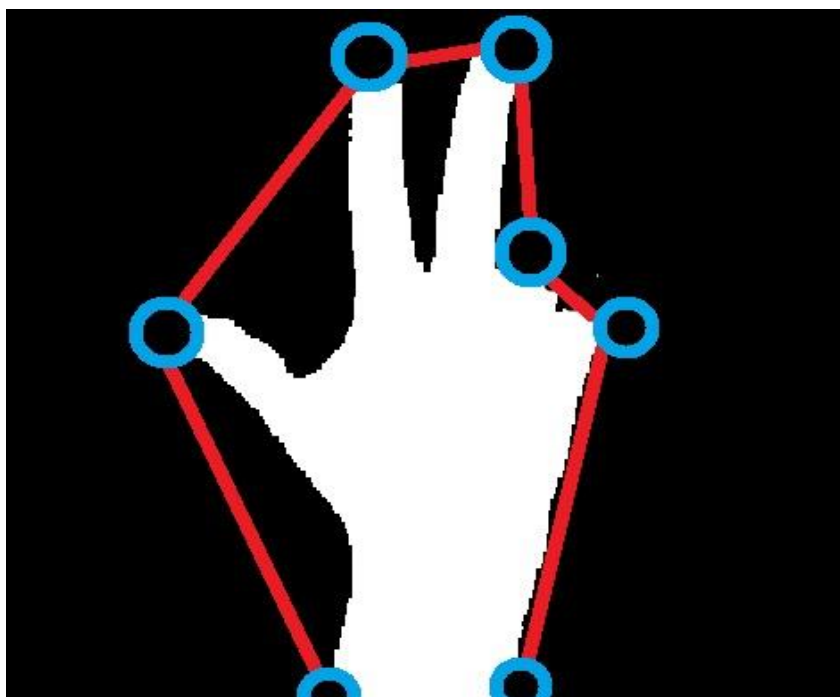
Ak teda polomer dlane netvorí aspoň 4% šírky snímky, funkcia skončí neúspechom a zistená pozícia dlane bude programom ignorovaná.

3.4.5. Nájdenie pozície prstov

Už máme určenú pozíciu a veľkosť dlane, ale k dosiahnutiu cieľa budeme potrebovať určiť pozíciu každého z prstov ruky a zároveň ich počet. Ešte predtým ako zavoláme funkciu `markFingers()` musíme zavolať funkciu:

```
convexHull(Contours[biggestContour], cnvxHull, false);
```

Význam funkcie spočíva vo vrátení konvexných bodov našej kontúry, teda body ktoré sú súčasťou opísaného útvaru. Čo to presne znamená si demonštrujeme na obrázku:



Obrázok 14 – Konvexné body ruky

Funkcií na vstupe posielame nasledovné parametre:

Contours[biggestContour] – najväčšia kontúra - kontúra našej ruky.

cnvxHull – vektor bodov do ktorého funkcia zapíše nájdené body.

false – booleovská hodnota hovoriaca že výstupné body majú byť v smere hodinových ručičiek čo však v našom prípade nemá žiadny vplyv.

Z obrázku je zjavné, že časť práce za nás urobila funkcia `convexHull`. Keďže nás budú zaujímať iba body, ktoré určujú pozície vztýčených prstov, bude našou úlohou tieto body odfiltrovať od nechcených zbytočných bodov.

Zavoláme teda našu funkciu `mark_fingers()`, ktorá prijíma nasledovné argumenty:

Mat frame – Snímka z kamery v ktorej hľadáme pozíciu dlane.

vector<Point> hull – Výstupné body z funkcie `convexHull`.

Point pt – Pozícia dlane, ktorú už máme vypočítanú.

int radius – Polomer dlane, ktorý už máme vypočítaný.

int *fingerCnt – Pointer na premennú, do ktorej zapíšeme počet prstov.

Prvým krokom bude, že odstránime tie body ktoré sú príliš blízko seba a teda s určitosťou nereprezentujú dve rôzne pozície prstov. Môžeme si všimnúť, že vyhovujúce body z vektora `hull` presúvame do nového vektora `finger`.

```
for(i=0;i<hull.size()-1;i++) {
    float dist = sqrt( (hull[i].x - hull[i+1].x)*(hull[i].x - hull[i+1].x) + (hull[i].y - hull[i+1].y)*(hull[i].y - hull[i+1].y) );
    if (dist>18) {
        finger.push_back( Point(hull[i].x,hull[i].y) );
    }
}
```

Ďalším krokom je že odfiltrujeme tie body, ktoré sa nachádzajú príliš blízko alebo príliš ďaleko od stredu dlane – teda nejde o body vztýčených prstov. Tie sa od stredu dlane musia nachádzať v istom rozmedzí. A tak isto sa zbavíme aj bodov ktoré sa nachádzajú pod dlaňou keďže predpokladáme že dlaň bude na kamere prstami smerom nahor.

```
int temp_len=finger.size();
i=0;
while(i<temp_len) {
    float dist = sqrt( (finger[i].x - cx)*(finger[i].x - cx) + (finger[i].y - cy)*(finger[i].y - cy) );
    if(dist<2*radius || dist>3.8*radius || finger[i].y>cy+radius) {
        finger.erase(finger.begin()+i);
        temp_len=temp_len-1;
    }
    else
        i=i+1;
}
```

Konštanty $2*\text{radius}$ a $3.8*\text{radius}$ sme získali postupným skúšaním rôznych vyhovujúcich hodnôt.

História počtu prstov

Na konci funkcie sme ešte zabezpečili aby neočakávané krátkodobé zmeny svetelných podmienok nespôsobovali častú nestálosť v počte detekovaných prstov. To sme vyriešili takzvanou históriou počtu prstov. Znamená to že počet prstov nie je počítaný priamo na základe počtu prvkov vektora `finger` ale nasledovne.

```
if(first_iteration) {
    finger_ct_history[0]=finger_ct_history[1]=finger.size();
    first_iteration=false;
}
else
    finger_ct_history[0]=0.34*(finger_ct_history[0]+finger_ct_history[1]+finger.size());

if( (finger_ct_history[0]-int(finger_ct_history[0])) > 0.8)
    finger_count=int(finger_ct_history[0])+1;
else
    finger_count=int(finger_ct_history[0]);

finger_ct_history[1]=finger.size();
*fingerCnt = finger_count;
```

Máme pole s dvoma prvkami `finger_ct_history[0]` a `finger_ct_history[1]`. Pri prvej iterácii sa ich hodnota nastaví na `finger.size()`; Pri každej ďalšej iterácii je počet prstov počítaný vzorcom:

```
finger_ct_history[0]=0.34*(finger_ct_history[0]+finger_ct_history[1]+finger.size());
```

Následne počet prstov zaokrúhlime podľa odchýlok v histórii. Takýto postup nám zabezpečí väčšiu stabilitu pri nepatrných pohyboch ruky, ktoré môžu spôsobiť chvíľkové chyby detekcie.

3.4.6. Komunikácia s Arduino UNO

Na komunikáciu s Arduino UNO využívame sériovú linku, ktorú už máme namapovanú na deskriptor v premennej USB. Teraz už môžeme posielat' a prijímať dáta tradičným linuxovým spôsobom – funkciami read a write.

```
write( USB, &message, 1 );
```

Do premennej message môžeme umiestniť ľubovoľne dáta, ale my sme zvolili posielanie jediného čísla, ktoré predstavuje počet detekovaných prstov. Rovnako ako zapisujeme môžeme z deskriptora aj čítať.

```
int i=0;
int cnt;
char buf;
do
{
    cnt = read( USB, &buf, 1 );
    if (cnt < 1)
        response[i]='\0';
    else
        response[i] = buf;
    i++;
} while(buf != '\n' && cnt > 0);

printf("Response: %s\n",response);
```

Opäť je to tradičný linuxovský prístup pre čítanie z deskriptora. Je nutné upozorniť, že funkcia read() sa môže zablokovať ak nedostane nič čo by mohla čítať. Z toho dôvodu je veľmi dôležité pri otváraní deskriptora nezabudnúť nastaviť parametre **O_NOCTTY** a **O_NONBLOCK**.

Naše Arduino nám späť posiela informácie od UAV ako napríklad aktuálna vzdialenosť od zeme.

```
Response: SetPoint [mm]:39.00 Vyska [cm]:24.50 Vrtula:1100.00
```

```
Response: SetPoint [mm]:38.00 Vyska [cm]:24.50 Vrtula:1100.00
```

```
Response: SetPoint [mm]:37.00 Vyska [cm]:24.50 Vrtula:1100.00
```


3.4.7. Kompilácia

Zdrojový kód aplikácie skompilujeme najjednoduchšie cez príkazový riadok Raspbianu pomocou prekladača gcc resp. g++. Pomocou príkazu cd sa musíme dostať do priečinka so zdrojovým kódom.

```
cd ./desktop/rgm
```

A spustíme nasledovný príkaz:

```
g++ GestureRecognition_Rpi.cpp -o gesture `pkg-config --libs opencv` -  
std=c++11 -lwiringPi
```

Môžeme si všimnúť že na miesto:

- GestureRecognition_Rpi.cpp dosadíme názov kompilovaného súboru
- Gesture je názov výstupného spustiteľného súboru, ktorý sa nám vytvorí

3.4.8. Spustenie

Ak prebehla kompilácia úspešne a bez chýb, mali by sme mať vytvorený nový súbor gesture. Spustíme ho nasledovným príkazom:

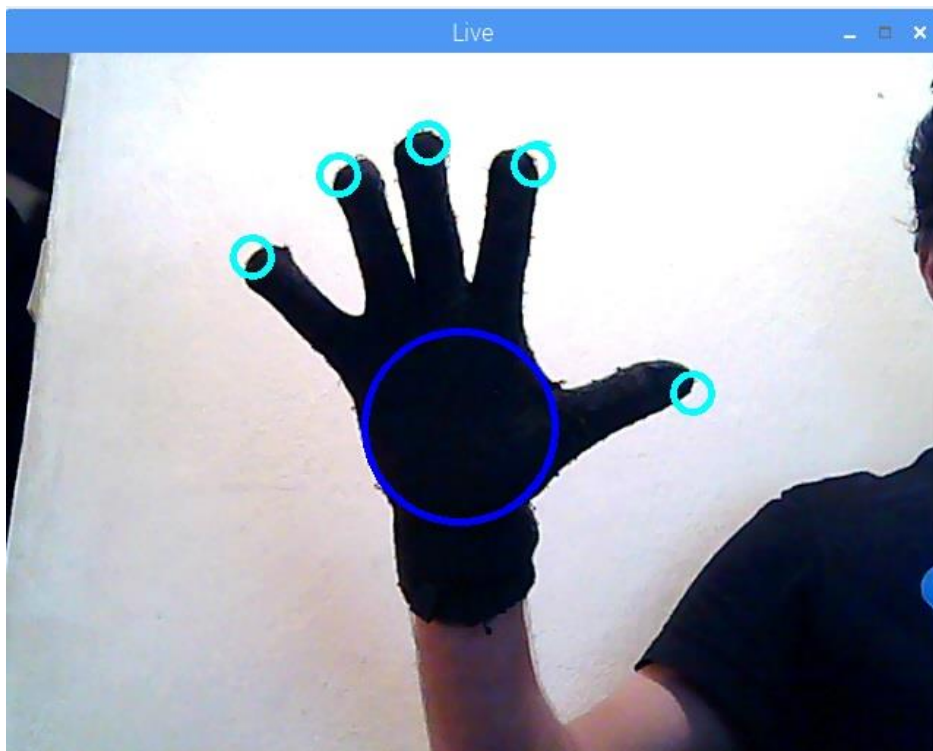
```
./gesture
```

Po spustení aplikácií môže trvať niekoľko sekúnd, kým sa zapne kamera a na obrazovke sa zobrazí jej výstup. Tak isto niekoľko sekúnd potrvá kým sa zapne UAV.

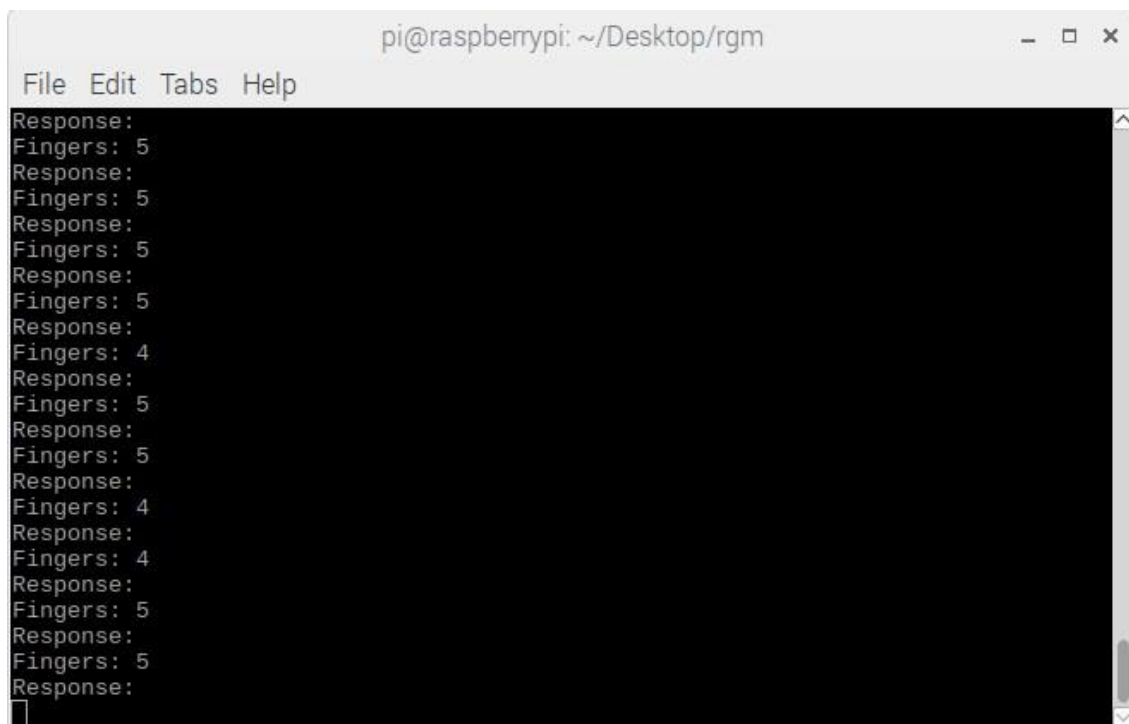
Ak všetko prebehlo správne, môžeme na kamere ukázať gesto. Odporúčame aby v pozadí bolo čo najmenej rušivých objektov tmavej farby a zároveň je vhodné si nasadiť rukavicu tmavej farby pre maximalizovanie úspešnosti rozpoznávania gesta. Na displeji by sme mali mať dve okná:

- Konzolu – zobrazuje informácie o počte detekovaných prstov a zároveň textové informácie od UAV.
- Okno s výstupom z kamery na ktorom graficky vidíme pozície prstov a dlane.

Teraz môžeme do kamery ukazovať gestá, z ktorých dostaneme ako výsledok počet zdvihnutých prstov. Rozpoznávané sú počty prstov 0,1,2,3,4 a 5. Na nasledujúcich obrázkoch si môžeme pozrieť ako to celé vyzerá.



Obrázok 15 – Náhľad grafického znázornenia aplikácie



Obrázok 16 – Náhľad konzolového výpisu aplikácie

5. Testovanie

Aplikáciu sme testovali na reálnom UAV v univerzitnom laboratóriu. Video z testovania sa nachádza v prílohe. Pre zjednodušenie som aplikáciu testoval na jednoduchom UAV . Arduino Uno prijímalo informáciu o počte rozpoznaných prstov a podľa toho posúvalo UAV tyč hore a dolu. Spať posielalo niektoré informácie od senzoro vo UAV ako napríklad výšku tyče.

Testovanie prebehlo úspešne ale je nutné dodať, že aplikácia je veľmi citlivá na pozadie za snímanou dlaňou. Ideálne je mať pozadie bez čo možno najviac rušivých objektov a ideálne mierne nižšími svetelnými podmienkami. Naša aplikácia bola testovaná s bielym pozadím a na ruke som mal čiernu rukavicu pre maximalizovanie úspešnosti detekcie.



Obrázok 17 – Testovanie v školskom laboratóriu

Na nasledujúcej tabuľke môžeme vidieť ako sa výška tyče na UAV menila v čase a so zmenou používateľovho gesta. Od času 1 do 8 bolo detekované gesto „1“ a od času 9-15 bolo detekované gesto „2“.

Čas	Výška [cm]	Počet prstov
1	19,11	1
2	19,12	1
3	19,14	1
4	19,18	1
5	20,81	1
6	21,65	1
7	23,13	1
8	24,04	1
9	26,32	2
10	25,22	2
11	24,13	2
12	24,13	2
13	23,88	2
14	22,52	2
15	21,89	2

Tabuľka 1 – Tabuľka meniacej sa výšky pri testovaní

Záver

V našej diplomovej práci sme si za cieľ zvolili vytvorenie jednoduchkej aplikácie, ktorá bude snímať používateľovu dlaň a rozpoznávať jej gestá. Aplikácia mala bežať na zariadení Raspberry PI 3 a mala byť schopná na základe rozpoznaného gesta odosielať signály na ovládanie bezpilotné lietadla (UAV).

Motiváciou pre písanie diplomovej práce bol najmä fakt, že sa zaujímam o oblasť informatiky zahrňujúcu spracovanie obrazu a detekciu jednotlivých objektov v obraze. Taktiež k tomu prispela aj túžba bližšie sa oboznámiť s operačným systémom Raspbian, ktorý je založený na linuxovej architektúre, zariadením Raspberry PI 3, ale aj s otvorenou multiplatformovou knižnicou OpenCv. Tá je v dnešnej dobe veľmi populárna a často používaná pretože poskytuje bohaté možnosti na spracovanie s obrazu, ale aj tvorbu aplikácií založených na strojovom učení.

Pokiaľ ide o splnenie cieľov práce, myslím že je možné prehlásiť že ciele diplomovej práce boli vo vysokej miere splnené. Dá sa povedať, že pri vypracovávaní našej práce sme sa potýkali s tromi hlavnými prioritami.

1. Prvým a hlavným bodom bolo naštudovať, premyslieť a implementovať technológiu rozpoznávania gest používateľovej ruky. To sa nám podarilo. S pomerne veľkou úspešnosťou vieme rozpoznať koľko prstov používateľ ukazuje v reálnom čase na kamere.
2. Druhou úlohou bolo zabezpečiť spustiteľnosť aplikácie na zariadení Raspberry PI 3. Keďže sme aplikáciu od začiatku vyvíjali priamo na tomto zariadení, tak s tým nebol problém a aplikácia na zariadení pomerne plynule beží
3. Tretia vec nad ktorou sme sa museli zamýšľať bola komunikácia našej aplikácie s UAV. To sme implementovali pomocou komunikácie s doskou Arduino UNO cez UART rozhranie.

Aplikáciu sme testovali v školskom laboratóriu a o úspešnosti jej testovania sa môžeme priamo presvedčiť na videu (Príloha B), ktoré sa nachádza na priloženom CD.

Poznatky získané v tejto diplomovej práci môžu nájsť široké využitie a môžu byť užitočné najmä v oblasti biometrických systémov pri rozpoznávaní fyziologických ...z obrazu. To je využívané najčastejšie v oblasti bezpečnosti, v hernom priemysle ale aj ako súčasť virtuálnej reality, ktorá je čoraz populárnejšia.

Použitá literatura

1. **Leap motion team.** LeapMotion.com. *LeapMotion*. [Online] LeapMotion, 2017. <https://www.leapmotion.com/>.
2. **Dočkal, Jakub.** *Drony zmenia svet, keď im to dovoľíme*. miesto neznámé : Stiahnut.sk, 2014.
3. **Raspberrypi.org.** *Raspberry PI*. [Online] Raspberry Foundation. <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>.
4. **Kolektív Autoru.** *Procesory*. miesto neznámé : Computer Press. ISBN: 80-7226-172-X.
5. **Estranky.sk.** *Estranky*. [Online] Estranky, 2016. <http://www.mozila.estranky.sk/clanky/webkamera.html>.
6. **Selecký, Matúš.** *Arduino*. miesto neznámé : Computer Press, 2016. ISBN 9788025148402.
7. **techopedia.com.** *Techopedia*. [Online] Techopedia, 2013. <https://www.techopedia.com/definition/26184/c-programming-language>.
8. **team, OpenCv.** OpenCv.org. *OpenCv*. [Online] OpenCv, 2016. <https://opencv.org/about.html>.
9. **Wikipedia.com.** *Wikipedia*. [Online] Google, 24. 2 2016. https://sk.wikipedia.org/wiki/Zabezpe%C4%8Den%C3%BD_pr%C3%ADstup_k_pr%C3%A4kazov%C3%A9mu_interpretov%C4%8Du.
10. **Microsoft team.** Understanding the Remote Desktop Protocol (RDP). *support.microsoft.com*. [Online] microsoft, 17. 4 2018. <https://support.microsoft.com/en-us/help/186607/understanding-the-remote-desktop-protocol-rdp>.
11. **Sériový port.** *it-slovník.cz*. [Online] it-slovník. <https://it-slovník.cz/pojem/seriovy-port>.
12. **Doxygen.** OpenCv.org. *docs.opencv.org*. [Online] opencv, 14. 5 2018. https://docs.opencv.org/3.4/d7/d9f/tutorial_linux_install.html.

Prílohy

Príloha A: Obsah priloženého CD nosiča

- \DP – Ovládanie UAV\Diplomová práca_Martin Molnár.pdf - Elektronická verzia tejto práce.
- \DP – Ovládanie UAV\video\Video z testovania.mp4 – Video z testovania aplikácie v školskom laboratóriu
- \DP – Ovládanie UAV\rgm\compile.txt - Príkaz na kompiláciu programu na operačnom systéme Raspbian
- \DP – Ovládanie UAV\rgm\konzola.txt - Výpis z konzoly počas testovania aplikácie v školskom laboratóriu
- \DP – Ovládanie UAV\rgm\GestureRecognition_Rpi.cpp - Zdrojový kód aplikácie v jazyku C++
- \DP – Ovládanie UAV\rgm\gesture.exe - Výsledný skompilovaný program spustiteľný na zariadeniach s ARM procesorom

Príloha B: Video z testovania aplikácie na reálnom UAV

Video z testovania aplikácie na reálnom UAV sa nachádza na priloženom CD.

Príloha C: Návod na inštaláciu OpenCv knižnice na Raspbian

Príloha C obsahuje podrobný postup ako na operačnom systéme Raspbian stiahnuť, skompilovať a nainštalovať knižnicu OpenCv. Nasledujúce kroky boli otestované na systémoch Ubuntu a Raspbian, ale mali by fungovať na všetkých distribúciach systémov Linux.

Balíčky potrebné na inštaláciu OpenCv

- GCC kompilátor verzie 4.4.x alebo vyššej
- CMake verzie 2.6 alebo vyššej
- Git repozitár
- GTK+ verzie 2.x alebo vyššej, spolu s (libgtk2.0-dev)
- pkg-config
- Python verzie 2.7 alebo vyššej
- Numpy verzie 1.5 alebo vyššej
- ffmpeg alebo libav development packages: libavcodec-dev, libavformat-dev, libswscale-dev
- libtbb2 libtbb-dev (nepovinné)
- libdc1394 2.x (nepovinné)
- libjpeg-dev, libpng-dev, libtiff-dev, libjasper-dev, libdc1394-22-dev (nepovinné)

Ak nemáte potrebné balíčky nainštalované, môžete ich doinštalovať nasledujúcimi príkazmi.

```
sudo apt-get install build-essential
```

```
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
```

```
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
```

Stiahnutie OpenCv zdrojových kódov

Na stiahnutie najaktuálnejšej verzie OpenCv môžete použiť stránku [sourceforge.net](https://sourceforge.net/projects/opencvlibrary/) alebo verejný Git repozitár.

Získanie najnovšej verzie OpenCv zo sourceforge

- Pôjdeme na stránku <https://sourceforge.net/projects/opencvlibrary/>
- Stiahneme zdrojový kód v tar archíve a rozbalíme ho do priečinku

Získanie najnovšej verzie OpenCv z Git repozitára

- Spustíme Git klienta a stiahneme repozitár <https://github.com/opencv/opencv>
- To môžeme docieľiť aj spustením nasledujúcich príkazov v príkazovom riadku

```
cd ~/< priečinok_kam_uložíme_repozitár >  
git clone https://github.com/opencv/opencv.git
```

Kompilácia OpenCV zdrojových súborov pomocou CMake, v príkazovom riadku

1. Vytvorte dočasný priečinok <cmake_binary_dir>, kam chcete aby kompilátor uložil skompilované projektové súbory
2. Presuňte sa do tohto priečinka pomocou príkazu `cd <cmake_binary_dir>`
3. Príkaz `cmake [<voliteľné parametre>] <cesta k OpenCv stiahnutým suborom >`

Napríklad:

```
cd ~/opencv  
mkdir release  
cd release  
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local  
..
```

4. Vstúpte do vytvoreného priečinka `cd <cmake_binary_dir>`
5. Spustíte príkaz `make` nasledovne

```
sudo make install
```

Poznámka:

Použite `cmake` príkaz bez medzier za `-D` ak predošlý postup nefungoval správne. (10)

```
cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local ..
```