

**UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH  
PRÍRODOVEDECKÁ FAKULTA**

**RIADENIE SOFTVÉROVÝCH PRODUKTOV POMOCOU  
HCI KOMPONENTOV**

**Diplomová práca**

**2019**

**Richard Staňa**

**UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH  
PRÍRODOVEDECKÁ FAKULTA**

**RIADENIE SOFTVÉROVÝCH PRODUKTOV POMOCOU  
HCI KOMPONENTOV**

**Diplomová práca**

Študijný program:

Informatika

Študijný odbor:

9.2.1. - informatika

Školiace pracovisko:

Ústav informatiky

Vedúci práce:

prof. RNDr. Gabriel S e m a n i š i n, PhD.

Konzultanti:

Ing. Viktor Michalčin, PhD., Matúš Kirchmeyer

**Košice 2019**

**Richard Staňa**

## Abstrakt

Práca sa zaoberá interakciou človeka s počítačom (HCI - Human computer interface). Jej cieľom je analyzovať existujúce možnosti HCI, navrhnúť vlastné HCI komponenty a pokúsiť sa ich pilotne implementovať.

Konkrétnie nás zaujíma situácia videokonferenčného hovoru, pri ktorej by sme chceli detegovať reč (VAD - Voice Activity Detection) a následne automaticky vypínať a zapínať mikrofón. Samotná detekcia reči zo zvuku (AVAD - Acoustic Voice Actividy Detection) často nie je presná, kvôli okolitému šumu. Pokúsime sa detegovať reč pomocou obrazu z videokamery (VVAD - Visual Voice Actividy Detection). Pomocou knižnice Dlib získavame z obrazu kamery pozíciu tváre a z nej získavame body na tvári spôsobom, ktorý je popísaný v práci [8]. Následne získavame informáciu o reči z pomeru výšky a šírky pier podobne ako to robia v práci [1]. Vytvorili sme dynamickú knižnicu v jazyku C++. Našu implementáciu sme nakoniec podrobili testovaniu na niekoľkých dvojiciach testovacích a trénovacích videí.

**Kľúčové slová:** *videokonferenčný a kolaboračný systém, detekcia reči, detekcia tváre, detekcia bodov na tvári, HCI, riadenie SW produktov.*

## **Abstract**

Here will be abstract.

**Keywords:** *voice aktivity detection, face detection, face landmarks detection, HCI, SW product control.*

# Obsah

<b>Zoznam použitých skratiek</b>	<b>6</b>
<b>Úvod</b>	<b>7</b>
<b>1 Teoretický prehľad</b>	<b>8</b>
1.1 Interakcia človek počítač . . . . .	8
1.2 Vývoj HCI . . . . .	8
1.2.1 Príkazový riadok . . . . .	8
1.2.2 Grafické používateľské rozhranie . . . . .	9
1.2.3 Prirodzené používateľské rozhranie . . . . .	9
<b>2 Riešenia detektie reči</b>	<b>11</b>
<b>3 Návrh riešenia</b>	<b>15</b>
3.1 Detekcia bodov na tvári - VVAD . . . . .	15
3.2 Existujúce implementácie . . . . .	17
3.2.1 Intel RealSense . . . . .	17
3.2.2 Knižnica OpenCV . . . . .	18
3.2.3 Knižnica Dlib . . . . .	19
3.2.4 Porovnanie knižníc OpenCV a Dlib . . . . .	19
3.3 Detekcia reči zo zvuku - AVAD . . . . .	22
3.4 VAD spojením AVAD a VVAD a vytvorenie knižnice . . . . .	23
<b>4 Implementácia</b>	<b>24</b>
4.1 Použitý hardware a software . . . . .	24
4.2 Implementácia VVAD . . . . .	24
4.2.1 Metóda Frame . . . . .	25
4.2.2 Metóda FrameForLearningThreshold . . . . .	30
4.3 Implementácia AVAD . . . . .	33

<b>5 Testovanie</b>	<b>34</b>
5.1 Metodika testovania . . . . .	34
5.2 Testovanie bodov na perách . . . . .	34
5.3 Testovanie natrénovaného modelu . . . . .	38
5.4 Testovanie videí . . . . .	39
5.5 Zhodnotenie výsledkov testovania . . . . .	40
<b>Záver</b>	<b>41</b>
<b>Zoznam použitej literatúry</b>	<b>42</b>
<b>Prílohy</b>	<b>44</b>

# Zoznam použitých skratiek

AVAD - Acoustic Voice Actividy Detection  
CLI - Command-Line Interface  
CNN - Convolutional neural network  
DCNN - Deep convolutional neural network  
DNN - Deep neural network  
EBGM - Elastic Bunch Graph Matching  
GMM - Gausian Mixture Model  
GUI - Graphical User Interface  
HCI - Human–Computer Interaction  
HoG - Histogram of oriented gradients  
SVM - Support-vector machine  
VAD - Voice Activity Detection  
VVAD - Visual Voice Actividy Detection

# Úvod

Predstavme si situáciu konferenčného hovoru. Jeden z účastníkov potrebuje niečo urobiť, no jeho aktivita by bola hlučná, čím by rušil ostatných účastníkov a teda si vypne mikrofón. Samozrejme, neskôr keď chce niečo povedať, nezapne mikrofón a ostatní ho nepočujú. Ako by sme sa mohli popísanej situácií vyhnúť? Čo tak zapínať a vypínať mikrofón automaticky podla toho, či človek sediaci pred kamerou rozpráva alebo nerozpráva. Jedným z riešení by mohla byť detekcia reči (Voice activity detection - VAD) pomocou mikrofónu (Acoustic Voice Actividy Detection - AVAD). Toto riešenie však nemusí byť najpresnejšie, napríklad by nemuselo detegovať tichý hlas alebo by detegovalo nechcený šum a podobne. Vhodným zlepšením by mohlo byť pridanie VAD pomocou obrazu webovej kamery (Visual Voice Actividy Detection - VVAD). VVAD by mohlo priniesť viacero vylepšení, napríklad: odstránenie detekcie nechcených zvukov a šumu.

V práci sa zaoberáme možnosťami detekcie tváre v obraze, detekciou bodov na tvári a detekciou reči všeobecne. Rozoberáme existujúce riešenia danej problematiky a snažíme sa navrhnúť a implementovať vlastné riešenia.

Cieľom práce je analyzovať existujúce prístupy k riadeniu SW produktov pomocou HCI komponentov. Následne navrhнемe vlastné HCI komponenty a pokúsime sa ich vhodným spôsobom ich implementovať. Nakoniec sa budeme snažiť pilotne implementovať vytvorené komponenty do existujúcich SW produktov.

V prvej kapitole sa zaoberáme pojmom Human-Computer Interface a jeho historiou. V druhej opisujeme existujúce riešenia detekcie reči. V tretej je analyzovaný návrh riešenia, možnosti VVAD a AVAD a ich existujúce implementácie. Detailný popis našej implementácie sa nachádza v kapitole 4. Posledná kapitola obsahuje výsledky testovania nami implementovaného riešenia VAD.

# **Teoretický prehľad**

## **Interakcia človek počítač**

HCI - Human-Computer Interaction (interakcia človek počítač), je medziodborová disciplína, ktorá skúma problematiku interakcie a komunikácie medzi človekom a počítačom. Vznikla na prelome 70. a 80. rokov dvadsiateho storočia, v dobe, keď začali vznikať prvé osobné počítače. HCI v sebe spája veľa odborov, ktoré na prvý pohľad nemajú nič spoločné. Ak sa však chceme zaoberať vytváraním používateľských rozhraní, kľúčové sú nasledujúce disciplíny: informatika, ergonómia, umenie, design, psychológia, kognitívna psychológia, lingvistika, sociológia, filozofia, antropológia, fyziológia, umelá inteligencia, inžinierstvo, kognitívna veda, etika, estetika a. i.

Informatika sa v oblasti HCI zameriava najmä na design a tvorbu informačných systémov a ich rozhraní tak, aby boli co najjednoduchšie a najintuitívnejšie pre špecifickú skupinu používateľov. HCI skúma aj vnímanie, správanie a informačné potreby koncového používateľa. Hlavným cieľom HCI je dosiahnuť lepšiu použiteľnosť a intuitívnosť informačných systémov aj pre menej odborných používateľov. [13]

## **Vývoj HCI**

Je zaujímavé sledovať, ako sa HCI vyvíja v priebehu času. Od primitívnych klávesníc sme sa postupne dostali cez počítačovú myš, dotykové obrazovky a podobne až ku hlasovému ovládaniu a rôznym gestám, ktoré nám zjednodušujú každodenný život. V tejto časti presnejšie popíšeme historický vývoj komunikácie medzi človekom a počítačom.

## **Príkazový riadok**

Od polovice 60. rokov sa CLI - Command-Line Interface (príkazový riadok) používa ako hlavný spôsob komunikácie človeka s počítačom. Používateľ zadáva príkazy pomocou klávesnice, výsledky vidí na monitore väčšinou v textovej podobe. Používanie CLI pokračuje v 70. a 80. rokoch systémoch OpenVMS, Unix a osobných počítačoch MS-

DOS, CP/M a Apple DOS. Pre skúseného používateľa má CLI veľa výhod, uvedieme niektoré z nich:

- rýchlosť a efektívnosť,
- možnosť použitie skriptov,
- história príkazov,
- nie je potrebná myš, stačí klávesnica, ...

Ale pre menej skúseného používateľa prináša veľa nevýhod, napríklad:

- prostredie je veľmi neintuitívne a striktné,
- nemožnosť použitia myši, ...

CLI bolo postupne nahradené grafickým prostredím, no v niektorých oblastiach sa stále používa. Najmä medzi používateľmi linuxu, na spravovanie serverov alebo pri programovaní v niektorých prostrediach.

## Grafické používateľské rozhranie

S príchodom väčšieho grafického výkonu vzniká GUI - Graphical User Interface (Grafické používateľské rozhranie). Okrem klávesnice sa používa počítačová myš a vzniká pracovná plocha, okná, ikony, rôzne gestá (dvojklik myši, drag and drop, ...), zlepšujú sa možnosti používania pre slabozrakých. Oproti CLI má GUI veľké výhody pre neskúseného používateľa:

- intuitívnosť,
- rýchlosť - v špecifických prípadoch (napr. presúvanie súborov), ...

Často ale môže nastať prípad, keď sú niektoré nastavenia príliš hlboko v systéme a pomocou GUI sa k nim nie je možné dostať. V dnešnej dobe je GUI najpoužívanejším typom HCI na osobných počítačoch.

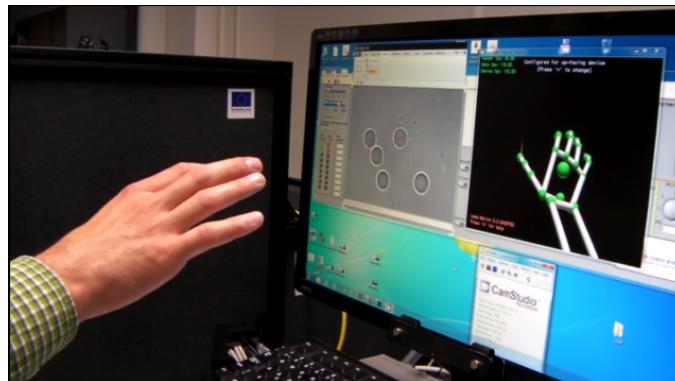
## Prirodzené používateľské rozhranie

S príchodom nových technológií (smartfónov, virtuálnej reality, ...) sa do popredia začína dostávať NUI - Natural User Interface (Prirodzené používateľské rozhranie). Pomocou NUI dokáže používateľ úplne prirodzene, priamo a intuitívne interagovať s počítačom. Príkladom NUI, s ktorým sa už asi každý stretol, je používanie gest

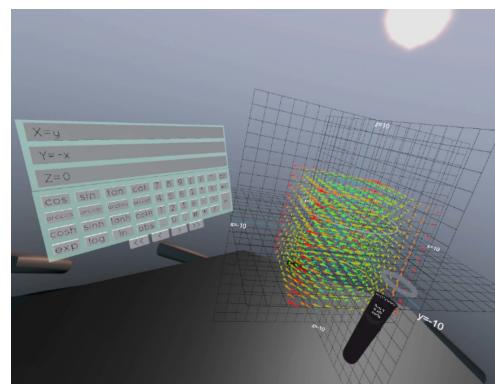
na dotykových obrazovkách. Ďalšie príklady NUI je možné vidieť na nasledujúcich obrázkoch 1, 2 a 3.



Obr. 1: Google Home - Inteligentný domáci asistent, s ktorým sa komunikuje pomocou hlasových príkazov. [2]



Obr. 2: Manipulácia objektov pomocou optickej pinzety, kontrolovaná pozíciou prstov. [20]

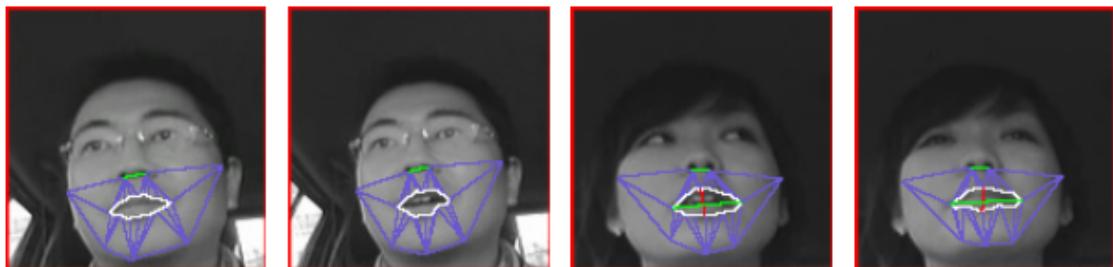


Obr. 3: Používanie aplikácie Calcflow vo virtuálnej realite. [12]

# Riešenia detekcie reči

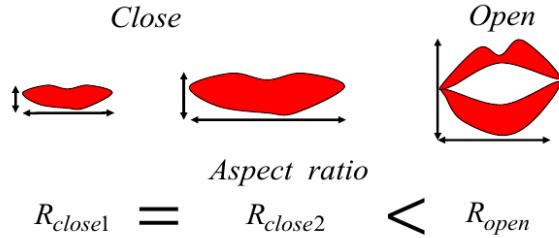
V nasledujúcich odsekoch ukážeme niekoľko existujúcich riešení. Zameriame sa na použité technológie, dosiahnuté výsledky a problémy, ktoré majú uvedené riešenia.

V [1] riešia VAD - Voice Activity Detection (detekciu reči) u šoféra v aute. V tomto prostredí je veľa nepriaznivých zvukov, ako je napríklad zvuk motora, rádio, reč spolucestujúcich, .... Detekcia reči môže byť pre šoféra veľmi užitočné, lebo je zaneprázdnéný riadením vozidla, no z dôvodov uvedených v predchádzajúcej vete je veľmi náročná. Reč detegujú zo zvuku pomocou Gausian mixture model (GMM). Túto detekciu kombinujú s VVAD - Voice Activity Detection (VAD pomocou videa). Kvôli odfiltrovaniu nepriaznivých odleskov a nedostatku svetla v noci používajú infračervenú kameru. Z sedo-tónového obrazu získavajú obrys pier, pomocou Elastic Bunch Graph Matching (EBGM). Ukážku získaného obrysu pier je možné vidieť na 4.



Obr. 4: Získavaný obrys pier pomocou EBGM. [1]

Z pier určujú pomer ich výšky a šírky. Získaný pomer má výhodu v nezávislosti od vzdialenosťi tváre od kamery. Obrázok 5 vysvetľuje prečo sa pomer mení, len keď človek rozpráva.



Obr. 5: Pomer výšky a šírky pier sa zmení len keď sa ústa otvárajú a zatvárajú. [1]

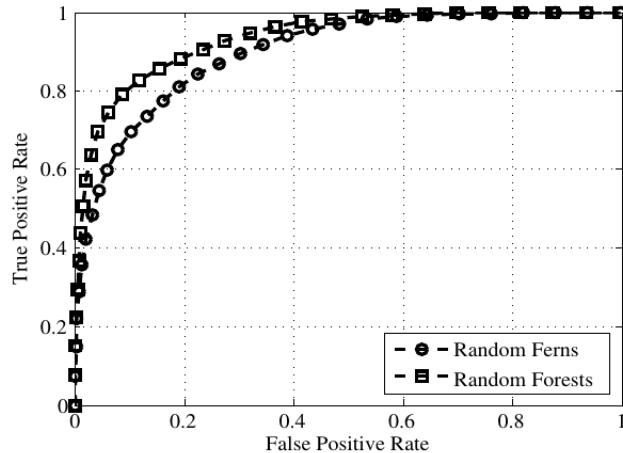
Navrhnutú metódu testovali tak, že šofér (raz muž a raz žena) prečítal 100 názvov japonských miest. Testovaná metóda priemerne zlepšuje detekciu reči o 40% oproti použitiu len AVAD - Audio Voice Activity Detection (VAD pomocou zvuku). Získane výsledky možno vidieť v tabuľke 1.

	Všetky zdetegovania reči	Správne zdetegovania reči	Úspešnosť	Presnosť
muž	106	100	100%	94,33%
žena	118	100	100%	84,75%

Tabuľka 1: Výsledky testovania metódy v práci [1]

Nevýhodou metódy je to, že funguje len pri pohľade spredu. V článku sa nepíše nič o rýchlosťi ich riešenia.

V [21] sa zaobrajú detegovaním reči z jednoduchej webovej kamery. Najprv orežú snímky videa na oblasť pier. Snímky prevedú do šedotónovej oblasti, vyrežú 200 náhodných oblastí a urobia rozdiel všetkých snímok a prvého. Takto vedia modelovať zmeny vo vzhľade podľa rozdielov v čase. Z každého rozdielu počítajú štatistické koeficienty priemer, smerodajnú odchýlku a priemer nad prvou deriváciou. Aplikovaním popísaného postupu vytvorili tréningovú množinu o veľkosti 130 000. Následne natrénovali klasifikátor Random Forest s veľkosťou 20 stromov a maximálnou hĺbkou 10. Random Forest porovnávali s klasifikátorom Random Ferns. Random Ferns dosahovali pri rôznych nastaveniach parametrom stále horšie výsledky ako Random Forests. Porovnanie týchto dvoch metód je na obr. 6.



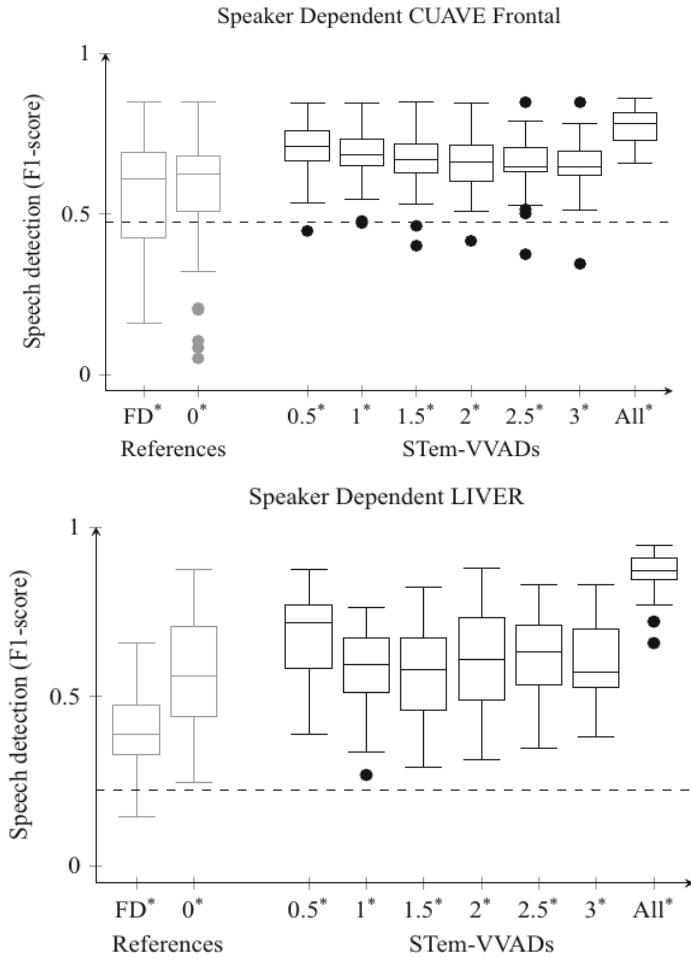
Obr. 6: Porovnanie klasifikátorov Random Forest a Random Ferns. [21]

Podľa článku navrhnutá metóda používa pohľad na tvár spredú a je použiteľná v reálnom čase (30 fps), kvôli rýchlosťi výpočtu Random Forest.

V článku [7] popisujú a testujú metódu VVAD založenú na časopriestorových Gáborových filtroch (angl. Spatiotemporal Gabor filters), ktorá podľa autorov nebola nikdy predtým na VVAD použitá. Používajú dve dátové sady: CUAVE - obsahuje nahrávky reči pri pohľade spredú aj z profilu a LIVER - obsahuje nahrávky vyslovovania holandského slova „liver“ pri pohľade spredú. Ich metóda sa skladá z 2 fáz:

- fáza predspracovania - aplikovanie časopriestorových Gáborových filtrov na zistenie energií v konkrétnych rýchlosťach (jeden z parametrov časopriestorových Gáborových filtrov),
- agregačná a klasifikačná fáza vytvárajúca sumáciu a klasifikátor, na priradzovanie agregovaných energetických hodnôt do binárnych tried (SPEECH a NON-SPEECH).

Autormi navrhnutú metódu porovnávajú s 2 referenčnými metódami - metódou založenou na rozdieloch snímok a metódou založenou na štandardných Gabor filters. Ich metóda bola v skoro všetkých prípadoch lepšia ako referenčné metódy. Niektoré porovnania je možné vidieť na obr. 7.



Obr. 7: Porovnanie referenčných metód založených na rozdieloch snímok (FD\*) a štandardných Gabor filters (0\*) s metódou založenou na časopriestorových Gáborových filtroch s rôznymi parametrami rýchlosťi (0,5\*, 1\*, 1,5\*, 2\*, 2,5\*, 3\*, All\*). [7]

Metóda z článku [7] funguje pri pohľade spredu aj z profilu. V článku sa nepíše nič o rýchlosti ich riešenia.

# Návrh riešenia

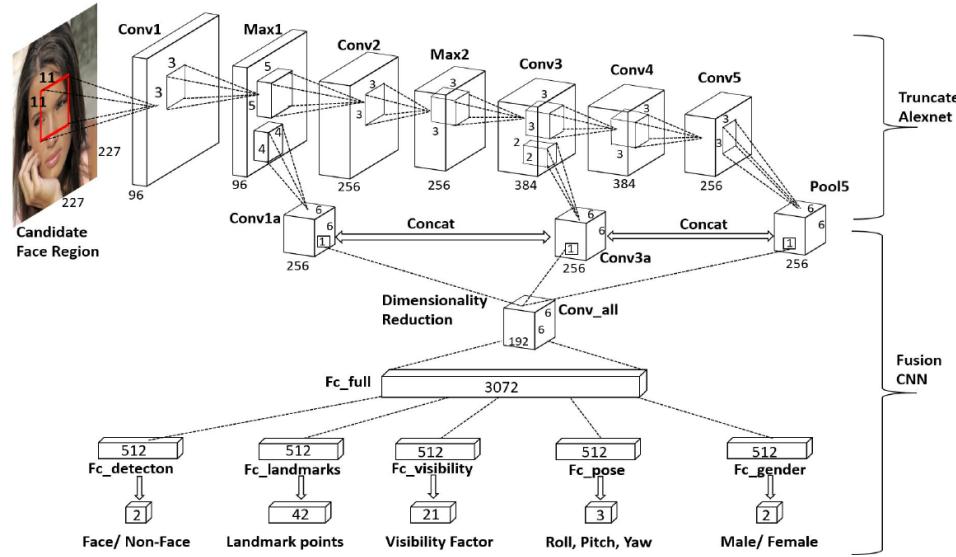
Po preštudovaní uvedených článkov, sa ako najpoužiteľnejšie riešenie pre VVAD javí sledovať zmeny pohybu pier popísanú v článku [1]. V článku sa nepíše nič o rýchlosti nimi vytvoreného riešenia EBGM na detekciu pier v obraze a riešenie asi nebude fungovať v reálnom čase. Naše riešenie bude musieť v reálnom čase fungovať, keďže zapínanie a vypínanie mikrofónu pri videohovore si to vyžaduje. V nasledujúcej časti opíšeme články zaobrajúce sa detekciou bodov na tvári.

## Detekcia bodov na tvári - VVAD

V článku [15] prezentujú algoritmus na simultánnu detekciu tvári, bodov na tvári, pozíciu (otočenie) tváre (hlavy) a rozoznávanie pohlavia s použitím DCNN. Uvádzajú dve verzie ich algoritmu nazývaného HyperFace:

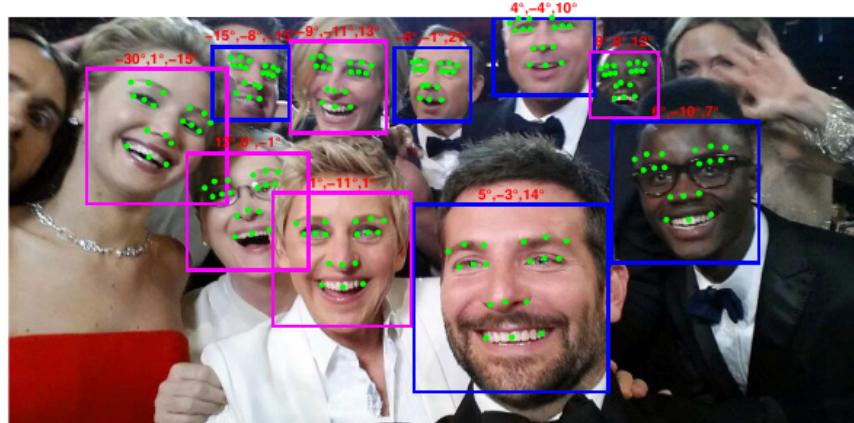
- HyperFace-ResNet, ktorý je postavený na modeli ResNet-101 a prináša značné zlepšenie výkonu algoritmu,
- Fast-HyperFace, ktorý používa rýchlejší detektor tvári na zrýchlenie algoritmu.

Na obr. 8 je architektúra siete HyperFace.



Obr. 8: Architektúra DCNN HyperFace. [15]

Kvôli testovaniu na rôznych dátových sadách trénovali siete pre rôzne počty bodov (21, 68, ...) na tvári. Ukážka výsledkov algoritmu je na obr. 9.



Obr. 9: Algoritmus simultánne deteguje tváre, body na tvári (zelené body), pohľavie (modrý štvorec - muž, ružový štvorec - žena) a pozíciu tváre (červené čísla nad štvorcami - priekýný sklon, pozdĺžny sklon a zatočenie). [15]

Práca [8] popisuje detekciu bodov na tvári pomocou súboru regresných stromov v reálnom čase. Na obr. 10 sú výsledky algoritmu na testovacej dátovej sade.



Obr. 10: Testovacie výsledky algoritmu používajúceho náhodné regresné stromy na nájdenie 194 bodov na tvári. [8]

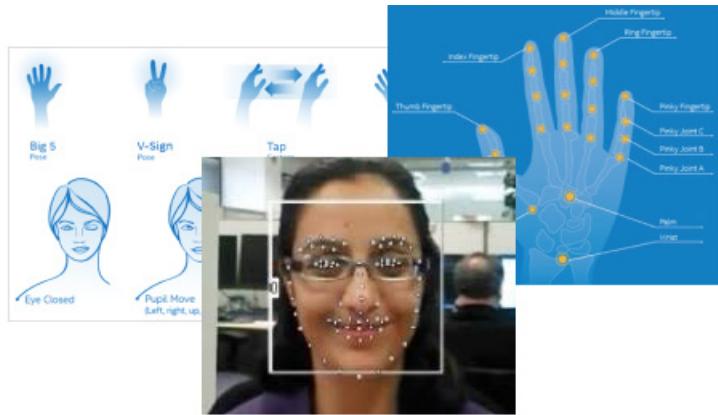
Pre náš problém sa javí riešenie z článku [8] ako lepšie, keďže už názov článku hovorí o jeho rýchlosťi (One millisecond face alignment with an ensemble of regression trees). Ďalšou výhodou tohto riešenia je to, že je implementované v knižniciach Dlib a OpenCV.

## Existujúce implementácie

V tejto podkapitole popíšeme existujúce knižnice (implementácie), ktoré sa zaobrajú detekciou tváre a bodov na nej.

### Intel RealSense

V roku 2018 predstavil Intel prvé hlíbkové kamery RealSense. K týmto kamerám vydal SDK [5] pre operačný systém Windows. Toto SDK dokázalo v obmedzenej miere pracovať aj s bežnou webovou kamerou. Dostupná bola pre nás dôležitá metóda detektie bodov na tvári. Ukážka funkcionality SDK je na obr. 11.



Obr. 11: Prvé SDK k Intel RealSense dokázalo sledovať ruku a prsty, analyzovať tvár, rozpoznávať reč a. i. [5]

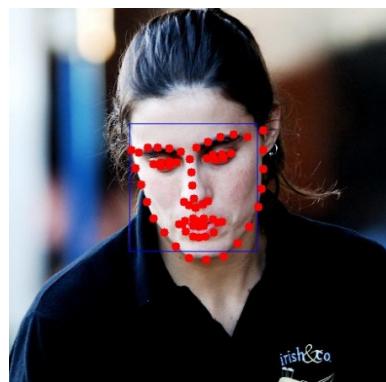
Existujú 2 dôvody, prečo toto riešenie nie je pre nás vhodné:

- podporovaný bol len operačný systém Windows - naše riešenie má byť multiplatformové,
- vývoj SDK bol zastavený.

Toto SDK bolo nahradené novým multiplatformovým SDK [6], ktoré už ale nevie pracovať s bežnou webovou kamerou.

## Knižnica OpenCV

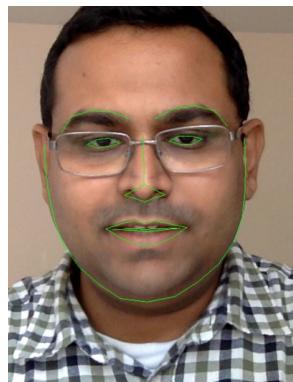
OpenCV [14] (Open Source Computer Vision Library) je multiplatformová knižnica zameraná na počítačové videnie. V knižnici OpenCV je implementovaná detekcia tvári a aj detekcia bodov na tvári z článku [8]. Prečo nepoužijeme knižnicu OpenCV si popíšeme v kapitole 3.2.4. Ukážku detektie je možné vidieť na obr. 12.



Obr. 12: Ukážka výsledku detekcie bodov na tvári pomocou knižnice OpenCV. [14]

## Knižnica Dlib

Dlib[9] je moderná multiplatformová knižnica obsahujúca nástroje pre strojové učenie a vytváranie komplexného softvéru v jazyku C++. Ako sme spomínali v predchádzajúcej podsekcii v knižnici Dlib je implementovaná detekcia bodov na tvári z článku [8]. Ukážka detektie bodov na tvári pomocou knižnice Dlib je na obr. 13.



Obr. 13: Detekcia detektie 68 bodov na tvári pomocou knižnice Dlib. [10]

## Porovnanie knižníc OpenCV a Dlib

V oboch knižniciach je implementovaná detekcia bodov na tvári z článku [8]. Ako problém ostáva nájdenie tváre v obraze. Obe knižnice ponúkajú viacero metód, ktoré tento problém riešia. Na stránke [3] porovnávajú metódy detektie tváre v knižniciach OpenCV a Dlib. Zameriavajú sa na 4 nasledujúce metódy:

- detekcia tváre pomocou Haar Cascade - OpenCV,
- detekcia tváre pomocou DNN - OpenCV,
- detekcia tváre pomocou HoG - Dlib,
- detekcia tváre pomocou CNN - Dlib.

Popíšeme výhody a nevýhody jednotlivých metod.

Detekcia tváre pomocou **Haar Cascade - OpenCV** bola špičkovou od roku 2001, kedy bola predstavená výskumníkmi Violom a Jonesom.

Výhody:

- funguje takmer v reálnom čase na CPU,

- jednoduchá architektúra,
- deteguje tváre rôznych veľkostí.

Nevýhody:

- deteguje veľa objektov, ktoré nie sú tvárami,
- nefunguje na tvárách, ktoré nie sú pri pohľade spredu,
- nefunguje ani pri čiastočnom zakrytí tváre.

Detekcia tváre pomocou **DNN** je v OpenCV implementovaná od verzie 3.3.

Výhody:

- najpresnejšia zo štyroch uvedených metód,
- funguje v reálnom čase na CPU,
- rozpozná rôzne otočené tváre,
- funguje aj pri značnom zakrytí tváre,
- deteguje tváre rôznych veľkostí.

Nevýhody:

- žiadne, až na tú, že nasledujúca metóda je rýchlejsia.

Detekcia tváre pomocou **HoG** je široko používaný model v knižnici Dlib založený na HoG a SVM.

Výhody:

- najrýchlejšia zo štyroch uvedených metód na CPU,
- funguje veľmi dobre pri pohľade na tvár spredu a mierne zboka,
- jednoduchý a nenáročný model v porovnaní s ostatnými,
- funguje pri čiastočnom zakrytí tváre.

Nevýhody:

- hlavná nevýhoda je, že metóda nedeteguje malé tváre (cca. 80x80 pixelov),

- box ohraničenia tváre často vynecháva čelo a bradu,
- nefunguje dobre pri značnom zakrytí tváre,
- nefunguje pre pohľad zboka a pre pohľad hore a dole.

Detekcia tváre pomocou **CNN** v knižnici Dlib používa Maximum-Margin Object Detector s CNN.

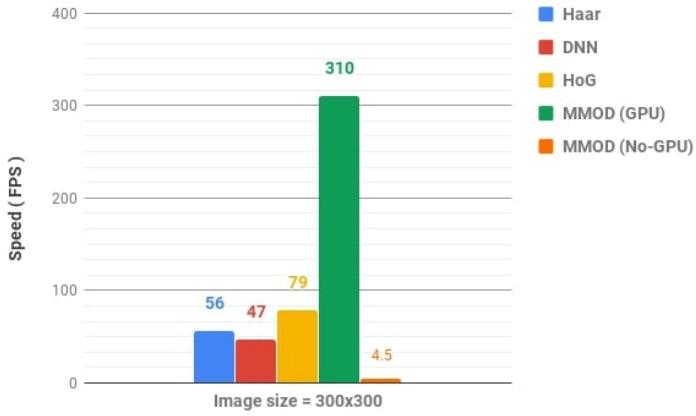
Výhody:

- funguje pre rôzne orientácie tváre,
- metóda je robustná na zakrytie tváre,
- funguje veľmi rýchlo na GPU.

Nevýhody:

- metóda je veľmi pomalá na CPU,
- metóda natrénovaná na detekciu tvári väčších ako 80x80 pixelov,
- box ohraničenia tváre je ešte menší ako v predchádzajúcom prípade.

Pre naše potreby je najvhodnejšia metóda detektie tváre pomocou HoG implementovaná v knižnici Dlib. Táto metóda nie je najpresnejšia, ale keďže chceme rozpoznávať reč pri videohovore, môžeme predpokladať, že vo väčšine prípadov sa osoba pred kamerou bude pozerať priamo na kameru (monitor) a tvár nebude mať ničím zakrytú. To, že metóda má problém s malými tvárami tiež pre nás nie je problém, lebo pri videohovore predpokladáme, že tvár nebude d'aleko od kamery. Zo spomínaných metód je jednoduchá, nenáročná a na CPU najrýchlejšia, čo je veľmi dobré, lebo kladieme veľký dôraz na to, aby naše riešenie fungovalo v reálnom čase. Porovnanie rýchlosťi jednotlivých metód je uvedené na obr. 14.

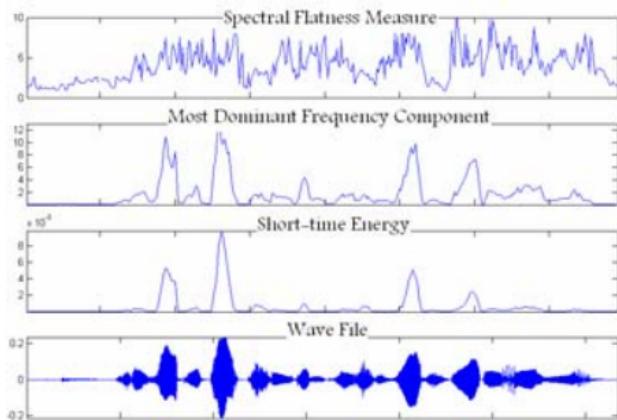


Obr. 14: Porovnanie rýchlosťí štyroch popísaných metód. [3]

## Detekcia reči zo zvuku - AVAD

Detekcia reči z videa nemusí byť pre naše potreby dostatočná. V nasledujúcej časti si popíšeme prácu zaoberajúcu sa detekciou reči zo zvuku.

V práci [11] predstavujú takmer ideálny AVAD algoritmus, ktorý je ľahký na implementáciu a robustný vzhľadom na šum. Pri detektii využívajú 3 rozdielne vlastnosti pre každú zvukovú snímku: energiu, spektrálnu rovinnosť (Spectral Flatness) a naj-dominantnejšiu frekvenčnú zložku. Pre každú z týchto vlastností sa zvolí prah a pre každú zvukovú snímku sa rátajú tieto 3 vlastnosti. Ak hodnota ktorejkoľvek vlastnosti bude väčšia ako prah prehlási sa aktuálna zvuková snímka za snímku s rečou. Prahy sa dynamicky menia počas behu algoritmu vzhľadom na predchádzajúce zvukové snímky. Ukážka vypočítaných vlastností na zvukovom súbore pomocou algoritmu z [11] je na obr. 15.



Obr. 15: Ukážka vypočítaných vlastností na zvukovom súbore bez šumu. [11]

## VAD spojením AVAD a VVAD a vytvorenie knižnice

Výstupom tejto diplomovej práce by mal program schopný zistiť, či používateľ počas videohovoru rozpráva alebo nerozpráva. Na to chceme použiť kombináciu metód VVAD a AVAD. Kombinácia metód je dôležitá v rôznych prípadoch. AVAD je dôležitá keď používateľ nemá webovú kameru, nie je ho na obraze vidieť, v miestnosti je zlé svetlo a. i. VVAD je dôležitá v prípade nemožnosti detektie zo zvuku, napríklad kvôli šumu alebo rušivému rozprávaniu inej osoby.

Výstup z práce by mal mať formu dynamickej multiplatformovej C++ knižnice. V knižnici by mala byť implementovaná metóda, ktorej vstupom by mali byť dve polia. Jedno pole s video snímkou a druhé so zvukovou snímkou. Výstup metódy by mal záležať od vstupných polí nasledovne:

- ak sú obe polia nenulové, metóda by mala vrátiť či bola detegovaná reč, či bola detegovaná zo zvuku alebo videa, pozíciu tváre ak bola detegovaná, a. i.,
- ak je pole so zvukovou snímkou nulové a s video snímkou nenulové, či bola detegovaná reč z videa, pozíciu tváre ak bola detegovaná, a. i.,
- ak je pole so zvukovou snímkou nenulové a s video snímkou nulové, či bola detegovaná reč zo zvuku, a. i.

Takýmto spôsobom môže byť využitie knižnice väčšie. Napríklad v prípade nedetegovania tváre sa môže znížiť kvalita prenášaného obrazu pri videohovore, čím sa zníži záťaž procesora aj sietovej linky.

# Implementácia

V tejto kapitole popíšeme implementáciu nami navrhnutého riešenia VAD v C++ knižnici.

## Použitý hardware a software

Implementácia a testovanie prebiehali na notebooku Asus Zenbook UX305FA (Intel Core M 5Y10, 8GB RAM) a na počítači HP Z240 (Intel Xeon E3-1245, 16GB ram). Pre testovanie sme používali webovú kameru LifeCam Cinema a mobilný telefón Xiaomi Pocophone F1. Vyvíjali sme jazyku C++, ktorý je najvhodnejší pre prácu s videom. Používali sme knižnice OpenCV, Dlib, SFML. Knižnica Dlib je závislá na knižnici OpenCV, ktorú sme používali vo verzií 3.4.5. Ako vývojárske prostredie bol zvolený CLion a buildovali sme pomocou Cmake. Kvôli zvýšeniu rýchlosťi komplujeme naprogramované zdrojové súbory so zapnutou optimalizáciou AVX, ktorá je podporovaná na procesoroch od roku 2011.

## Implementácia VVAD

Vytvorili sme dynamickú knižnicu s názvom VVAD. V hlavičkovom súbore `VVAD.h` sa nachádzajú definície tried, metód a premenných definovaných v súbore `VVAD.cpp`. V hlavičkovom súbore sú definované dve triedy: `VVAD` a `Output`. Trieda `VVAD` je hlavnou triedou, ktorá obsahuje verejne metódy:

- `Frame` - hlavná metóda, ktorej vstupom je video snímka a výstupom je objekt triedy `Output`.
- `FrameForLearningThreshold` - metóda sa volá na začiatku používania knižnice na určenie prahu. Vstupom metódy je video snímka a výsledným výstupom je prah, pomocou ktorého metóda `Frame` vyhodnocuje, či vo videosekvencii nastala reč alebo nie.

- `SaveThresholdToFile` - metóda uloží prah do súboru určeného reťazcom, ktorý dostane na vstup.
- `LoadThresholdFromFile` - metóda načíta prah zo súboru určeného reťazcom, ktorý dostane na vstup.
- `getThreshold` - vráti hodnotu prahu.
- `setThreshold` - nastaví hodnotu prahu.
- `isCalibrated` - vráti pravdivostnú hodnotu, ktorá hovorí, či prah bol nastavený.
- `getCalibrated` - nastaví pravdivostnú hodnotu, ktorá hovorí, či prah bol nastavený.

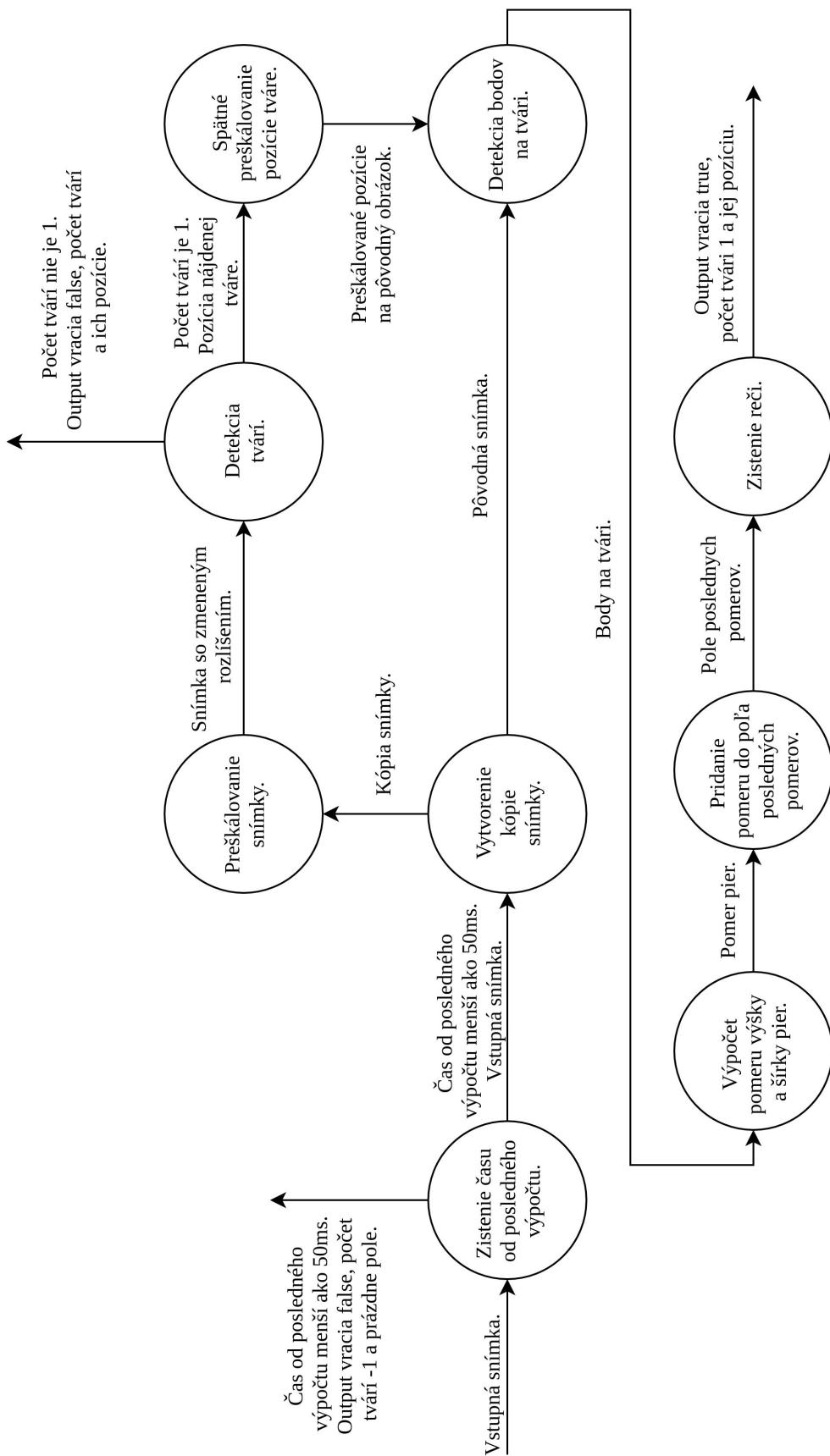
Trieda `Output` sa používa ako výstup metóda `Frame`. Trieda obsahuje nasledujúce privátne premenné, ktoré sú prislúchajúcimi metódami dostupné na čítanie:

- `_talking` - pravdivostná premenná, ktorá ak platí, tak reč bola detegovaná, inak reč detegovaná nebola.
- `_count_of_faces` - počet nájdených tvári.
- `_faces_positions` - pozície nájdených tvári na snímke, ktorý dostala metóda `Frame` na vstupe.

V nasledujúcej časti popíšeme fungovanie niektorých metód detailne.

## Metóda `Frame`

Zo snímky, ktorú dostane metóda `Frame` na vstupe sa vytvorí kópia, ktorá sa predspracuje. Predspracovanie spočíva v zmene rozlíšenia tak, aby šírka snímky bola 400 pixelov (toto predspracovanie budeme nazývať preškálovanie). Rozlíšenie bolo zvolené tak, aby nasledujúca detekcia tváre v snímke bola dostatočne rýchla aj na menej výkonných procesoroch. Detekcia tváre je presnejšie popísaná v časti 4.2.1.1. Výstupom z detektie tváre je pole pozícii nájdených tvári v kopijí pôvodnej snímky. Následne, ak sa nenašla práve jedna tvár, tak sa daná snímka preskakuje. Ak sa našla práve jedna tvár, prebieha detekcia bodov na tvári v pôvodnej snímke. Detekcia bodov na tvári je detailne popísaná v časti 4.2.1.2. Nájdené body na tvári spracúva metóda `ComputeDifferenceBetweenRatios`, ktorá je popísaná v časti 4.2.1.3. Nakoniec už len metóda `CheckSpeechInLastFrames`, vysvetlená v časti 4.2.1.4, skontroluje, či nastala reč a podľa toho sa vytvorí objekt triedy `Output`, ktorý sa dáva na výstup. Detailný popis metódy `Frame` je na obr. 16.



Obr. 16: Graf priebehu metódy **Frame**. Vo vrcholoch sú akcie, ktoré sa vykonávajú, šípky znázorňujú výstupy z akcií.

Pri používaní knižnice je možné volať metódu `Frame` ľubovoľne veľa krát za sekundu, minimálne však 20 krát za sekundu, inak nebude fungovanie korektné. Metóda `Frame` si zapamätá aktuálny čas, kedy spracovala snímku na vstupe a v prípade, že metóda je volaná viac ako 20 krát za sekundu, najprv skontroluje čas od posledného spracovania snímky. Ak je menší ako 50ms tak snímku zahadzuje, inak prebieha spracovanie.

## Detekcia tváre

Ako sme popísali v časti 3.2.4, na detekciu tvári sme použili knižnicu Dlib, konkrétnie detekciu tvári pomocou HoG. Táto detekcia je najpomalšia časť celého algoritmu, no aj napriek tomu funguje dostatočne rýchlo na oboch počítačoch, ktoré sme pri implementácii používali. Dôvodom je to, že detekcia tváre je spúšťaná na preškálovanej snímke. Pozície tvári sa po nájdení preškálujú tak, aby boli na správnych miestach v pôvodnej snímke a uložia sa do vektora.

## Detekcia bodov na tvári

Pre detekciu bodov na tvári budeme potrebovať natrénovaný model. V príkladových súboroch knižnice sa nachádzajú aj 2 predtrénované modely, ktoré detegujú 68 (obr. 17) alebo 5 bodov na tvári.



Obr. 17: Ukážka detekcie modelu so 68 bodmi v reálnom čase pomocou knižnice Dlib.

Model so 68 bodmi má približne 95MiB, čo je príliš veľa a model s 5 bodmi zas neobsahuje body na perách. Rozhodli sme sa teda vytvoriť vlastný model, ktorý by mal menšiu veľkosť a bol dostatočne presný pre naše potreby. Pre trénovanie modelu budeme potrebovať fotografie osôb a k nim súbor s popisom umiestnenia bodov na tvárich (súbory sú štandardne formátu xml, budeme ich preto skrátene nazývať xml

súbory). V príkladových súboroch sa nachádzajú fotografie aj xml súbory pomocou ktorých je možné trénovať model. Na trénovanie sú určené 4 fotografie s celkovo 18 tvárami. Na testovanie je určených 5 fotografií s celkovo 25 tvárami. Pomocou uvedených súborov však nie je možné natrénovať dostatočne presný model. Na stránke knižnice Dlib sa dá nájsť dátová sada `ibug_300W_large_face_landmark_dataset`, čo je vlastne dátová sada 300-W [4] s pridanými zrkadlovo otočenými obrázkami (pri použití dátovej sady 300-W [4] žiadajú citovať [16], [17] a [18]) Dátová sada 300-W [4] obsahuje fotografie z dátových sád `afw`, `helen`, `ibug` a `lfpw`. Dátová sada samozrejme obsahovala aj testovaci a trénovací xml súbor s označenými umiestneniami 68 bodov na tvári, pomocou ktorých je možné natrénovať presnejší model. Trénovací súbor obsahoval 6 666 tvári a testovací obsahoval 1 008 tvári. Knižnica Dlib obsahuje nástroj na vytváranie a úpravu xml súborov s informáciami o bodech na tváraх na fotografiách a aj nástroj na trénovanie a testovanie modelu na základe fotografií a xml súborov. Pomocou nástroja na prácu s xml súbormi z knižnice Dlib a editora Sublime Text sme upravili počet bodov na tvári v xml súboroch. Nástroj na trénovanie má možnosť nastaviť parametre trénovania. Niektoré z nich bližšie popíšeme v časti 5.3, kde je popísaný návrh a testovanie modelov.

Model, ktorý natrénujeme sa uloží do súboru a následne sa načíta pri volaní konštruktora triedy `VVAD`. Vytvorí sa objekt triedy `shape_predictor` z knižnice Dlib, ktorého metóda na detekciu bodov na tvári sa volá po zdetegovaní tváre. Metóda dostane na vstup pôvodnú snímku a pozíciu zdetegovanej tváre a na výstup vráti objekt triedy `full_object_detection`, ktorý obsahuje pozíciu tváre a vektor bodov na tvári.

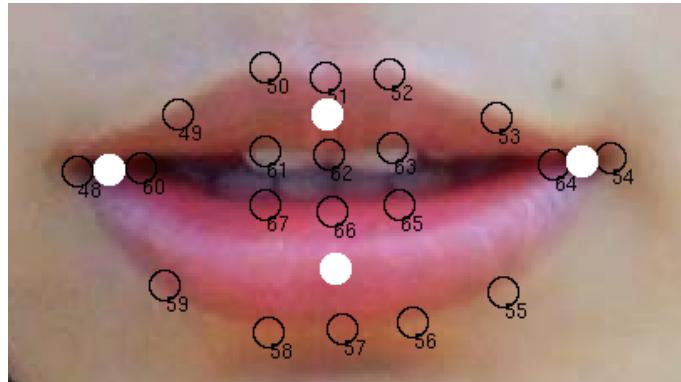
## Pomer pier

V metóde `ComputeDifferenceBetweenRatios` je implementovaná časť práce [1]. Z dvadsiatich bodov na perách sa najprv zrátajú pozície bodov z ktorých sa bude rátat pomer výšky a šírky pier:

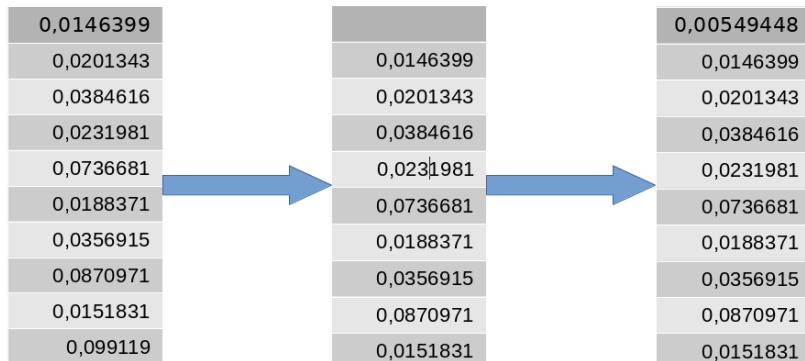
- horný - zoberú sa 4 body v hornej časti pery, z ktorých sa spraví priemer,
- dolný - zoberú sa 4 body v dolnej časti pery, z ktorých sa spraví priemer,
- pravý - zoberú sa 2 body v hornej časti pery, z ktorých sa spraví priemer,
- ľavý - zoberú sa 2 body v hornej časti pery, z ktorých sa spraví priemer.

Ukážku popísaných bodov je možné vidieť na obr. 18. Z pravého a ľavého bodu sa vyráta šírka pier a z horného a dolného bodu výška. Následne sa vypočíta pomer výšky a šírky a zráta sa absolútна hodnota rozdielu aktuálneho pomeru a pomeru z predchádzajúcej snímky. Pomocou metódy `ShiftAndAddRatioDifference` sa

vypočítaná absolútна hodnota uloží do 10 prvkového posuvného poľa. Ako toto desať prvkové pole funguje, je vysvetlené na obr. 19.



Obr. 18: Ukážka bodov na perách. Horný biely bod je priemer čiernych bodov 50, 52, 61, 63, biely dolný 67, 65, 58, 56, biely pravý 64 a 54 a biely ľavý 48 a 60.



Obr. 19: Vysvetlenie posuvného poľa. Pri zrátaní absolútnej hodnoty rozdielov pomerov sa posunú hodnoty v poli a nová hodnota sa vloží na prázdro miesto. Takto si pamäťame posledných desať rozdielov pomerov.

## Kontrola reči

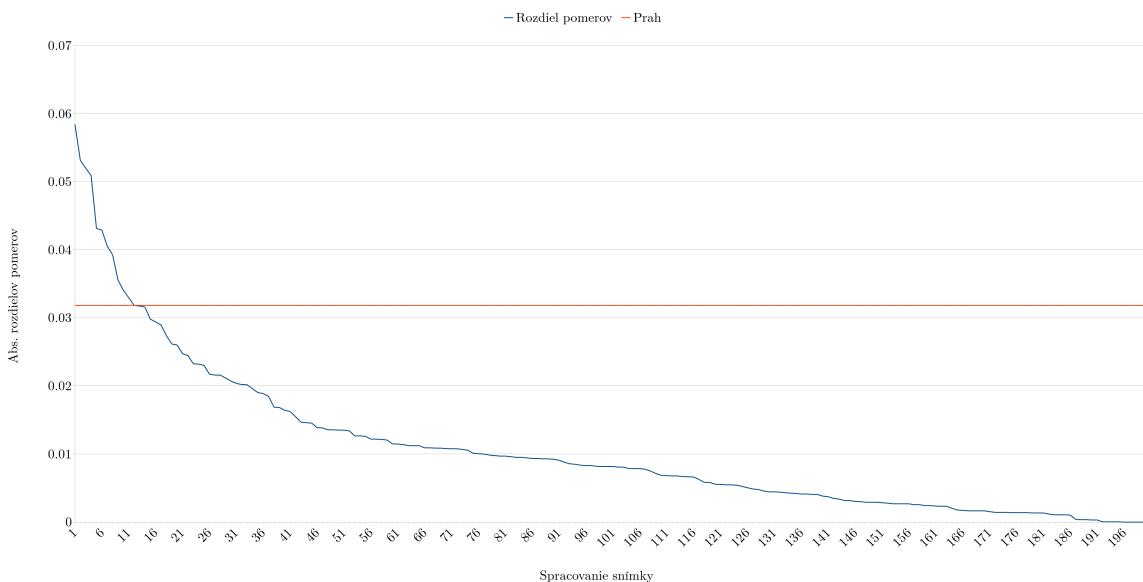
Po pridaní hodnoty do posuvného poľa sa v metode **Frame** zavolá metóda **CheckSpeechInLastFrames**. V nej sa skontroluje, či sa v poli nachádza hodnota väčšia ako určený prah. Ak áno metóda **Frame** dá na výstup objekt tiedy **Output** s parametrami pravda, počtom tvári 1 a s pozíciou danej tváre. Ak nie metóda **Frame** dá na výstup objekt tiedy **Output** s parametrami nepravda, počtom tvári 1 a s pozíciou danej tváre.

To, že sa kontroluje hodnota z posledných 10 snímok znamená, že program kontroluje aktivitu úst pol sekundu do minulosti. V prípade reči sa vyskytne hodnota väčšia ako prah v poli hned' na prvom mieste a teda program okamžite korektne zareaguje.

na reč. Za problém by mohlo byť považované to, že hodnota väčšia ako prah sa bude v posuvnom poli vyskytovať ešte pol sekundy. To bude spôsobovať, že program bude vracať zdetegovanú reč, aj keď už reč nebude prebiehať. Výsledný efekt je ale opačný. Ak by sa reč vyhodnocovala iba z aktuálnej snímky, respektíve rozdielu dvoch snímok, výstup programu by priveľmi skákal medzi zdetegovanou a nezdetegovanou rečou. Naše riešenie teda spôsobuje, že výstup programu je hladší a v prípade reči hned' korektne zareaguje.

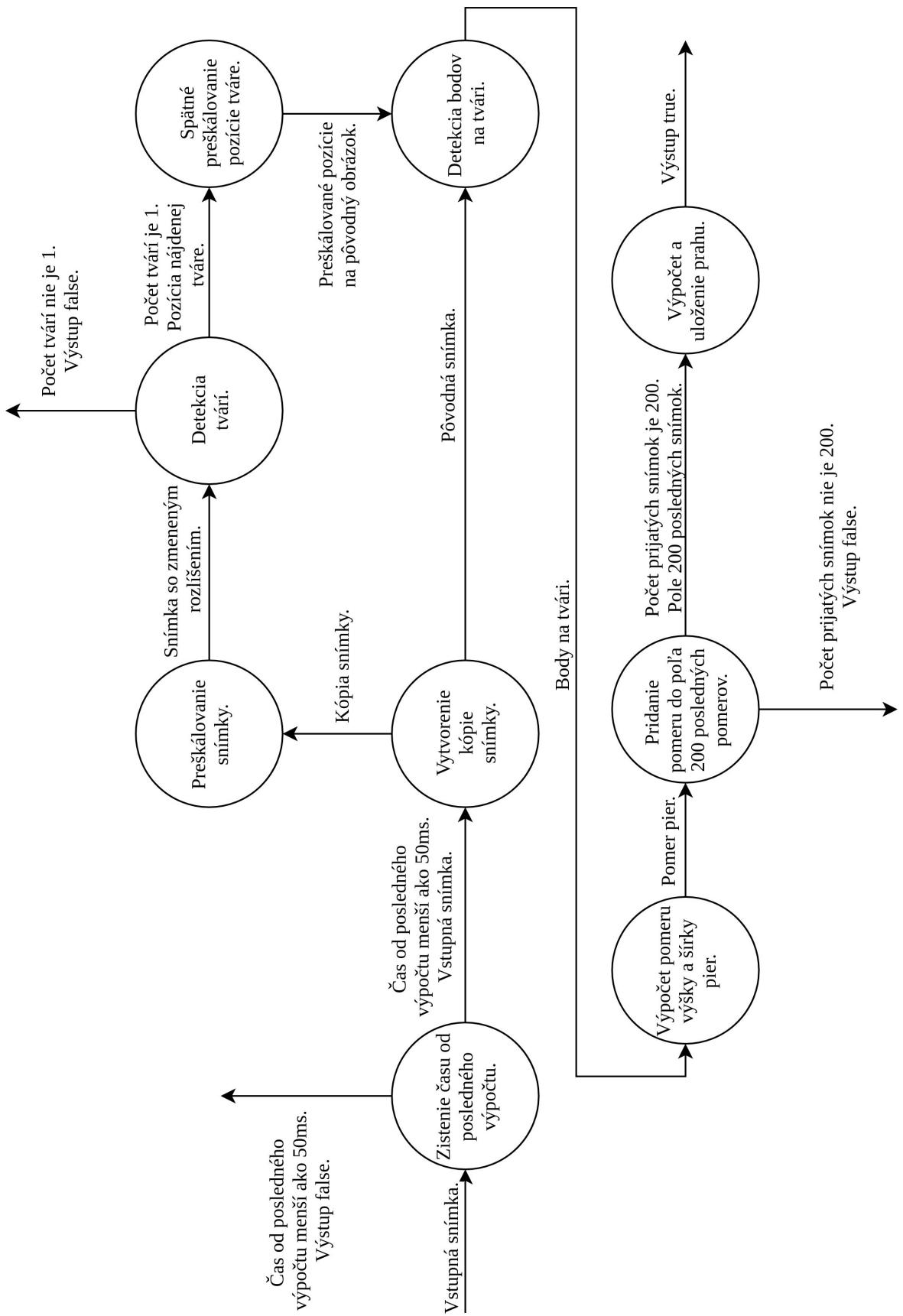
## Metóda FrameForLearningThreshold

Pred začatím používanie detekcie reči je potrebné určiť prah, ktorý sa používa v metóde **Frame**. Keďže každý človek ma ústa trocha iného tvaru, každý bude potrebovať iný prah. Vymysleli sme poloautomatickú metódu určenia prahu, ktorá je účinná a používateľsky prívetivá. Pred začatím používania metódy **Frame** je potrebné posielat približne desať sekúnd snímky metóde **FrameForLearningThreshold**, ktorá určí prah. Počas týchto desiatich sekúnd musí používateľ neustále priamo pozerať do kamery a kedykoľvek povedať vetu: „Toto je trénovacie video.“ a okrem tejto vety nepohybovať perami. Takto počas desiatich sekúnd ukladá metóda **FrameForLearningThreshold** vypočítané pomery do poľa 200 posledných pomerov. Hned' ako sa pole zaplní sa zavolá metóda **FindThreshold**, ktorá tieto hodnoty usporiada podľa veľkosti a ako nájdený prah vráti dvanásť najväčšiu hodnotu. Hodnota dvanásť bola zvolená pri testovaní ako najvhodnejšia. Príklad takto určeného prahu je na obr. 20.



Obr. 20: Graf usporiadaných absolútnych hodnôt pomerov, na základe ktorých sa určuje prah. Dvanásť najväčšia hodnota je vrátená ako prah.

Ked' máme určený prah, metóda `FrameForLearningThreshold` vráti pravdivostnú hodnotu pravda a tým sa určovanie ukončí a programátor, ktorý bude používať knižnicu VVAD bude vedieť, že prah je nastavený. Vo všetkých predchádzajúcich volaniach vračia metóda `FrameForLearningThreshold` pravdivostnú hodnotu nepravda. Detailný popis metódy `FrameForLearningThreshold` je na obr. 21.



Obr. 21: Graf priebehu metódy **FrameForLearningThreshold**. Vo vrcholoch sú akcie ktoré sa vykonávajú, šípky znázorňujú výstupy z akcií.

Rovnako ako v metóde `Frame` je možné volať metódu `FrameForLearningThreshold` ľubovoľný počet krát za sekundu, opäť však minimálne dvadsať krát a metóda spracúava rovnakým spôsobom každú snímku, ktorá prišla po 50 ms od predchádzajúcej spracovanej snímky.

## Implementácia AVAD

S použitím knižnice SFML [19] sa nám čiastočne podarilo implementovať algoritmus z práce [11]. Kvôli problémom s implementáciou, kvôli zložitosti výsledného riešenia a kvôli problémom so spojením VVAD a AVAD sme sa rozhodli AVAD do nášho riešenia neimplementovať.

# Testovanie

Implementovanú knižnicu VVAD sme nakoniec podrobili testovaniu. Museli sme navrhnúť metodiku testovania a spôsob vyhodnocovania. Natočili sme niekoľko trénovacích a testovacích videí a na nich sme vyhodnotili funkčnosť implementácie.

## Metodika testovania

Knižnica VVAD sa skladá z dvoch hlavných metód: **Frame** a **FrameForLearning-Threshold**. **FrameForLearningThreshold** určí prah, podľa ktorého následne **Frame** určuje, či prebieha reč alebo nie. Na to, aby sme otestovali funkčnosť týchto metód natočili sme dvojice sekvencií trénovacích a testovacích videí. Poprosili sme niekoľko kolegov o nahranie týchto videí, kde v trénovacom videu, ktoré malo dĺžku viac ako 10 sekúnd, mali povedať vetu „Toto je trénovacie video.“ a okrem toho mať ústa zatvorené a nehýbať nimi. V testovacom videu, ktoré malo ľubovoľnú dĺžku, mali povedať dve vety. Prvá veta bola „Pod'me vyskúšať, či je dobrý threshold.“, kde za čiarkou mali urobiť v reči väčšiu pauzu. Druhá veta bola „Toto bolo testovacie video.“. Aj medzi týmito dvoma vetami mali spraviť väčšiu pauzu. Dôvodom týchto paúz bolo vyčlenenie úsekov vo videu s rečou a bez reči.

Testovacie video bolo použité na určenie prahu pomocou metódy **FrameForLearning-Threshold** a prah bol následne použitý v metóde **Frame** na určenie, či prebieha alebo neprebieha reč v prislúchajúcom testovacom videu. Na jednotlivých testovacích videách určíme časy, kedy prebieha a neprebieha reč a budeme ich porovnávať s výstupom metódy **Frame**.

## Testovanie bodov na perách

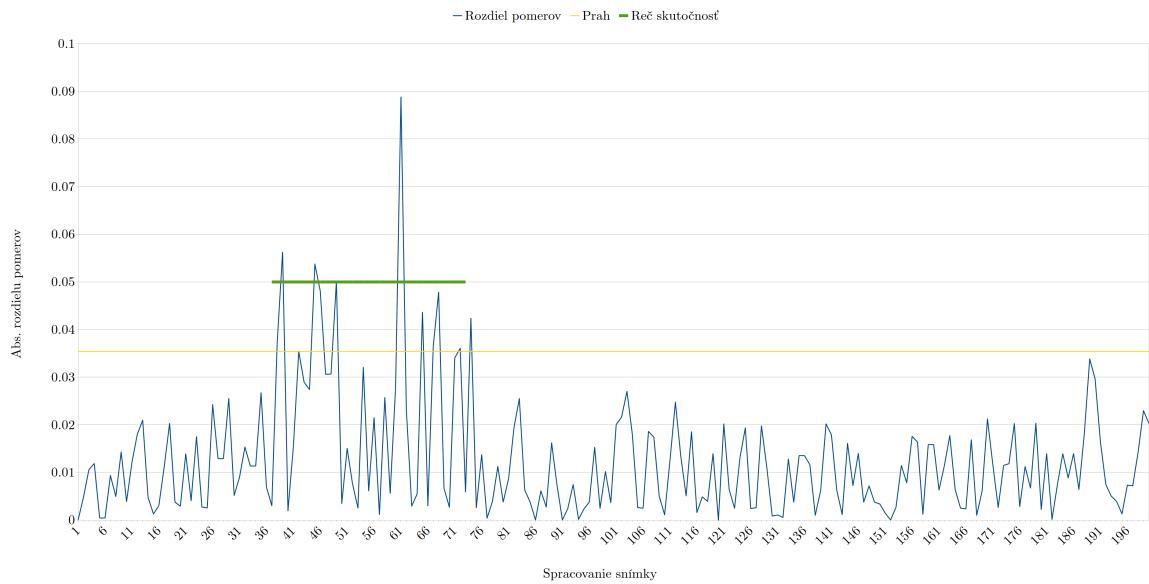
V časti 4.2.1.2 sme popísali použitie bodov na perách, z ktorých sa vypočítava pomer výšky a šírky pier. Brali sme priemery vnútorných a vonkajších bodov na perách. Rozhodli sme sa otestovať vhodnosť týchto bodov. Testovanie prebiehalo na pôvodnom 68 bodovom modeli a použili sme testovacie a trénovacie video, na ktorom je natočený

autor práce. Upravili sme a 3 krát spustili trénovanie a testovanie na týchto dvoch videách s tým, že raz sme brali vonkajšie body na perách (na obr 19 horný bod ako priemer bodov 50 a 52, dolný ako priemer bodov 58 a 56, ľavý bod bol bod 48 a pravý 54), druhý raz vnútorné body (na obr 19 horný bod ako priemer bodov 61 a 63, dolný ako priemer bodov 67 a 65, ľavý bod bol bod 60 a pravý 64) a tretí raz priemer vnútorných a vonkajších bodov ako na obrázku 19. Na nasledujúcich obrázkoch 22, 23, 24, 25, 26, 27 sa nachádzajú grafy popisujúce výsledky testovania.

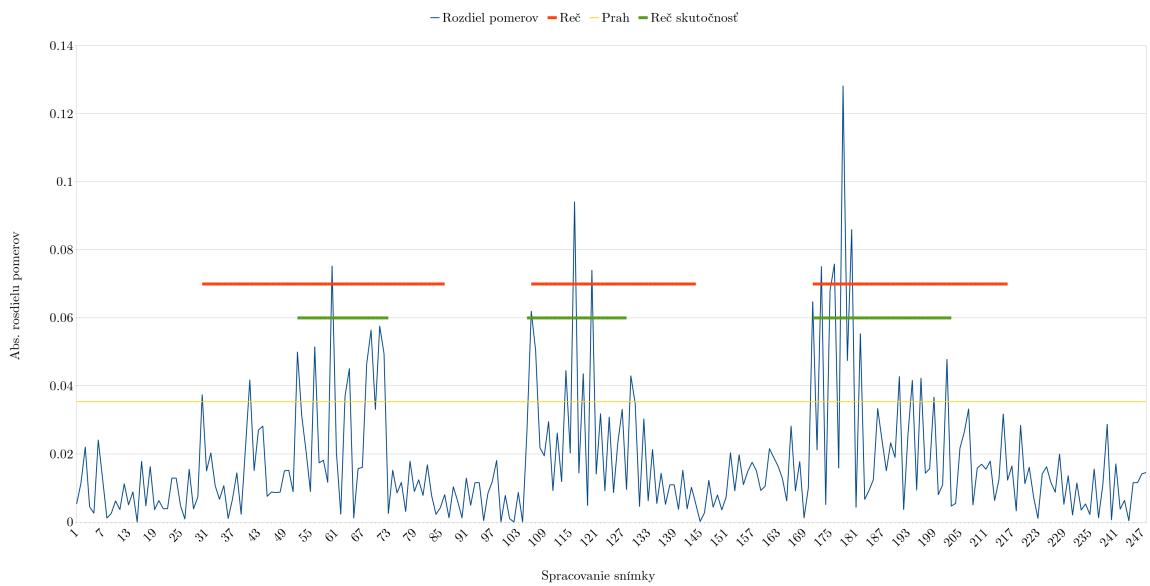
(Aby sme neuvádzali dlhý popis pri každom grafe uvedieme ho teraz:

- na vodorovnej osi sa chádza spracovanie snímky metódou **Frame** alebo **FrameForLearningThreshold**,
- na zvislej osi je absolútна hodnota rozdielu aktuálneho pomeru a pomeru z predchádzajúcej snímky,
- modrou farbou je zobrazená aktuálna absolútна hodnota rozdielu aktuálneho pomeru a pomeru z predchádzajúcej snímky,
- žltou farbou je zobrazený prah určený metódou **FrameForLearningThreshold**,
- zelenou farbou je zobrazená oblasť, kde vo videu prebiehala v skutočnosti reč,
- červenou farbou je zobrazená oblasť, kde vo videu určila metóda **Frame** reč.

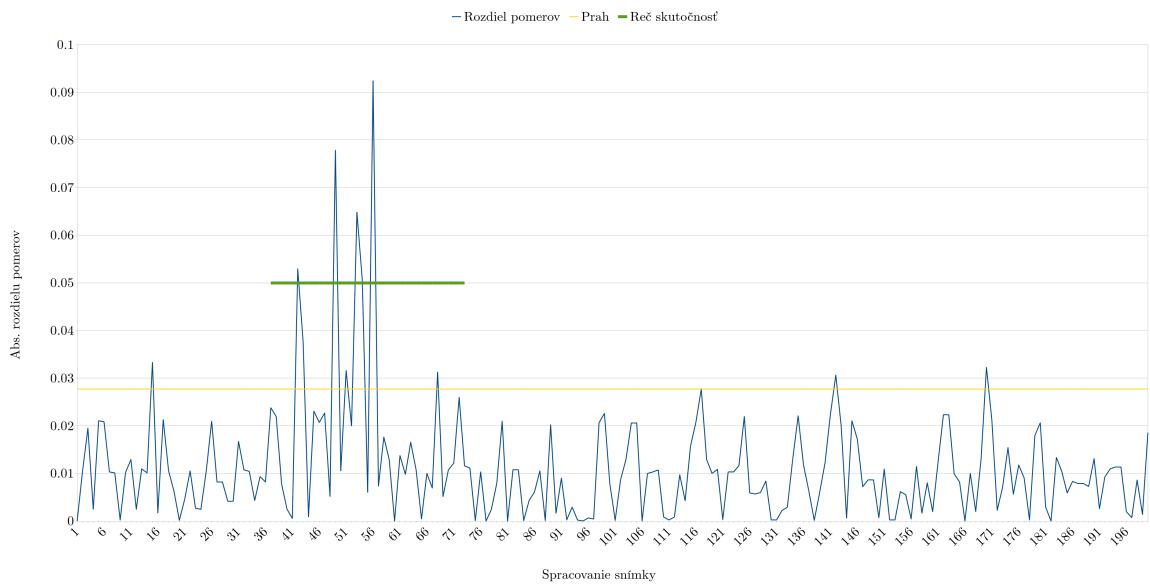
)



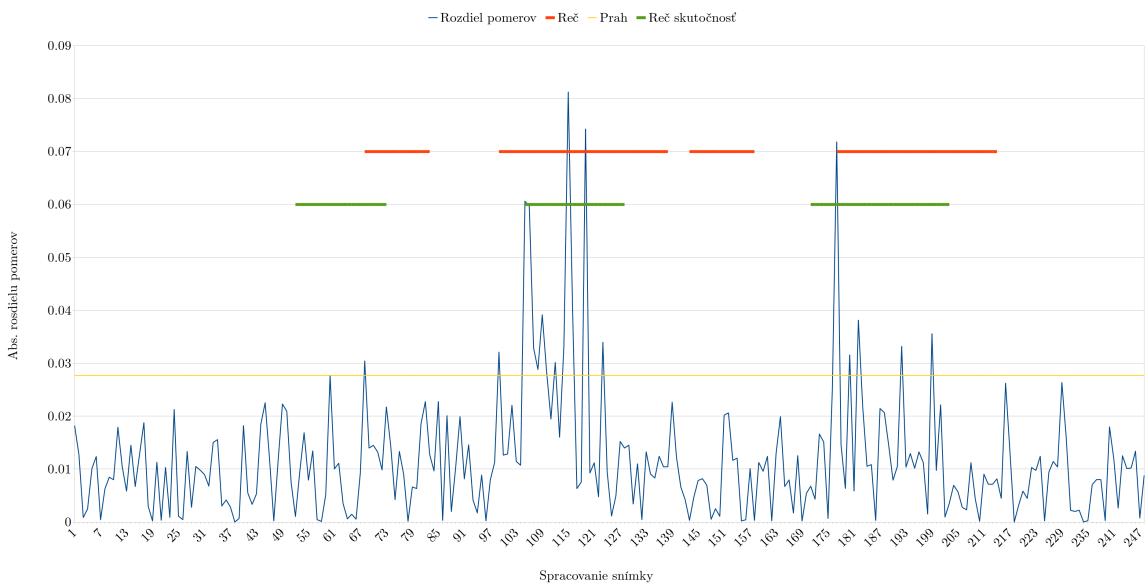
Obr. 22: Graf absolútnych hodnôt pomerov na vonkajších bodoch pier na trénovacom videu s autorom práce.



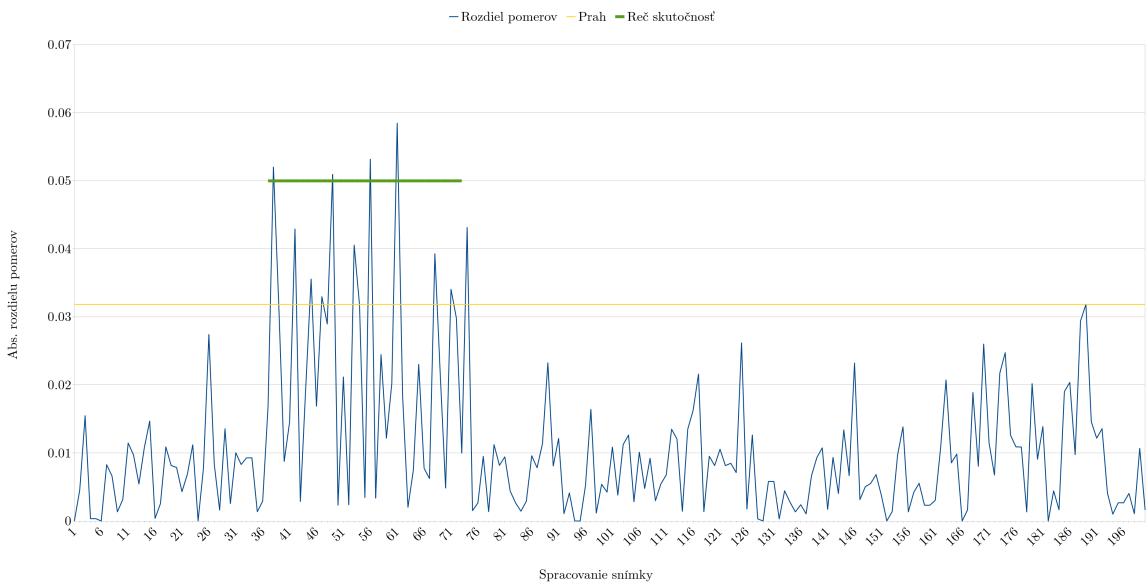
Obr. 23: Graf absolútnych hodnôt pomerov na vonkajších bodoch pier na testovacom videu s autorom práce.



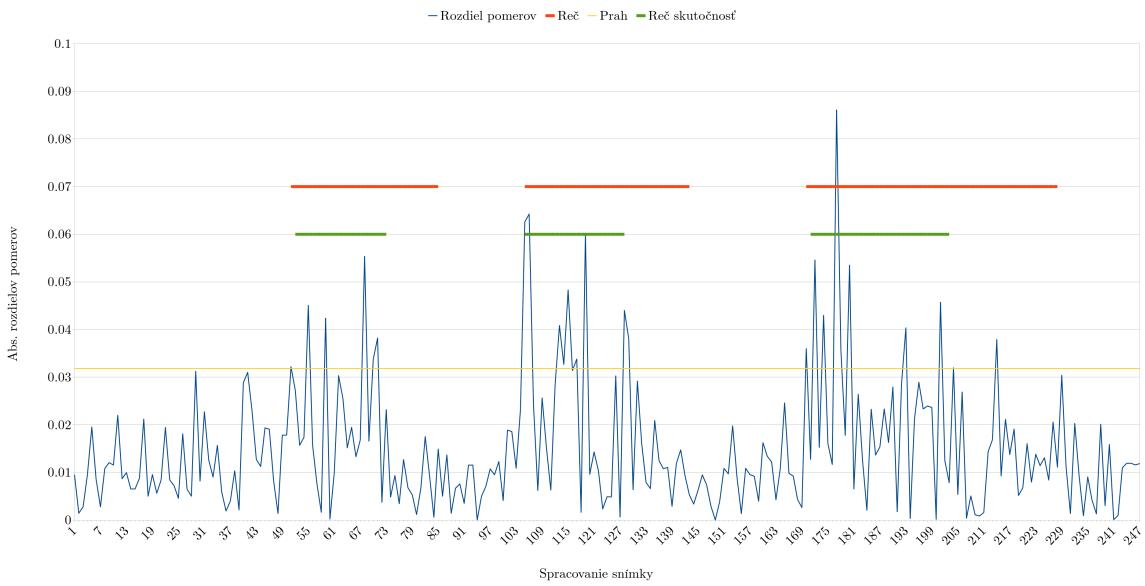
Obr. 24: Graf absolútnych hodnôt pomerov na vnútorných bodoch pier na trénovacom videu s autorom práce.



Obr. 25: Graf absolútnych hodnôt pomerov na vnútorných bodoch pier na testovacom videu s autorom práce.



Obr. 26: Graf absolútnych hodnôt pomerov na priemeroch vnútorných a vonkajších bodov pier na trénovacom videu s autorom práce.



Obr. 27: Graf absolútnej hodnoty rozdielov pomerov na priemeroch vnútorných a vonkajších bodov pier testovacom videu s autorom práce.

Z uvedených grafov je vidno, ktorá metóda ma akú presnosť. Vnútorné body nie sú vhodné už pri pohľade na trénovacie video, lebo v druhej polovici časti kde prebiehala reč boli absolútne hodnoty rozdielov pomerov veľmi nízke. Aj ukážka následnej detektie nie je presná. V prípade vonkajších bodov a priemeru vnútorných a vonkajších bodov vyzerá graf trénovania dobre. Graf testovania ale ukazuje, že použitie priemeru vnútorných a vonkajších bodov je vhodnejšie, lebo je presnejšie na detegovanie začiatku reči.

## Testovanie natrénovaného modelu

Ked'že sme sa rozhodli vytvoriť vlastný model pre hľadanie bodov na tvári, je potrebné najprv overiť jeho presnosť. Vytvorili sme 2 nové modely, úpravou xml súborov pre 68 bodový model, odstránením niektorých bodov:

- 20 bodový model pier. Ponechali sme len body na perách. Už prvé pokusy natrénovať takýto model ukázali, že nie je vhodný. Dosť často sa stávalo, že model našiel pery v oblasti brady alebo medri bradou a perami. Dôvodom je podobnosť týchto oblastí s perami. Ukážka chybovosti modelu je na obr. X.
- 27 bodový model pier. Kvôli zvýšeniu presnosti modelu sme okrem 20 bodov na perách nechali body na spodnej časti uši, bod na brade, nose, medzi očami a vonkajšie krajné body očí. Skúšali sme rôzne parametre trénovania. Vačšina

natrénovaných modelov bola stále nepresná, ústa boli užšie ako mali byť a podobne. Nakoniec sa podarilo nájsť parametre, s ktorými vyzeral byť natréovaný model dostatočne presný. Ukážka natréovaného modelu je na obr. X.

Trénovanie najpresnejšieho modelu trvalo približne 28 hodín na počítači HP Z240. Uvedieme použité parametre trénoania aj s popisom:

- `_cascade_depth = 12` - hĺbka kaskády,
- `_tree_depth = 3` - hĺbka stromu,
- `_num_trees_per_cascade_level = 600` - hĺbka stromov v úrovni kaskády,
- `_nu = 0,1` - regularizačný parameter,
- `_oversampling_amount = 550` - množstvo prevzorkovaní, teda veľkosť trénovacej množiny sa zväčší náhodne zvolenou transformáciou obrázkov z množiny,
- `_feature_pool_size = 750` - počet náhodne vybraných pixelov z obrázka v každej úrovni kaskády,
- `_lambda = 0,2` - na rozhodnutie ako rozdeliť uzly v regresných stromoch sa algoritmus pozera na náhodné dvojice pixelov a tento parameter určuje ako daleko sú od seba - číslo bližšie pri nule znamená, že pixely sú bližšie pri sebe, bližšie pri jednotke, že dalej od seba,
- `_num_test_splits = 300` - počet náhodne generovaných rozdelení pri generovaní stromov na každom uzle,
- `_verbose` - pravdivostná premenná, ktorá ak platí, tak program bude pri trénovaní vypisovať v akom je stave, inak nebude,
- `_num_threads = 8` - počet vlákien, ktoré sa používajú pri trénovaní,

## Testovanie videí

Videá boli natáčané mobilným telefónom Xiaomi Pocophone F1 v rozlíšení 1280x720 pixelov pri 30 dnímkach za sekundu. Vytvorili sme testovací program `testVVAD`, v ktorom za pomoci knižnice OpenCV načítavame snímky z jednotlivých videí a dávame ich na vstup metódam `Frame` a `FrameForLearningThreshold`. Snímku z videa získavame každých 33 ms, tak aby obraz šiel plynulo vzhľadom na to, že videá boli natočené so snímkovacou frekvenciou 30. Vytvorili sme spolu X dvojíc videí, ku každému testovaciemu videu sme určili časy začatia a ukončenia reči a výsledky sme zobrazili za pomocí

grafov. Na nasledujúcom grafe X je zobrazených 200 rozdielov absolútnej hodnôt rozdielov pomerov v dvoch po sebe nasledujúcich spracovaniach snímky metódou `FrameForLearningThreshold` na trénovacom videu, na ktorom je natočený autor práce. Metóda určila prah X.

Použitie prahu metódou `Frame` je na grafe X.

## Zhodnotenie výsledkov testovania

Jedným z cieľov práce bolo navrhnúť vlastný HCI komponent a pilotne ho implementovať. Tento cieľ sa podarilo splniť. Vytvorená C++ knižnica je funkčná a dostatočne rýchla (na notebooku Asus Zenbook UX305FA trvalo spracovanie snímky metódou `Frame` menej ako 30 ms) aj na menej výkonných počítačoch. Ak chceme zhodnotiť presnosť a teda použiteľnosť knižnice z testovania uvedeného v predchádzajúcej kapitole vyplýva, že knižnica by mohla byť použiteľná aj v praxi.

# Záver

V práci sme sa zaoberali prvkami HCI, popísali sme ich história a aktuálne možnosti. Zameriavame sa na detekciu reči a hľadali možnosti využitia tejto detekcie na automatické vypínanie a zapínanie mikrofónu pri videokonferenčnom hovore. Vytvorili sme prehľad existujúcich riešení v nastolenej problematike. Na základe naštudovanej literatúry sme navrhli a implementovali dynamickú C++ knižnicu VVAD, ktorú je možné použiť na detekciu tvári v obraze kamery a na detekciu reči u osoby pred kamerou. Nakoniec sme knižnicu otestovali a ukazuje sa ako reálne použiteľná. Do budúcnosti je možné:

- vylepšiť detekciu reči pomocou detekcie zo zvuku,
- urobiť knižnicu viac modulárnu - rozdeliť ju na viac knižníc, kde jedna by detegovala tváre, ďalšia reč a podobne,
- nájsť rýchlejsiu a presnejšiu možnosť detekcie tvári,
- vytvoriť presnejší model detekcie bodov na tvári, . . .

# Zoznam použitej literatúry

- [1] AOKI, M., MASUDA, K., MATSUDA, H., TAKIGUCHI, T., AND ARIKI, Y. Voice activity detection by lip shape tracking using ebgm. In *Proceedings of the 15th ACM international conference on Multimedia* (2007), ACM, pp. 561–564.
- [2] GOOGLE. Google home. Dostupné na internete: [https://store.google.com/us/product/google\\_home?hl=en-US](https://store.google.com/us/product/google_home?hl=en-US). [cit. 21. 1. 2019].
- [3] GUPTA, V. Face detection – opencv, dlib and deep learning ( c++ / python ). Dostupné na internete: <https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/>. [cit. 21. 1. 2019].
- [4] IBUG. 300 faces in-the-wild challenge (300-w), imavis 2014. Dostupné na internete: [https://ibug.doc.ic.ac.uk/resources/300-W\\_IMAVIS/](https://ibug.doc.ic.ac.uk/resources/300-W_IMAVIS/). [cit. 21. 1. 2019].
- [5] INTEL. Intel® realsense sdk for windows\* (discontinued). Dostupné na internete: <https://software.intel.com/en-us/realsense-sdk-windows-eol>. [cit. 21. 1. 2019].
- [6] INTEL. Intel® realsense™ sdk 2.0. Dostupné na internete: <https://realsense.intel.com/sdk-2/>. [cit. 21. 1. 2019].
- [7] JOOSTEN, B., POSTMA, E., AND KRAHMER, E. Voice activity detection based on facial movement. *Journal on Multimodal User Interfaces* 9, 3 (2015), 183–193.
- [8] KAZEMI, V., AND SULLIVAN, J. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1867–1874.
- [9] KING, D. Dlib. Dostupné na internete: <http://dlib.net/>. [cit. 21. 1. 2019].
- [10] MALLICK, S. Facial landmark detection. Dostupné na internete: <https://www.learnopencv.com/facial-landmark-detection/>. [cit. 21. 1. 2019].

- [11] MOATTAR, M. H., AND HOMAYOUNPOUR, M. M. A simple but efficient real-time voice activity detection algorithm. In *Signal Processing Conference, 2009 17th European* (2009), IEEE, pp. 2549–2553.
- [12] NONAME. Calcflow. Dostupné na internete: <http://calcflow.io/>. [cit. 21. 1. 2019].
- [13] NĚMEČKOVÁ, L. Rozvoj problematiky hci (human-computer interaction) na Úisk ff uk. Dostupné na internete: [http://clovek.ff.cuni.cz/pdf/nemeckova\\_zprava\\_18.pdf](http://clovek.ff.cuni.cz/pdf/nemeckova_zprava_18.pdf), 2010. [cit. 21. 1. 2019].
- [14] OPENCV. opencv. Dostupné na internete: [https://docs.opencv.org/4.0.1/d2/d42/tutorial\\_face\\_landmark\\_detection\\_in\\_an\\_image.html](https://docs.opencv.org/4.0.1/d2/d42/tutorial_face_landmark_detection_in_an_image.html). [cit. 21. 1. 2019].
- [15] RANJAN, R., PATEL, V. M., AND CHELLAPPA, R. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).
- [16] SAGONAS, C., ANTONAKOS, E., TZIMIROPOULOS, G., ZAFEIRIOU, S., AND PANTIC, M. 300 faces in-the-wild challenge: Database and results. *Image and vision computing* 47 (2016), 3–18.
- [17] SAGONAS, C., TZIMIROPOULOS, G., ZAFEIRIOU, S., AND PANTIC, M. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2013), pp. 397–403.
- [18] SAGONAS, C., TZIMIROPOULOS, G., ZAFEIRIOU, S., AND PANTIC, M. A semi-automatic methodology for facial landmark annotation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2013), pp. 896–903.
- [19] SFML. Sfml - simple and fast multimedia library. Dostupné na internete: <https://www.sfml-dev.org/>. [cit. 21. 1. 2019].
- [20] TOMORI, Z. Microrobotics. Dostupné na internete: <http://home.saske.sk/~tomori/tweezers.htm>. [cit. 21. 1. 2019].
- [21] VIERIU, L. Real-time voice activity detection using a simple webcam. *Proceedings of WCSIT* (2014).

# Prílohy

V prílohách sa nachádzajú nasledujúce položky:

**Príloha A:** CD médium - diplomová práca v elektronickej podobe, komentované zdrojové kódy.

**Príloha B:** Grafy vyhodnotenia testovacích a trénovacích videí s popismi.