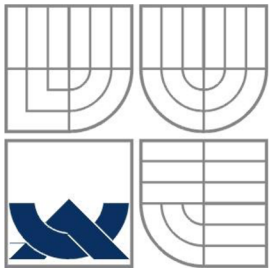
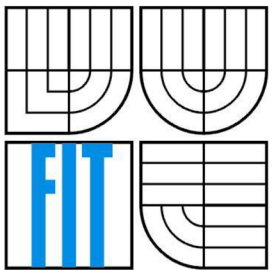


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# OVLÁDÁNÍ POČÍTAČE GESTY

VIDEO-BASED HUMAN-COMPUTER INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — David Chrápek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Michal Španěl
SUPERVISOR

BRNO 2010

# Abstrakt

Tato bakalářská práce je zaměřena na ovládání počítače pohybem celé hlavy. Pomocí sledování vzájemné polohy očí a nosu je určován směr natočení hlavy a pomocí pohybu hlavy jsou rozpoznávána jednoduchá gesta. K zpracování obrazu byly využity Haar-kaskády klasifikátorů založené na metodě Viola-Jones k rozpoznání hlavy, očí a nosu ve snímaném obraze a k jejich sledování byla využita především metoda Lucas-Kanade optického toku.



# Abstract

This bachelor´s thesis is focused on video-based human-computer interface controlled by movement of the whole head. By tracking relative positions of eyes and nose angles of head are determined and by movement of head simple gestures are recognized. For image processing Haar-cascades based on Viola-Jones of classifiers were used to recognize head, eyes and nose within the image and for their tracking mainly Lucas-Kanade optical flow method was used.



# Klíčová slova

Ovládání počítače gesty, Zpracování obrazu, Rozpoznávání gest, Haar-kaskády, Lucas-Kanade, Viola-Jones

# Keywords

Video-Based Human-Computer Interface, Image processing, Gesture recognition, Haar-cascades, Lucas-Kanade, Viola-Jones


# Citace

Chrápek David: Ovládání počítače gesty, bakalářská práce, Brno, FIT VUT v Brně, 2010

# Ovládání počítače gesty

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Španěla

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

David Chrápek

19.května 2010

## Poděkování

Především bych chtěl poděkovat mému vedoucímu Ing. Michalu Španělovi za odborné vedení, rady a čas, který mi věnoval. Dále pak také Tomáši Ižákovi za demonstrační příklad 3D scény, na kterém je předvedeno natáčení této scény pohybem hlavy. V neposlední řadě bych také rád poděkoval rodině, přátelům a hlavně mé lásce za ohromnou podporu. Děkuji moc Vám všem.

# Obsah

# 1 Úvod

Prvním typem komunikace mezi lidmi byla komunikace neverbální, tedy řeč gest a posunků. Pak se začal člověk dorozumívat slovy, následovala strukturovaná řeč a nakonec i písmo. Vývoj komunikace mezi člověkem a počítačem také proběhl v několika fázích. Na začátku byly pouze děrné štítky a pásky, později klávesnice, pak ke klávesnicím přibyly i počítačové myši. Nejnovější přístupy v komunikaci člověka s počítačem jsou pak obrazové vjemy, hlas a popřípadě různá polohovací zařízení, jako jsou například senzory na rukou.

Oblast zpracování obrazu je v poslední době velmi diskutovanou a zkoumanou oblastí a vzniká nespočet vědeckých prací a projektů v této oblasti. Jde o poměrně rychle se vyvíjející oblast se spoustou rozličných přístupů a metod využitých při řešení problémů, které se při implementacích projektů vyskytují.

Specifičtější oblastí pak je rozpoznávání informací v obraze počítačem tak, aby jim počítač „rozuměl“. Tato oblast už je poměrně náročná, jelikož počítač není schopen autonomního myšlení. Proto musíme počítač „naučit“ porozumět v obraze specifikům, která jsou pro náš projekt důležitá.

Snad ještě těžším a specifičtějším teritoriem v této oblasti je naučit počítač porozumět lidským gestům, protože mnohdy jim nerozumí ani jiný člověk. Každý dané gesto interpretuje trochu odlišně. Zatím se systémy, kde člověk ovládá počítač gesty, netěší masivnímu rozšíření mezi uživateli počítačů, ale dle mého názoru je to jen otázka času. Dříve takovéto systémy nebyly příliš přesné a vzhledem k výkonu personálních počítačů ani nemohly být využívány běžným uživatelem a tudíž se jejich potenciál příliš nerozvíjel. Avšak s nástupem poměrně výkonných personálních počítačů a rozšíření tak zvaných webkamer mohl naplno začít vývoj aplikací pro rozpoznávání lidských gest. V dnešní době jsou již běžné kancelářské počítače schopny takovou aplikaci spouštět v reálném čase, a tudíž tato platforma nabývá na významu.

V mé bakalářské práci jsem využil některé metody používané v oblasti detekce a sledování objektů v obraze a doplnil tyto metody o vlastní přístup k rozpoznávání jednoduchých gest na základě pohybu objektu v obraze a také jsem sestavil zjednodušené rovnice pro výpočet natočení hlavy pomocí pozice očí a nosu.

# 2 Použité metody

Než začnu popisovat vytvořenou aplikaci, je potřeba objasnit některé z metod, které v aplikaci využívám a jejichž nejsem autorem. Ve zkratce zde popíši princip fungování kaskády klasifikátorů pro detekci objektů v obraze a metodu Viola-Jones, základní informace o optickém toku mezi dvěma následujícími obrazy a stručně i metodu Lucas-Kanade. Pro detailnější informace, především obecného charakteru, z oblasti zpracování obrazu poslouží například [3, 7, 8].

## 2.1 Detekce objektů v obraze

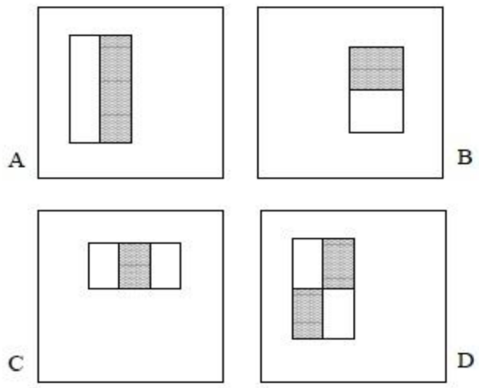
Existuje několik způsobů, jak detekovat objekty v obraze. Jsou to například neuronové sítě (neural nets), porovnávání grafů (graph matching), genetické algoritmy (genetic algorithms), simulované žíhání (simulated annealing), fuzzy logika (fuzzy logic) nebo třeba statistické rozpoznávání vzorů (statical pattern recognition). Každý z daných přístupů má své pro a proti. Metoda, která je využívána v knihovně OpenCV v Haar-kaskádách klasifikátorů je metoda Viola-Jones, a proto jsem si také tuto metodu zvolil a využil ji pomocí OpenCV knihovny.

### 2.1.1 Viola-Jones

V této podkapitole nastíním základní princip metody pánů Viola a Jones dle [1].

Rozpoznávací metody jsou založeny na principu porovnávání vybraných vlastností vzoru a výřezu obrazu, který je kandidátem na rozpoznávaný objekt. Těmito vlastnosti mohou být například pixely obrazu (pixel-based features). U metody Viola-Jones jsou to však obdélníkové oblasti, které se v různých regionech objektu od sebe odečítají. Jelikož se zde zabýváme detekcí hlavy v obraze, budu názorně vysvětlovat tuto metodu právě na příkladu detekce hlavy v obraze.

Metoda Viola-Jones používá tyto 4 typy rozdělení regionu na obdélníky, jak jsou znázorněny na obrázku 2.1.1.1.




*OBR. 2.1.1.1: Rozdělení regionů vlastností metody Viola-Jones*

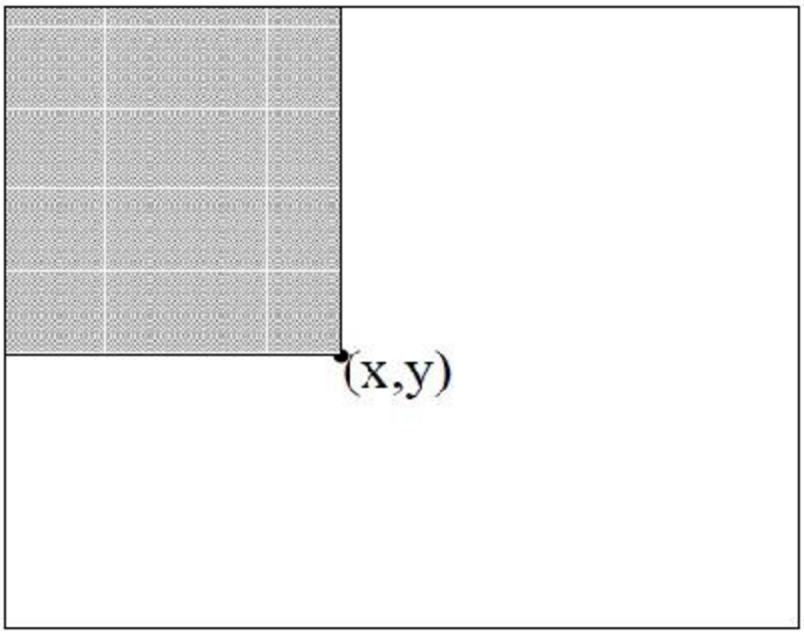
Hodnota dané vlastnosti je vždy jednoduše spočítána jako rozdíl sum začerněných a prázdných obdélníků. Suma dané oblasti (obdélníku) je vypočtena jako suma všech intenzit obrazových bodů dané oblasti.

Jelikož metoda pracuje s obrazem ve stupních šedi, je intenzita daného bodu obrazu vypočtena z intenzit jeho barevných složek podle známého empirického vztahu, jak jej popisuje následující rovnice:

$$I = 0.299R + 0.587G + 0.114B\text{h} \qquad \textit{Rovnice 2.1.1.1}$$

kde R je intenzita červené složky, G intenzita zelené složky a B intenzita modré složky barvy.

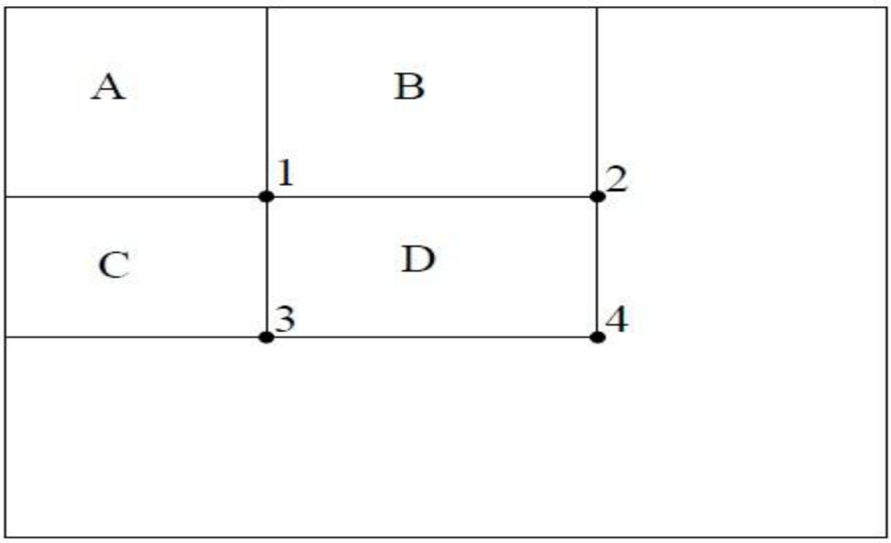
Pomocí metody integrovaných obrazů (integral image, summed area table) je potom výpočetně nenáročné určit intenzitu kterékoliv výřezu v obrazu, jelikož hodnoty intenzit pro všechny body v obraze jsou vypočteny předem jako součet všech intenzit bodů nad a vlevo od daného bodu, jak je znázorněno vyplněnou oblastí na obrázku 2.1.1.2.




*OBR. 2.1.1.2: Integral Image*

Potom například oblast D na obrázku 2.1.1.3 se vypočte jednoduše jako:

$$I = 4 - ((2 + 3) - 1)\text{h}$$ *Rovnice 2.1.1.2*




*OBR. 2.1.1.3:*

Tyto principy se pak využívají při detekci. Například můžeme říci, že oblast očí bývá tmavší, než oblast nosu a lící a tudíž tuto skutečnost můžeme popsat vlastností s následujícím rozložením dvoj-obdélníkového okolí, jak znázorňuje obrázek 2.1.1.4.

*OBR. 2.1.1.4: Dvoj-okolí*

Pak bychom mohli říci, že oblast mezi očima je světlá, na rozdíl od očí samotných a reprezentovat tuto skutečnost troj-obdélníkovým okolím, jako na obrázku 2.1.1.5

*OBR. 2.1.1.5: Troj-okolí*

Na základě stejného principu je hlava popsána při detekci mnohem podrobněji. Tyto vlastnosti jsou řazeny do klasifikátorů, které podle těchto vlastností rozpoznávají, jak moc se daný obraz podobá hledanému objektu.

Pro detekci lidské hlavy je použito řádově desítky až stovky tisíc takových vlastností řazených do několika desítek až stovek klasifikátorů. Klasifikátory po přijmutí příslušného obrazu určí, zda má dané vlastnosti, žádoucí pro hledaný objekt. Pokud všechny klasifikátory odpoví, že daný obraz vyhovuje, je považován za hledaný objekt. Takto by však bylo nutné pro všechny možné velikosti objektu, posunutých do všech možných pozic vstupního obrazu, počítat všech několik set tisíc vlastností. Proto metoda Viola-Jones zavedla užití tak zvaných kaskád těchto klasifikátorů. Kaskády mají několik úrovní a v každé mají zařazeno několik klasifikátorů. Pro dané měřítko a pozici v obraze se počítá nejdříve nejnižší úroveň této kaskády a popřípadě ty další. Ale jakmile jedna úroveň zamítne daný obraz, je detekce pro tuto oblast okamžitě ukončena. Nižší úrovně kaskády pak bývají složeny z klasifikátorů obecnějších, které jsou relativně rychle spočitatelné a vylučují až 60% nevhodných objektů. Dalších 40% pak musí být vyloučeno vyššími úrovněmi. Tato procentuální úspěšnost vyloučení neplatných objektů se snižuje geometricky s narůstající úrovní kaskády. Výpočetní náročnost vyšších úrovní kaskády naopak roste. Jelikož v obraze většinou hledáme hlavu jednu a ta nemusí být přes celý obraz, znamená toto řešení značnou úsporu času, protože pro naprostou většinu kandidátů je nutno počítat pouze první úroveň kaskády, která je zamítne a ty

výpočetně náročnější kaskády se vypočítávají pouze v případě objektů velice podobným hlavě, nebo když v obrazu opravdu hlava je. Těchto je však v obraze minimum.

Kaskády jsou automaticky natrénované na několika tisících (v případě lidské tváře) vzorků pozitivních, které daný objekt obsahují a na několika tisících náhodně vybraných obrazů, které daný objekt neobsahují. Natrénování daných kaskád se provádí pomocí AdaBoostu dle [2].

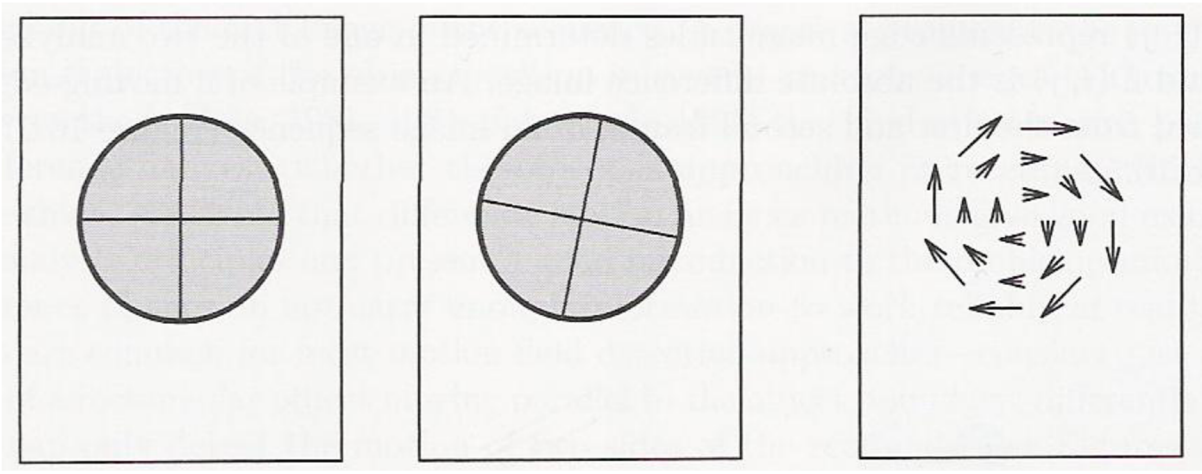
Také v zájmu rychlosti je z nepřeberného množství vlastností v oblasti obličeje potřeba vybrat jen ty nejvýznamnější, které povedou k rychlé a správné detekci. Nemá smysl počítat vlastnosti všechny.

## 2.2 Sledování objektů v obraze

Pro sledování objektů v obraze jsem si zvolil metodu Lucas-Kanade, jelikož je jednou z nejpoužívanějších metod v oblasti optického toku, který se využívá na sledování pohybů v obraze. Tato metoda je implementována v knihovně OpenCV, kterou jsem v hojné míře využíval. Nejdříve však okrajově vysvětlím, co je to optický tok a pak nastíním princip metody Lucas-Kanade.

### 2.2.1 Optický tok

Optický tok je zjevný posun objektů (hran, povrchů) v obrazové scéně na základě pohybu kamery, objektů či obojího. Tohoto pohybu objektů v obraze využívají všechny techniky optického toku. Při vyhodnocování optického toku se hledají v následujících snímcích obrazové body se stejnou či podobnou intenzitou. Na obrázku 2.2.1.1 pak můžeme vidět optický tok mezi dvěma navazujícími obrazy.



*OBR. 2.2.1.1: Optický tok mezi snímky (obrázek z [3])*

Metod pro výpočet optického toku je několik druhů:

- Fázová korelace (Phase correlation)
- Blokové metody (Block-based methods)
- Metody diskrétní optimalizace (Discrete optimalization methods)
- Diferenciální metody

Mezi diferenciální metody patří například Horn-Shunck metoda [6], Black-Jepson, Buxton-Buxton metoda a také metoda Lucas-Kanade [4]. Tyto metody staví na parciálních derivacích obrazového signálu či na parciálních derivacích vyšších řádů hledaného pole toků.

Některé problémy při určování optického toku:

- Při pohybu například tmavého objektu obdélníkového typu ve směru jeho hran se při diferenci následujících obrazů zdá, že se hýbou pouze okraje obdélníka a střed zůstává na svém místě, jak je znázorněno na obrázku2.2.1.2.

*OBR. 2.2.1.2: Neúplné určení optického toku*

- Nerovnoměrné osvětlení scény - při pohybu objektu mění objekt své světelné charakteristiky (intenzitu jednotlivých obrazových bodů) a pokud je tato změna příliš náhlá, již se nám daný objekt nejeví jako objekt z předešlého snímku.

Toto byl pouze velice stručný výklad pojmu optický tok, tak jak jej chápeme v počítačovém vidění. Detailnější informace lze nalézt v [3].

## 2.2.2 Lucas-Kanade

Metoda Lucas-Kanade vychází pouze ze dvou po sobě jdoucích snímků a jejich diferencí. Předpokládá, že tok v okolí vybraného pixelu je konstantní v kterémkoliv čase. Jelikož metoda uvažuje, že okolí daného bodu má konstantní směr a rychlost toku, je pak daný systém přeurčitý (over-determined). Jak je vidět na následující rovnici, lze sestavit matici, která znamená sestavení mnohem více rovnic, než je neznámých:

$$\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}$$

*Rovnice 2.2.2.1*

Kde *Vx* a *Vy* je rychlost toku ve směrech osy *x* a *y*. *I* značí intenzitu pixelů. Z rovnice opravdu vyplývá, že tento systém je přeurčitý, protože v tomto případě vznikne mnohem více rovnic, než je hledaných proměnných, kterými jsou *Vx* a *Vy*.

Proto je nutné použít například metodu nejmenších čtverců pro nalezení optimálního řešení, jak je vidět na následující rovnici:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} I_{x_i}{}^2 & \sum_{i=1}^{n} I_{x_i} I_{y_i} \\ \sum_{i=1}^{n} I_{x_i} I_{y_i} & \sum_{i=1}^{n} I_{y_i}{}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{i=1}^{n} I_{x_i} I_{t_i} \\ -\sum_{i=1}^{n} I_{y_i} I_{t_i} \end{bmatrix}$$

*Rovnice 2.2.2.2*

Problémy, se kterými se metoda potýká, jsou:

- Pohyb je příliš rychlý (částečně lze řešit pyramidovým přístupem, to znamená, že vyhledávací okno se postupně rozšiřuje).
- Intenzita bodů, které mají být sledovány, je podobná intenzitě bodů, které tvoří pozadí. Tento problém není prakticky řešitelný. Pokud pozadí splývá se sledovaným objektem, je to vždy problém.
- Body v okolí zvoleného bodu se nepohybují ve stejném směru či rychlosti. Je zde problém se stanovením optimální velikosti okolí, se kterým bude metoda pracovat.

Toto byl pouze nástin metody Lucas-Kanade. Podrobnější informace lze nalézt v [4].

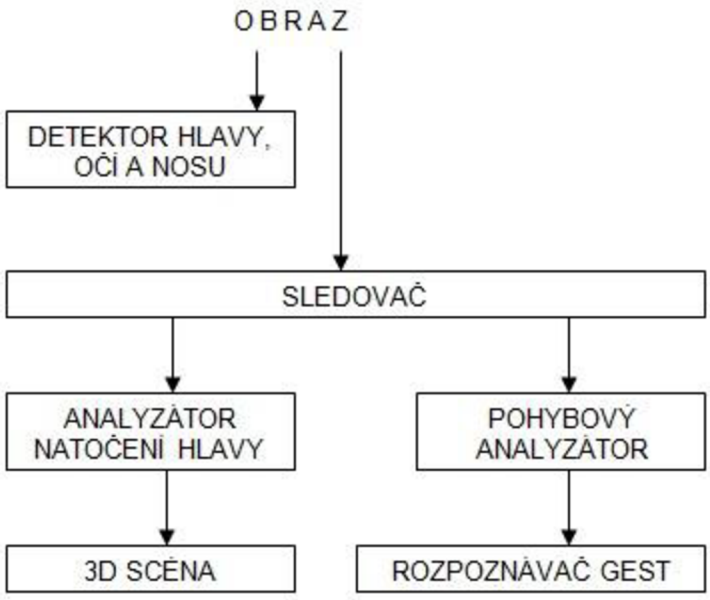
# 3 Popis aplikace jako celku

Cílem této bakalářské práce bylo vytvořit program, který bude reagovat na pohyb hlavy uživatele. K tomu, aby počítač byl schopen reagovat na pohyb hlavy v reálném čase, je samozřejmě zapotřebí kamerka připojená k počítači. Obrazový tok z kamerky je přesměrován do této aplikace, která jej dále zpracovává. Je zapotřebí obraz zpracovat, nalézt v něm hlavu uživatele, jeho oči a nos a ty dále sledovat. Na základě vzájemné polohy očí a nosu jsou určovány tři úhly natočení hlavy, a to: předklon/záklon, natočení doleva/doprava a náklon hlavy doleva/doprava. Pak dále na základě pohybu nosu jsou vyhodnocovány pohyby a z nich následně pak rozpoznávána gesta. Pro zjednodušení byl vybrán pouze nos, který byl upřednostněn před očima, jelikož je to bod přibližně uprostřed hlavy a při pohybu hlavy je nejvíce znatelná změna jeho polohy.

Aplikace je složena z těchto logických celků:

- Rozpoznávač obličeje, očí a nosu
- Sledovač očí a nosu
- Analyzátor natočení hlavy
- Pohybový analyzátor
- Rozpoznávač gest

Každému z těchto celků bude věnována samostatná kapitola. Propojení celků je pak názorně ukázáno na obrázku 3.1.




*OBRAZEK 3.1: Propojení logických celků aplikace*

Aplikace byla vyvíjena převážně na operačním systému Windows 7. Volba padla na operační systém Windows, jelikož nabízí více potencionálních uživatelů programu. Jako jazyk byl zvolen převážně jazyk C pro svou efektivitu a také kvůli mým znalostem, protože právě jazyk C ovládám na poměrně dobré úrovni. Pro zjednodušení některých úseků byly použity i principy objektového programování a jazyk C++. Právě překladačem jazyka C++ je následně celá aplikace překládána. Dále byla v hojné míře využita knihovna OpenCV ve verzi 2.0, jež je „open source“ multiplatformní knihovna zaměřená na zpracování obrazu. Instrukce k instalaci OpenCV knihovny naleznete zde [9]. Výbornou knihou zaměřenou na OpenCV knihovnu je [5].

# 4 Rozpoznávač obličeje, očí a nosu

Detekce obličeje, očí a nosu je základem celé aplikace. Bez správné detekce těchto objektů bychom nemohli pokračovat s jejich sledováním a tudíž rozpoznáváním gest, či pohybovat s 3D scénou.

V této práci byly k rozpoznávání obličeje, očí a nosu použity Haar-kaskády z OpenCV knihovny. Haar-kaskády jsou kaskády klasifikátorů, které po přijetí obrazu posuzují, zda předložený obraz odpovídá parametricky hledanému objektu, reprezentovanému danou kaskádou. Postupně prochází obraz klasifikátory nižších, obecnějších úrovní, až po ty nejvyšší, detailnější úrovně. Pokud úspěšně projde obraz přes všechny tyto úrovně, pravděpodobně reprezentuje hledaný objekt. Pokud jakýkoliv z klasifikátorů v průběhu odmítne daný obraz, není nutné dále tento obraz zkoumat a je klasifikován jako obraz neobsahující daný objekt.

V tomto programu je hlavním úkolem rozpoznávače rozpoznat obě oči a nos, jelikož tyto body potřebujeme pro výpočet natočení hlavy, popřípadě pro rozpoznávání gest. Tudíž detekce hlavy je zde jen vedlejší úloha, která slouží pro urychlení detekce očí a nosu a pro jejich přesnější detekci. Hlava jako hledaný objekt je mnohem komplexnější a musí splňovat řádově více parametrů než nos či oči a proto program s větší přesností nalezne hlavu v obraze a omezí se nalezení objektů, které by jen připomínaly nalezený objekt, jinými slovy máme větší jistotu, že nalezený objekt, v našem případě hlava, je opravdu hlavou. Pokud již hlavu detekovanou máme, lze detekovat oči a nos mnohem rychleji, jelikož je detekujeme pouze v oblasti nalezené hlavy a také si můžeme být jistější, že detekované objekty jsou skutečně oči nebo nos. Přesto však při detekci očí dochází občas k detekci objektů na tváři člověka, které očima nejsou a to zejména v oblasti, kde je vrhán stín (například kolem nosu). Proto byl program obohacen o tak zvaný antropologický analyzátor, který má za úkol zjistit, jestli nalezená trojice oko-nos-oko je z hlediska antropologického možná.

## 4.1 Detektor

Jak již bylo zmíněno, detektor rozpoznává objekty ze vstupního obrazového toku. Při detekci objektu prohledává rozpoznávač celý vstupní obraz, a to postupně v několika měřítcích. V každém měřítku detektor pracuje s okénkem o různé velikosti, které postupně přikládá na celý obraz, posouvá jej a klasifikátory v dané kaskádě posuzují, zda se v daném okně objekt vyskytuje, či ne. Pokud je objekt nalezen, je umístění a velikost okna vrácena z detektoru a dále zpracovávána programem.

## 4.2 Antropologický analyzátor

Při detekci očí a nosu je generováno více kandidátů na oči a jeden kandidát nosu. Antropologický analyzátor má za úkol najít nejpravděpodobnější trojici „levé oko/nos/pravé oko“ a určit, zda tato trojice opravdu může představovat oči a nos člověka. Antropologický analyzátor pracuje v několika krocích:

- Sestaví dvojice očí z množiny nalezených očí, přičemž nemůže do dvojice vybrat dvě totožné oči (oči na stejné pozici).
- Porovná pozice očí a nosu. Pro správnou trojici uvažuje, že oči musí být umístěny výše než nos.
- Porovná vzdálenosti mezi očima a vzdálenosti každého oka od nosu a sestaví si pro každou takovouto trojici oko-nos-oko koeficient správnosti.
- Vybere trojici s nejlepším koeficientem a porovná, zda koeficient není příliš odchýlen od vzoru. Pokud ne, nalezená trojice je považována za správně detekované oči a nos.

Rovnice 4.2.1 vyjadřuje vztah vzdálenosti očí od sebe a vzdálenosti jednotlivých očí od nosu.

$$n = o \times K_{EnEe} \qquad \textit{Rovnice 4.2.1}$$

kde o je vzdálenost očí od sebe a n je vzdálenost oka od nosu a kde

$$K_{EnEe} = 1{,}0833 \qquad \textit{Rovnice 4.2.2}$$

je koeficient jejich poměru.

Vidíme, že koeficient poměru těchto vzdáleností je poměrně malý a že trojice oko-nos-oko tvoří téměř rovnostranný trojúhelník. Takto tomu ovšem je jen pro výchozí stav, kdy kamera je umístěna přímo před uživatelem a uživatel je na kameru natočen čelem a není vůči kameře nijak nakloněn či pootočen. Díky této skutečnosti musí uživatel při detekci být v této, pro tento program normované poloze. Jelikož uživatel by jen stěží byl schopen tato kritéria splnit naprosto přesně, jsou zde zavedeny koeficienty pro odchylky od normovaného stavu. Tyto odchylky byly nastaveny po sérii pokusů tak, aby program byl schopen detekovat hlavu i v mírných úhlech jejího natočení a zároveň aby nedocházelo k chybnému stanovení bodů oko-nos-oko v případě, že jedno z očí nebo nos byly detekovány chybně. V tabulce 4.2.1 jsou názorně uvedeny některé odchylky a v procentech vyjádřené případy chybné detekce a stanovení bodů oko-nos-oko.

| Koeficient α | Koeficient β | Maximální natočení γ | Chybná detekce | Zamítnutí správných trojic |
|---|---|---|---|---|
| 1,2 | 1,3 | 46,79° | 5% | 10% |
| 1,2 | 1,1 | 26,47° | 2% | 30% |
| 1,25 | 1,3 | 66,01° | 15% | 1% |

*Tab. 4.2.1: Vliv antropologických odchylek na chybnou detekci*

*Koeficient α: Koeficient maximálního poměru vzdáleností oko-nos/oko-oko*

*Koeficient β: Koeficient maximálního poměru vzdáleností 1.oko-nos/2.oko-nos*

*Maximální natočení γ: Maximální natočení hlavy doleva/doprava pro dané koeficienty, které program při detekci uzná. Vyšší považuje za chybu detekce.*

*Chybná detekce: Udává v procentech přibližný počet trojic, které byly považovány za skutečné oči a nos a přitom jimi nebyly.*

*Zamítnutí správných trojic: Přibližný počet zamítnutých správně detekovaných trojic v procentech*

Výpočet maximálního úhlu natočení:

Podle antropologického průměru platí pro ideální poměr vzdáleností *n* (oko-nos) ku *o* (oko-oko) při natočení 0° tento vztah:

$$n = o \times K_{EnEe} \qquad \textit{Rovnice 4.2.3}$$

kde

$$K_{EnEe} = 1{,}0833 \qquad \textit{Rovnice 4.2.4}$$

Dále vypočteme výšku *v* jako:

$$v = \sqrt{n^2 - \left(\frac{o}{2}\right)^2} \qquad \textit{Rovnice 4.2.5}$$

Nyní musíme spočítat vzdálenosti $n_1$ a $n_2$ (vzdálenosti jednotlivých očí od nosu), které simulují naklonění hlavy a tudíž nahrazují normovanou vzdálenost *n*. Začneme vyjádřením vzájemného poměru těchto délek:

$$n_1 = \beta \times n_2 \qquad \textit{Rovnice 4.2.6}$$

Trojúhelník oko-nos-oko již tedy nebude mít dvě strany *n* a třetí *o,* a tudíž výška *v* na stranu *o* nebude tuto stranu půlit, ale rozdělí ji na díly $o_1$ a $o_2$. Potom zřejmě platí:

$$o = o_1 + o_2$$ *Rovnice 4.2.7*

Dále pro takto rozdělený trojúhelník platí vztahy:

$$n_1 = \sqrt{{o_1}^2 + v^2}$$ *Rovnice 4.2.8*

$$n_2 = \sqrt{{o_2}^2 + v^2}$$ *Rovnice 4.2.9*

Z těchto vztahů sestavíme kvadratickou rovnici například pro $o_1$:

$$n_2 = \sqrt{(o - o_1)^2 + v^2}$$ *Rovnice 4.2.10*

$$\sqrt{{o_1}^2 + v^2} = \beta \times \sqrt{(o - o_1)^2 + v^2}$$ *Rovnice 4.2.11*

$${o_1}^2 + v^2 = \beta^2 \times (o - o_1)^2 + \beta^2 v^2$$ *Rovnice 4.2.12*

$$(1 - \beta^2) \times {o_1}^2 + \beta^2 \times 2 \times o_1 - \beta^2 \times v^2 + v^2 - \beta^2 \times o^2 = 0$$ *Rovnice 4.2.13*

Z této rovnice vyjdou dva výsledky pro $o_1$, jeden z nich je však větší než *o*, což pochopitelně nevyhovuje, protože $o_1$ je část z *o*. Pak tento výsledek dosadíme do vztahu:

$$d_x = \frac{o}{2} - o_1$$ *Rovnice 4.2.14*

$d_x$ představuje vychýlení nosu v ose *x* od pozice při natočení 0°. Nyní potřebujeme zjistit normovanou vzdálenost nosu od bodu otáčení v ose *z*.

$$z_n = o \times K_{ND}$$ *Rovnice 4.2.15*

kde $K_{ND}$ je poměr „výšky“ nosu k vzdálenosti mezi očima a jeho hodnota je:

$$K_{ND} = 0{,}3$$ *Rovnice 4.2.16*

Nyní již můžeme dosadit do zjednodušeného vztahu pro výpočet *yaw* úhlu:

$$yaw = arcsin\left(\frac{d_x}{z_n}\right)$$ *Rovnice 4.2.17*

Pro normované polohy zvolíme:

$$o = 1 \qquad \textit{Rovnice 4.2.18}$$

Takto vypočteme maximální úhel pro koeficient $\beta$, ale musíme ještě vypočíst maximální úhel pro koeficient $\alpha$ a vybrat nižší hodnotu. To je finální maximální hodnota natočení hlavy pro dvojici koeficientů $\alpha$ a $\beta$.

Výpočet maximálního úhlu pro koeficient $\alpha$:

Pro normované $o$ je $n_2$ pak:

$$n_2 = \alpha \times o \qquad \textit{Rovnice 4.2.19}$$

Výpočet pro $v$ již známe z předchozích rovnic. Pokračujeme takto:

$$o_2 = \sqrt{{n_2}^2 - v^2} \qquad \textit{Rovnice 4.2.20}$$

$$o_1 = o - o_2 \qquad \textit{Rovnice 4.2.21}$$

Pokud již známe $o_1$, dokončíme výpočet stejně jako pro odchylku $\beta$, to znamená vypočtené hodnoty dosadíme postupně do rovnic a spočítáme od rovnice 4.2.14 až 4.2.18.

V tomto programu je prioritou bezchybná detekce těchto tří bodů, proto byly po těchto pokusech vybrány koeficienty z 1. řádku tabulky 4.2.1. Koeficienty jsou v tomto případě dostatečně velké na to, aby uživatel, který usedne k PC a nevěnuje příliš pozornost tomu, aby natočil hlavu do přesné polohy, byl bez problému detekován a zároveň detekce, při níž dojde k mylnému označení objektů jako očí a nosu byla minimální. V praxi ovšem program není schopen detekovat hlavu při natočení až o 46,79°. Tato schopnost je omezena možnostmi natrénovaných kaskád klasifikátorů, které detekují hlavu správně přibližně do 10° natočení.

## 4.3 Manažer rozpoznávání

Manažer rozpoznávání se stará o celý proces rozpoznání očí a nosu. Vstupem manažeru je jediný obraz. Manažer tento obraz předá detektoru a sdělí mu, že má použít kaskády pro hledání lidské hlavy v obraze. Pokud detektor vrátí, že hlava nebyla v obraze nalezena, manažer s tímto obrazem dále již nic neprovádí. Pokud však byla nalezena hlava, převezme od detektoru její pozici a velikost a tento výřez z původního obrazu pošle postupně detektoru očí i detektoru nosu. Pokud detektor nosu vrátí nalezený nos a detektor očí vrátí aspoň dvě různé pozice nalezených očí, manažer pošle tyto pozice antropologickému analyzátoru. Pokud antropologický analyzátor vrátí trojici oko-

nos-oko, vrátí tuto trojici a její velikost i manažer rozpoznávání jako úspěšně detekované objekty očí a nosu.

# 5 Sledovač očí a nosu

Sledovač očí a nosu má za úkol nalezené oči a nos sledovat. Co znamená sledovat nalezené objekty? V podstatě jde o to, tyto objekty opět nalézt v dalších snímcích. Proč tedy používáme sledovač a nevyužijeme již existujícího rozpoznávače? Důvodů je několik. Asi nejzávažnější důvod je neschopnost rozpoznávače rozpoznat objekty, které jsou natočeny vůči kameře o větší než minimální úhel. Dalším podstatným důvodem je rychlost rozpoznávání, která oproti rychlosti nalezení sledovaných objektů je výrazně nižší. Také schopnost sledovače přizpůsobit se některým změnám světelných podmínek, pokud nejsou příliš náhlé, kdy například jedna polovina hlavy se díky natočení ocitne ve stínu, kde by rozpoznávač již nebyl schopen oko rozeznat, hovoří v jeho prospěch.

## 5.1 Jak sledovač pracuje?

Sledovač v tomto programu využívá základní princip metody Lucas-Kanade optického toku. Pro větší využitelnost byla tato metoda obohacena o pyramidový přístup. Co znamená pyramidový přístup? Body z předešlého obrazu se hledají v blízkém okolí, ve kterém byly minule nalezeny. Pokud v tomto okolí nalezeny nebudou, pravděpodobně se objekty pohnuly příliš rychle a budou za hranicemi tohoto okolí. Proto se okolí rozšíří a hledání se opakuje. Takto se okolí rozšiřuje do té doby, než se hledaný objekt nalezne, nebo dokud okolí nebude totožné s celým vstupním obrazem. Pokud hledaný objekt nebude nalezen ani v celém obraze, je možné, že z obrazu vystoupil, nebo jej již nedokážeme identifikovat. V takovém případě hovoříme o ztrátě objektu.

Pokud daný objekt nalezneme, uložíme si jeho polohu a také je nutné si uložit celý vstupní obraz, abychom při dalším kroku mohli dané objekty ze starého obrazu „vyříznout“ a použít k porovnávání s obrazem novým.

## 5.2 Kontrolor správného sledování

V průběhu sledování objektů se může stát, že se sledovač ztratí a začne sledovat úplně jiný objekt nebo část obrazu, než který by měl. Tento nepříznivý jev je třeba co nejdříve odhalit a napravit.

### 5.2.1 Jak odhalit ztrátu sledovače?

Odhalit ztrátu sledovače může být někdy velmi těžké až nemožné. Přesto hlavně ty velké omyly sledovače jde odhalit celkem brzo a spolehlivě. Když sledovač pouze sleduje místo objektu část obrazu (jiný objekt), který je jen nepatrně vzdálen od původního objektu, je těžké tento jev odhalit. Pokud ovšem sledovač „uteče“ od sledovaného objektu dále, stává se tento úkol snazším. Abychom odhalili mylné sledování, využíváme:

- Omezení natočení hlavy do různých úhlů: Pokud body oko-nos-oko „utečou“ od sebe natolik, že výpočet těchto úhlů překročí povolenou hranici, je jasné, že se sledovač ztratil. Při přesném sledovači by bylo dobré tyto úhly nastavit tak, aby byly co největší do doby, kdy jdou všechny tři objekty vidět na 2D obraze. Díky zvyšující se nepřesnosti sledovače při větších náklonech hlavy byly tyto úhly omezeny po řadě testů na 45° pro náklon hlavy doleva a doprava a pro náklon hlavy nahoru a dolů, a 60° pro natočení hlavy doleva a doprava.
- Také je nutné, aby sledovač byl považován za ztracený, pokud se přiblíží objekty okrajům obrazu, jelikož za okrajem je již není možné sledovat.
- Také pokud se objekty od sebe nepřirozeně vzdálí, tzn. pokud například jedno oko začne „utíkat“ nahoru, začne se měnit úhel naklonění hlavy. Úhel bude stále v normě, ale již zřejmě mylně. Na detekci tohoto stavu opět poslouží antropologická měřítka vzdáleností nosu od očí a očí mezi sebou. Pokud je hodnota mimo limity, sledovač se opět považuje za neplatný.

### 5.2.2 Jak tuto ztrátu napravit?

Co dělat při ztrátě sledovače? Nabízí se několik možností:

- Pokusit se o korekci sledovače a posunout jej zpátky na sledovaný objekt. Toto by mohlo být docíleno pomocí pozice ostatních sledovaných objektů a pomocí historie sledování, kde bychom mohli použít například Kalmanovy filtry (vysvětleno v [3]) pro odhad, kde by se daný objekt mohl nacházet. Tato varianta by byla ale zřejmě velice početně náročná a nezaručovala by stoprocentní úspěch. Nicméně by se sledovač zdánlivě neztratil a rozpoznávání pohybů a natočení hlavy by se pro uživatele jevilo plynulé bez jakéhokoliv přerušení.
- Vrátit se o pár sledovaných snímků nazpět a pokusit se sledovat daný objekt znova s trošku jinými parametry. Zde by však byla nutnost uchovávat si historii několika

snímků dozadu. Určitě by tato varianta byla početně náročná a zde bychom si již vůbec nemohli být jistí výsledkem. Pokud bychom ale považovali výsledek za správný, opět by nedocházelo ke ztrátě sledovače a s tím spojené přerušení detekce úhlů či gest.

- Nejjednodušším řešením je prohlásit sledovač za ztracený a proces detekce těchto bodů zopakovat. Nevýhodou je přerušení detekce gest a natočení hlavy a nutnost, aby se uživatel opět natočil přímo na kameru pro další detekci. Ale pouze u tohoto řešení si můžeme být jisti, že sledování bude nadále pokračovat tak, jak má. A proto i v této aplikaci je použito toto řešení.

## 5.3 Nedostatky sledovače

Jedním z hlavních nedostatků tohoto, ale i jiných sledovačů (trackerů), je neschopnost sledovat příliš rychlé pohyby kvůli velké změně osvětlení sledovaného objektu a potažmo i celé scény. Tento fakt však v této práci uvádím pouze z teoretického hlediska, jelikož v mém případě znemožnila detekci rychlých pohybů webkamera, která tyto pohyby rozmazává a sledovač pak již zcela pochopitelně nemůže sledovat něco, co se již jeví značně odlišně. Dále pak již z pochopitelných důvodů není tento sledovač schopen sledovat objekty, které již nelze vidět, jelikož je založen v podstatě na porovnávání 2D obrazových bodů. Tudíž není schopen sledovat oko, které je díky natočení hlavy zakryto nosem apod. Také při náhlých změnách osvětlení se sledovač může ztratit, jelikož objekt ze snímku na snímek najednou změní markantně své parametry jasu. Zde by například lépe obstál rozpoznávač, který objekty rozpoznává na základě parametrů více invariantních na změnu sytosti a jasu celého objektu.

# 6 Analyzátor natočení hlavy

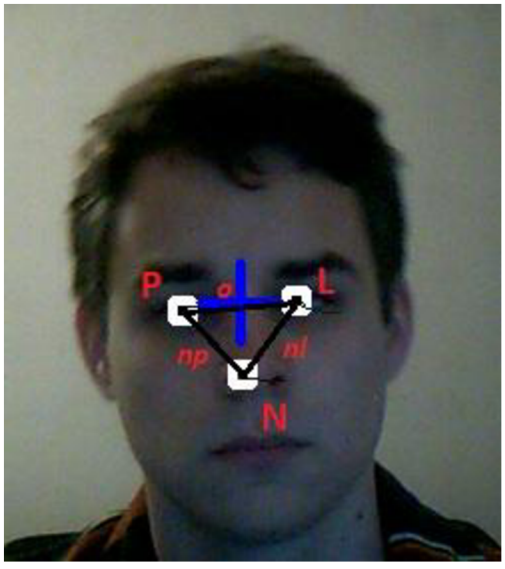
Analyzátor natočení hlavy má na svém vstupu pozice tří bodů, reprezentující pozice očí a nosu ve 2D vstupním obraze a na výstup analyzátoru má za úkol dodat tři hodnoty úhlů natočení hlavy uživatele.

Práci analyzátoru lze popsat v těchto bodech:

- Výpočet vzdáleností ve 2D obraze
- Výpočet bodu rotace
- Podle bodů v prostoru analyzátor vypočte úhly *roll* a *yaw*
- Pomocí cyklu numericky vypočte s krokem 1° úhel *pitch*, který nejlépe odpovídá hledanému úhlu náklonu hlavy nahoru/dolů

## 6.1 Výpočet vzdáleností ve 2D obraze

Výpočet vzdáleností mezi očima a mezi jednotlivými oky a nosem je poměrně triviální záležitost. Na obrázku 6.1.1 vidíme detekované oči a nos v 2D obraze.




*OBR. 6.1.1: Detekované oči a nos v obraze a znázornění jejich vzdáleností*

Některé proměnné získáme ze snímaného obrazu. Jsou to pozice očí a nosu ve 2D prostoru, body *L[Lx, Ly]* (levé oko), *P[Px, Py]* (pravé oko), *N[Nx, Ny]* (nos).

Výpočet vzdáleností očí od nosu a vzdálenost mezi očima vypočteme podle následujících vztahů:

Vzdálenost mezi očima:

$$o = \sqrt{(L_x - P_x)^2 + (L_y - P_y)^2} \qquad \textit{Rovnice 6.1.1}$$

Vzdálenost levého oka od nosu $n_l$ je definována takto:

$$n_l = \sqrt{(L_x - N_x)^2 + (L_y - N_y)^2} \qquad \textit{Rovnice 6.1.2}$$

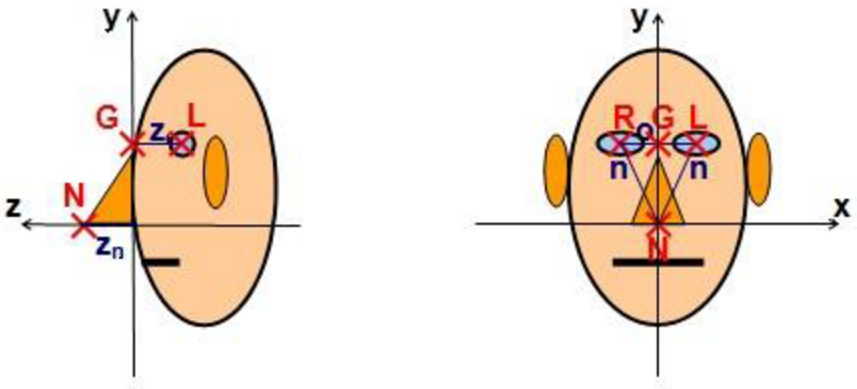
Vzdálenost pravého oka od nosu $n_p$ je počítána jako:

$$n_p = \sqrt{(P_x - N_x)^2 + (P_y - N_y)^2} \qquad \textit{Rovnice 6.1.3}$$

Tyto tři vzdálenosti jsou jediné důležité proměnné spolu s umístěním bodů očí a nosu, které ze vstupního obrazu lze vyextrahovat. Zbytek proměnných je potřeba vypočítat.

## 6.2 Výpočet bodů rotace

V této práci reprezentují hlavu tři body v troj-rozměrném prostoru, a to: levé oko, pravé oko a špička nosu. Jelikož však výstup z kamery je reprezentován dvoj-rozměrným obrazem, musíme tyto body z dvoj-rozměrného prostoru transformovat do troj-rozměrného prostoru. Zvolil jsem tedy, že oči leží v rovinně kolmé na rovinu *xy* tak, že každé oko je stejně vzdáleno od osy *y*. Nos leží na ose *z*. Ilustraci posazení těchto tří bodů znázorňuje obrázek 6.2.1.




*OBR. 6.2.1: Znázornění některých vzdáleností a bodů na hlavě*

Aplikace ovšem všechny body a vztahy ihned přepočítává do 2D prostoru a také v 2D prostoru jsou všechny proměnné ukládány, tudíž se jedná pouze o „pseudo“ 3D prostor. Pro lepší

představu Vám zde představuji myšlenky v 3D prostoru, vztahy a rovnice jsou již pak počítány ve 2D prostoru.

Bod rotace $G[G_x,G_y]$ je umístěn mezi očima v rovinně kolmé na rovinu *xy* a umístěn ve směru osy *z* mezi nosem a středem mezi očima. Souřadnice bodu G jsou pak vyjádřeny triviálními vztahy:

$$G_x = \frac{L_x + P_x}{2}$$ *Rovnice 6.2.1*

$$G_y = \frac{L_y + P_y}{2}$$ *Rovnice 6.2.2*

Jak již bylo řečeno, body jsou uloženy či přepočítávány do 2D prostoru. Pokud však uvažujeme 3D prostor, pak *z*-ová souřadnice bodu *G* by byla 0.

## 6.3 Výpočet úhlů natočení

Hlava může být natočena ve všech třech osách. Rozlišujeme natočení hlavy podle osy *x* „pitch“, podle osy *y* „yaw“ a podle osy *z* „roll“. Pro výstižnost jsem se rozhodl používat tyto tři anglické termíny, které na rozdíl od českých ekvivalentů jsou snáze rozlišitelné.

Dříve než si ukážeme, jak jednotlivé úhly vypočítat, je potřeba si stanovit několik proměnných, které k výpočtu budeme potřebovat. Z obrázku 6.2.1 by mělo být patrné, co které proměnné reprezentují.

Je nutné znát některé koeficienty. Tyto koeficienty byly odvozeny a vypočítány na základě průměrných lidských měřítek.

Koeficient poměru „výšky“ nosu od bodu rotace G k vzdálenosti mezi očima je roven:

$$K_{ND} = 0{,}3$$ *Rovnice 6.3.1*

Koeficient poměru „hloubky“ očí od bodu G k vzdálenosti mezi očima je roven:

$$K_{ED} = 0{,}2786$$ *Rovnice 6.3.2*

Koeficient poměru vzdálenosti středu očí a nosu k vzdálenosti mezi očima je roven:

$$K_{CnEe} = 0{,}5857$$ *Rovnice 6.3.3*

Tyto body a koeficienty jsou základem pro výpočet proměnných, které budeme potřebovat pro výpočty úhlů natočení hlavy. Výpočty těchto proměnných vysvětlují následující rovnice:

Vzdálenost mezi očima *o* je vypočtena jako:

$$o = \sqrt{(L_x - P_x)^2 + (L_y - P_y)^2}$$ *Rovnice 6.3.4*

Nyní jsme schopni vyjádřit a spočítat úhel natočení hlavy *roll*:

$$roll = arcsin\frac{L_y - R_y}{o}$$ *Rovnice 6.3.5*

Nyní je potřeba vypočíst „výšku" nosu $z_n$ jako:

$$z_n = o \cdot K_{ND}$$ *Rovnice 6.3.6*

Úhel natočení *yaw* můžeme spočítat takto:

$$yaw = arcsin\frac{N_x + \sin roll \cdot (-o \cdot K_{CnEe})}{z_n}$$ *Rovnice 6.3.7*

Pak je nutné vypočítat si „hloubku" očí jako:

$$z_e = o \cdot K_{ED}$$ *Rovnice 6.3.8*

Následující rovnice pro výpočet úhlu *pitch* nelze vypočíst analyticky a proto je nutné použít numerické metody:

$$0 = \left|\cos pitch \cdot o \cdot K_{CnEe} - \sin pitch \cdot (z_n + z_e) - \sqrt{{N_x}^2 + {N_y}^2}\right|$$ *Rovnice 6.3.9*

Je pochopitelné, že použitím koeficientů určujících standardní poměry vzdáleností bodů na lidské tváři nebude výpočet natočení hlavy do daných úhlů zcela přesný, zvláště pak pro atypické lidské tváře. Nicméně pro většinu lidí bude tento výpočet, za předpokladu zcela přesného rozpoznávače a sledovače, velice přesný. Přesto se domnívám, že i u lidí s netypickými rysy obličeje bude přesnost výpočtů uspokojivá.

# 7 Pohybový analyzátor

Pohybový analyzátor slouží k rozeznání pohybů učiněných uživatelovou hlavou. Z posloupnosti obrázků, respektive souřadnic bodu nosu, určí pohybový analyzátor, jaký pohyb uživatel provedl, zda pohnul nosem doleva, doprava, nahoru, dolů apod. Pohybový analyzátor na svůj vstup dostává pozici bodu, v této aplikaci je to pozice nosu, v každém snímku. Jeho úkolem je analyzovat řadu bodů a z nich vyvodit směr pohybu, jeho délku, popřípadě další parametry. Informace o rozpoznaném pohybu, po jeho ukončení, jsou výstupem analyzátoru.

Pohybový analyzátor jsem rozdělil na tři logické celky:

- Správce historie
- Analyzátor částečných pohybů
- Analyzátor pohybů

Tyto tři celky na sebe bezprostředně navazují a tvoří tedy souhrnně pohybový analyzátor. Analyzátor pohybů je jeho podstatou, ale bez ukládané historie pozice bodů ani bez analýzy částečných pohybů by nemohl pracovat.

## 7.1 Správce historie

Abychom mohli analyzovat nějaký pohyb bodu z místa A do místa B, je nutné, abychom věděli, přes které pozice sledovaný bod prošel. Je nutné proto tyto informace ukládat a o toto se stará správce historie. Základními funkcemi správce historie jsou:

- Uložit bod do historie tak, aby body na sebe navazovaly tak, jak chronologicky mají
- Poskytnout požadovaný počet bodů z historie
- Management historie

Pro ukládání historie jsem použil způsob cyklického pole, a to z důvodu rychlosti. Cyklické pole není nic jiného, než obyčejné pole jak jej známe z jazyka C, do kterého se hodnoty po sobě jdoucí ukládají na vyšší indexy. Pokud ovšem by se měla nová hodnota uložit za hranice pole, uloží se na jeho začátek. Při takovémto přístupu je nutné si vždy pamatovat, na kterém indexu pole je uložen bod nejstarší a kolik bodů je v poli celkem uloženo. Tuto logiku má na starost logický celek Management historie.

## 7.2 Analyzátor částečných pohybů

Tento analyzátor využívá funkcí Správce historie. Pro svou práci potřebuje znát vždy 3 poslední pozice, kterými daný bod prošel a tyto pozice získá právě ze Správce historie. Proč zrovna 3 poslední pozice bude vysvětleno níže. Pokud historie neobsahuje ani tři prvky, analyzátor svou práci nezahájí a vyčká, až budou tři pozice připraveny. Na výstupu tohoto analyzátoru je tak zvaný částečný pohyb, který dál využívá Analyzátor pohybů.

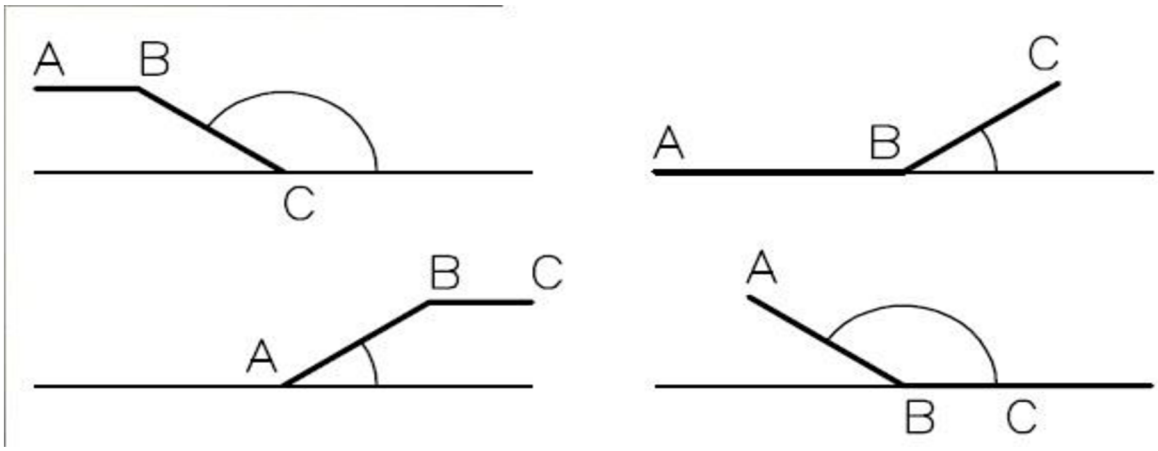
### 7.2.1 Co je to částečný pohyb?

Pro zjednodušení rozpoznávání pohybů jako celků byl zaveden částečný pohyb a jeho analýza. Částečný pohyb je složen z minima bodů, které jsou schopny vyjádřit všechny jeho parametry.

Parametry částečného pohybu, které nás zajímají, jsou: směr pohybu, jeho velikost a konvexnost/konkávnost pohybu.

### 7.2.2 Proč je potřeba minimálně tří bodů?

O tom rozhoduje právě posledně zmíněný parametr částečného pohybu a tím je konvexnost či konkávnost. Z jednoho bodu nelze určit ani směr ani velikost pohybu, respektive směr není žádný a velikost pohybu je nulová. Ze dvou po sobě jdoucích bodů již můžeme určit velikost a směr pohybu, ale zda je konvexní či konkávní zjistit nemůžeme. Na to je zapotřebí třetí bod, abychom viděli, jestli se mění úhel natočení křivky, kterou body reprezentují. Pouze ze dvou bodů bychom zjistili pouze počáteční úhel křivky, ale nevěděli bychom, jak bude zakřivení pokračovat. Při snižování, respektive zvyšování úhlu se jedná o křivku konkávní, respektive konvexní. Na následujícím obrázku je demonstrace křivek konkávních a konvexních a úhlů, které se mění. Křivka vždy začíná v bodě *A* a končí v bodě *C*.



*OBR. 7.2.2.1: Změna úhlů a vlastnosti křivek*

*Vlevo nahoře: Konkávní a úhel se od AB k BC zvětšil*

*Vpravo nahoře: Konvexní a úhel se od AB k BC zvětšil*

*Vlevo dole: Konkávní a úhel se od AB k BC snížil*

*Vpravo dole: Konvexní a úhel se od AB k BC snížil*

Jak je vidět na obrázku 7.2.2.1, je nutné zkoumat zvětšování či zmenšování úhlů v závislosti na směru pohybu.

### 7.2.3 Které částečné pohyby rozlišujeme?

Částečné pohyby rozlišujeme pouze podle jejich směru a konvexnosti/konkávnosti. Parametr délky pohybů je zde navíc a je využit až dále.

Rozlišujeme tedy tyto pohyby:

- Žádný pohyb
- Přímočaré pohyby ve směru os: doleva, doprava, nahoru a dolů
- Přímočaré šikmé: doprava nahoru, doprava dolů, doleva dolů a doleva nahoru
- Křivočaré: Konvexní doprava nahoru, konvexní doprava dolů, konvexní doleva dolů, konvexní doleva nahoru a jejich konkávní varianty
- Break: To je „pohyb“, který nebyl zařazen ani do jedné z výše uvedených kategorií. Jedná se například o pohyb složený z úseček, kdy jedna úsečka, z bodu *A* do bodu *B*, stoupá a druhá, z bodu *B* do bodu *C*, klesá.

Pohyb „Break“ vlastně postihuje pohyby, které byly sejmuty špatně, a to například v důsledku „zachvění“ sledovače. Je možno říci, že pohybem „Break“ můžeme vlastně ztratit i kousek validní pohybové informace, což je pravda, protože pohyb „Break“ může být nositelem správné informace například v případě, kdy opisujeme elipsoid. V takovém případě na čtyřech místech v elipsoidu nastane detekce pohybu „Break“, a to při přechodu opisu elipsoidu mezi kvadranty, přitom se jedná o zcela správnou detekci. Tento jev je ovšem žádoucí a pohyb „Break“ je dále využíván pohybovým analyzátorem.

### 7.2.4 Tolerance v analýze

V ideálním případě by pohyby například nahoru či dolů měly svou *y*-ovou souřadnici stejnou na začátku i konci pohybu. Jelikož však je to člověk, který před počítačem sedí, nikdy takto přesné pohyby udělat nedokáže a koneckonců sledovač tak přesně detekovat také nesvede. Proto bylo nutné

zavést jakousi toleranci výchylky, kdy je pohyb stále považován za pohyb kolmý na osu *y*. Obdobně toto platí i pro osu *x*. Následující tabulka znázorňuje provedené testy různých konstant.

| $T_{orthogonal}$ | Chybné rozpoznání pohybu kolmého na osu | Chybějící rozpoznání pohybu kolmého na osu |
|---|---|---|
| 0.05 | 1% | 7% |
| 0.1 | 10% | 5% |
| 0.02 | 0% | 30% |

*Tab. 7.2.4.1: Procentuální chybovost při určování pohybů kolmých na souřadnicové osy*

Po řadě testů, jež některé znázorňuje tabulk7.2.4.1, jsem se rozhodl použít toleranci $T_{orthogonal}$ s hodnotou 0.05, což znamená, že pohyb bude zařazen jako kolmý na osy, pokud bude vychýlen od kolmého směru maximálně o 5% své délky vyjádřené v obrazových bodech.

Dále je pak také potřeba zavést konstantu tolerance přímého pohybu. Jedná se zde o rozlišení pohybů přímočarých a křivočarých. Konstanta $T_{direct}$ tedy znamená, o kolik stupňů (radiánů) se od sebe mohou lišit úseky *AB* a *BC* částečného pohybu za každých *x* pixelů jejich společné délky.

| $T_{direct}$ [rad/pixel] ($1° \approx 0.017$rad) | Chybné rozpoznání pohybu přímého | Chybějící rozpoznání pohybu přímého |
|---|---|---|
| 0.017/10 | 5% | 10% |
| 0.034/10 | 15% | 5% |
| 0.017/20 | 2% | 20% |

*Tab. 7.2.4.2: Procentuální chybovost při určování pohybů přímých a křivých*

Po těchto testech vyplynulo, že nejvhodnější bude zavést toleranci výchylky $T_{direct}$ rovnu 1° na 10 obrazových bodů. Nabízí v této situaci optimální poměr mezi mylně rozeznanými pohyby.

# 7.3 Analyzátor pohybů

Název tohoto celku by mohl budit dojem, že se jedná o velmi podobnou věc, jakou je Analyzátor částečných pohybů. Není to však tak úplně pravda. Analyzátor pohybů na svém vstupu očekává právě výstup z Analyzátoru částečných pohybů, tudíž částečný pohyb. Úkolem Analyzátoru pohybu je tyto částečné pohyby serializovat do spojitých pohybů stejné orientace, popřípadě přehlédnout pohyby, které jsou velmi malé, pravděpodobně způsobené nepřesným sledováním bodu v obraze.

### 7.3.1 Které pohyby rozeznáváme?

Odpověď je jednoduchá. Rozeznáváme v podstatě totožné pohyby, které jsme rozeznávali v případě částečných pohybů. Výjimku tvoří jen typ pohybu „Break“, pro který již zde nemáme uplatnění. Tento pohyb se uplatnil u částečných pohybů, aby podle něj mohl Pohybový analyzátor pohyby například oddělovat, ale při práci Analyzátoru pohybů už jeho význam zaniká a dál není uplatnitelný, jak vyplývá z vysvětlení významu „Break“ částečného pohybu v kapitole 7.3.3.

### 7.3.2 Parametry pohybů

U pohybů nás zajímá několik parametrů. Samozřejmě nejdůležitějšími jsou orientace pohybu a jeho délka, přičemž orientací rozumíme souhrnný název pro směr pohybu a jeho zakřivení (konvexnost/konkávnost). Také nás mohou zajímat souřadnice začátku a konce pohybu v obraze, relativní rychlost pohybu apod. V tomto případě však ukládáme pouze začáteční a koncovou pozici, délku a orientaci pohybu.

### 7.3.3 Význam částečného pohybu „Break“ a princip analyzátoru

Částečný pohyb „Break“ má v podstatě dvojí opodstatnění. Kdy jej program detekuje již víme z kapitoly 7.2.3, proč je ovšem pro Pohybový analyzátor důležitý bude vysvětleno zde. V ideálním případě by se částečný pohyb „Break“ detekoval pouze při skutečné změně orientace pohybu. V tomto případě se ovšem detekuje i při změně orientace pohybu, která nebyla ve skutečnosti provedena, ale vznikla z důvodu drobného „záchvěvu“ detektoru, kdy sledovač na chvíli detekoval objekt, v našem případě nos, o několik málo obrazových bodů vedle, než ve skutečnosti měl. Ideálně by tedy „Break“ v posloupnosti detekovaných částečných pohybů znamenal, že se jedná o změnu orientace pohybu a doposud analyzovaný pohyb by se tedy uložil jako výsledný pohyb, spočítala by se jeho délka jako součet částečných pohybů stejné orientace po sobě a stanovil by se jeho začáteční a koncový bod podle prvního a posledního částečného pohybu v posloupnosti, popřípadě spočítala jeho průměrná rychlost. Reálně musíme ovšem uvažovat, že „Break“ se může vyskytnout i uprostřed analyzovaného pohybu stejné orientace, kvůli nepřesnosti sledovače. Proto je nutné zavést konstantu $K_{ignore}$, která určuje, kolik částečných pohybů se může maximálně objevit uprostřed analyzovaného pohybu a pohyb bude přitom brán jako validní. V této souvislosti je nutno si i stanovit, kolik částečných pohybů stejné orientace může minimálně tvořit pohyb, protože „záchvěv“ ze sledovače může kromě detekce „Break“, detekovat i jiný částečný pohyb, který je mylný. Tuto konstantu pojmenujeme $K_{valid}$. Pokud tedy analyzovaný pohyb přeruší alespoň $K_{ignore}$ částečných pohybů s jinou orientací, než je orientace právě analyzovaného pohybu, analyzovaný pohyb a jeho charakteristiky se

uloží, stejně tak, když právě analyzovaný pohyb přeruší alespoň $K_{valid}$ částečných pohybů po sobě jdoucích. S těmito konstantami jsem provedl několik testů a některé hodnoty jsou tedy uvedeny v tabulce 7.3.3.1.

| $K_{ignore}$ | $K_{valid}$ | Chybná detekce | Chybějící detekce |
|---|---|---|---|
| 3 | 2 | 5% | 5% |
| 2 | 2 | 7% | 10% |
| 1 | 3 | 2% | 50% |
| 0 | 1 | 15% | 0% |

*Tab. 7.3.3.1: Procentuální chybovost při detekci pohybů*

Sloupec „Chybná detekce“ v tabulce 7.3.3.1 znamená, v přibližně kolika procentech případů analyzátor vyhodnotil nějaký pohyb, který se ve skutečnosti neudál. Sloupec „Chybějící detekce“ pak v procentech uvádí, kolik přibližně pohybů, které se skutečně udály, analyzátor „přehlédl“. V programu je nakonec používána kombinace konstant z 1. řádku tabulky 7.3.3.1, jelikož poskytují únosnou míru chybné a chybějící detekce. Poslední řádek tabulky 7.3.3.1 zde byl představen pouze jako ilustrativní příklad s předem odhadnutelnými výsledky. Tato kombinace konstant by byla vhodná pro bezchybný sledovač a pro přesné pohyby uživatele, ale v praxi jak je vidět je tato kombinace zcela nevhodná.

### 7.3.4 Činnost analyzátoru pohybů

Analyzátor pohybů potřebuje ke své analýze posloupnost částečných pohybů, které byly rozpoznány. Z této potřeby vzniká také potřeba tyto částečné pohyby ukládat. Ukládány jsou do cyklického pole, principálně stejného, jakým je pole pro ukládání historie pohybů, což bylo vysvětleno v kapitole 7.1.

Původně byla implementace tohoto analyzátoru udělaná velice jednoduše, a to tak, že v každém cyklu detekce a sledování objektu se procházely všechny dosud detekované částečné pohyby a tato posloupnost byla prohledávána pro výskyt celistvých pohybů. Toto řešení ovšem nedosahovalo příliš uspokojující rychlosti, když se detekovaly velice dlouhé pohyby. Proto jsem optimalizoval tento proces následujícím způsobem: Rozpoznávaný pohyb každým příchozím částečným pohybem, pokud byly shodující se orientace, aktualizoval své parametry délky pohybu, koncového bodu pohybu a popřípadě rychlosti. Vždy, když analyzátor rozhodne, že daný pohyb již skončil, smaže historii ukládaných částečných pohybů od koncového pohybu po začátek historie.

V následujícím cyklu analyzuje toto pole od začátku celé a pak zase v každém dalším cyklu jen právě příchozí částečný pohyb.

# 8 Rozpoznávač gest

Nyní se dostáváme ke koncové části tohoto projektu, a to je rozpoznání gest z pohybů, které jsme analyzovali. Již máme za sebou rozpoznání hlavy, očí a nosu, jejich sledování a rozpoznávání pohybů. Nyní potřebujeme posloupnosti pohybů přidat nějaký význam a tyto posloupnosti hledat.

V příloze (2) je popsán formát gest a jak daná gesta do souboru přidat.

## 8.1 Gesta

Lidská gesta mohou být velice komplexní, či naopak jednoduchá. Význam gest se může lišit ve své velikosti, rychlosti provedení a dalších aspektech. Zatímco člověk je schopen i mírně poupravená gesta rozlišovat jako gesta původní, či odvozovat jejich význam z kontextu či dané situace, počítač tuto schopnost nemá. Jen stěží bychom byli schopni v této aplikaci význam gest hádat z kontextu. Proto tato aplikace nabízí jen rozpoznání, které gesto člověk provedl, ne však jeho sémantický význam. Stejně provedená gesta jiných velikostí, popřípadě i rychlostí, však mohou mít odlišný význam. Tento program však pouze rozlišuje gesta jiných velikostí. Co tedy v rámci programu rozumíme pod pojmem gesto?

### 8.1.1 Definice gesta

Pro účely aplikace budeme pod pojmem gesto rozumět přesně danou posloupnost pohybů, kde každý pohyb je definovaný orientací a délkou. Celé gesto pak je definované kromě posloupnosti pohybů také dvěma tolerancemi, a to tolerancí lokální a tolerancí globální.

### 8.1.2 Tolerance lokální

Jedná se o toleranci délky daného pohybu uvnitř gesta k jeho délce v normovaném tvaru gesta. Například pokud pohyb má normovanou velikost v daném gestu 100 a lokální tolerance gesta je 1.2, je možné, aby tento pohyb náležel gestu, když bude mít délku mezi 80 a 120. Není však zaručeno, že s délkou například 110 bude pohyb tomuto gestu opravdu připsán. Záleží totiž na celkovém velikostním koeficientu rozpoznávaného gesta. Lokální tolerance tedy určuje maximální možné zvětšení či zmenšení jednotlivých pohybů v gestu a potažmo i zvětšení nebo zmenšení celého gesta.

### 8.1.3 Tolerance globální

Globální tolerance je tolerancí daných pohybů mezi sebou. U každého pohybu daného gesta se vypočítá, o kolik se daný pohyb liší velikostně od normované velikosti pohybu daného gesta. Tento koeficient je pak porovnáván s koeficienty ostatních pohybů gesta totožně vypočtenými. Pokud se alespoň dva tyto koeficienty pohybů v daném gestu od sebe liší více než o globální toleranční koeficient gesta, netvoří tyto pohyby dané gesto.

## 8.2 Princip činnosti rozpoznávače gest

Rozpoznávač gest v každém cyklu hlavní smyčky programu kontroluje, zda nebyl rozeznán nový pohyb. Pokud ne, nemá v tomto cyklu nic na práci a je neaktivní. Pokud však byl nový pohyb rozpoznán, musí jej analyzovat.

Ještě než vysvětlíme, jak rozpoznávač postupuje, je nutné si říct o dalších celcích rozpoznávače gest, které jsou k tomuto zapotřebí.

### 8.2.1 Historie pohybů

Stejně tak jako při rozpoznávání pohybů potřebujeme ukládat kolekci částečných pohybů, i zde je nutností uchovávat posloupnost rozpoznaných pohybů, abychom byli na základě této posloupnosti schopni rozeznávat gesta. I zde bylo pro ukládání použito cyklické pole. Zde však je možno určit přesně velikost takovéhoto pole, a to tak, že bude mít kapacitu rovnu nejdelšímu gestu, které jsme schopni rozeznat (které je načteno ze souboru gest).

### 8.2.2 Seznam rozpoznávaných gest

Je to seznam gest, které chceme, aby program rozpoznával. U každého gesta je uložen jeho název, počet pohybů, ze kterých se dané gesto skládá, a pak pole těchto pohybů. Zpravidla tento seznam načteme ze souboru hned po spuštění aplikace a jeho odkaz předáme rozpoznávači gest při jeho inicializaci.

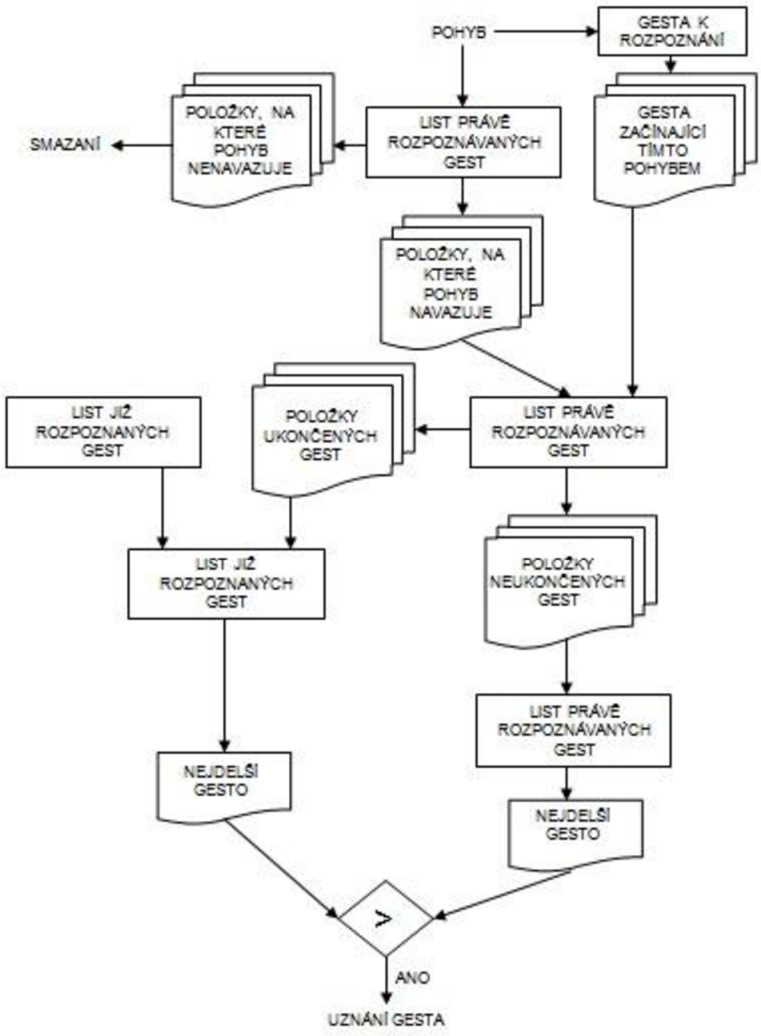
### 8.2.3 List právě rozpoznávaných gest

Jedná se o list, ve kterém budou gesta, která jsou zatím jen z části rozpoznaná. Jedná se o obousměrně vázaný list. Každá položka listu bude kromě reference na předešlou položku a následující položku obsahovat i referenci do Seznamu rozpoznávaných gest na gesto, které je kandidátem na rozpoznání. Dále pak bude obsahovat číslo, které bude znamenat, kolikátý pohyb v daném gestu byl již rozpoznán.

### 8.2.4 List již rozpoznaných gest

Tento list je podobný Listu právě rozpoznávaných gest. Každá položka listu má také odkazy na předešlou a následující položku, odkaz na gesto a také číslo, které značí také, kolik pohybů bylo rozpoznáno, a v tomto případě to tedy znamená, jak dlouhé rozpoznávané gesto bylo. Proč je tento list nutností, bude vysvětleno níže, při vysvětlování činnosti rozpoznávače.

Na obrázku 8.2.4.1 je názorně ukázáno, jak postupuje rozpoznávač gest při obdržení nového pohybu.




*OBR.8.2.4.1: Postup rozpoznávače gest*

- Nejdříve nový pohyb zkontroluje s Listem právě rozpoznávaných gest, zda daný pohyb nenavazuje na některé z gest. Gesta, kterým tento pohyb nevyhovuje buď svou orientací, nebo svými rozměry, jsou smazána z listu. Naopak u gest, kterým vyhovuje nový pohyb jak orientací, tak i velikostí (podle tolerancí gesta), se zvýší číslo počtu rozpoznaných pohybů o 1

a pokud již jsou tato gesta kompletní (počet pohybů v rozpoznávaném gestu se rovná počtu pohybů v předloze gesta), přesune se toto gesto do Listu již rozpoznaných gest.

- Nyní se zkontroluje, zda nový pohyb není potenciálně začátkem nového gesta, to znamená, jestli by daný pohyb nemohl být prvním pohybem v posloupnosti některého z gest v Seznamu gest k rozpoznání. Pokud ano, je dané gesto přidáno do Listu právě rozpoznávaných gest na jeho konec a číslo počtu pohybů je nastaveno na 1.
- Dále se projde List právě rozpoznávaných gest a zjistí se nejvyšší prozatímní délka rozpoznávaných gest. V listu již rozpoznaných gest se podíváme, které gesto je nejdelší. Tyto dvě hodnoty porovnáme. Pokud existuje v Listu právě rozpoznávaných gest gesto, které má prozatímní délku stejnou, popřípadě vyšší, než nejdelší gesto v Listu již rozpoznaných gest, nepodnikáme zatím žádné další kroky. Pokud se ovšem zjistí, že ani jedna z prozatímních délek gest z Listu právě rozpoznaných gest není aspoň tak dlouhá, jako nejdelší z gest v Listu již rozpoznaných gest, je toto nejdelší rozpoznané gesto uznáno jako gesto, které uživatel udělal. Následují tyto akce:
    - Z Listu již rozpoznaných gest se vymaže uznané gesto.
    - Z Listu již rozpoznaných gest i z Listu právě rozpoznávaných gest se vymažou všechna gesta, která začala v posloupnosti pohybů v úseku, který náleží právě rozpoznanému gestu, nebo před tímto úsekem.
    - Historie pohybů bude smazána od začátku až po konec právě uznaného gesta (koncový pohyb gesta v historii pohybů).

Proč bylo nutné již rozpoznané gesta ukládat do nějakého listu, když mohly být jednoduše uznány a pokračovalo by se v rozpoznávání dále? Důvod je následující. Představme si, že jedno gesto je složeno z pohybů například nahoru a dolů (pro zjednodušení nebudeme uvádět délky pohybů) a další gesto je složeno z pohybů nahoru, dolů a doleva. Možná již vidíme, co by se stalo, pokud by se gesta uznávala hned, jak by byla dokončena. V tomto případě by se po detekci pohybu nahoru uložila obě tato gesta do Listu právě rozpoznávaných gest s číslem velikosti 1. Pak by přišel pohyb dolů a obě tato gesta by se v tomto listu o jedničku zvětšily. Jenže 1. gesto by bylo již dokončeno, uznalo by se a tím pádem by se smazalo gesto 2. Kdybychom tedy takto postupovali, na 2. gesto by nikdy nedošlo. Proto je nutné, abychom se nejdříve podívali, jestli náhodou nedetekujeme ještě jiné, potenciálně větší gesto, které již rozpoznaná gesta může obsahovat. V takovém případě musí mít delší a komplexnější gesto přednost před gesty, z kterých se skládá, jinak by nikdy nemohlo být rozpoznáno.

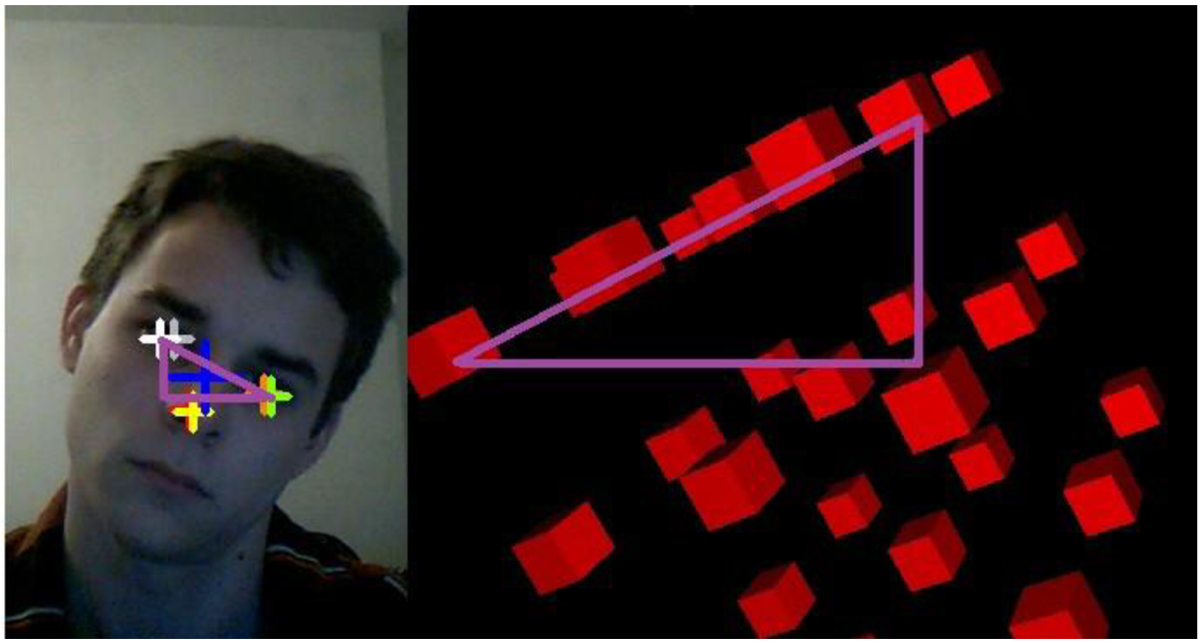
# 9 3D scéna

Tento celek jsem nezařadil do přehledu logických celků aplikace, jelikož se jedná v podstatě o samostatnou aplikaci, která slouží pouze pro demonstraci výpočtů natočení hlavy. Zde bych ještě jednou rád poděkoval Tomáši Ižákovi, který tuto jednoduchou scénu vytvořil.

Se scénou komunikuje aplikace pomocí zasílání zpráv. Posílá ji tři reálné hodnoty úhlů *pitch*, *roll* a *yaw* v radiánech. Scéna se podle těchto tří hodnot natáčí uživateli, takže uživatel natáčením hlavy tuto scénu ovládá.

Aby 3D scéna správně fungovala, je třeba mít nainstalovánu grafickou knihovnu GLUT a také je třeba spustit 3D scénu dříve než samotnou aplikaci rozpoznání.

Na obrázku 9.1 je demonstrace natočení scény pomocí naklonění hlavy. Můžeme si všimnout, že úhel natočení scény je totožný s úhlem natočení hlavy.



*Obr. 9.1: Demonstrace natočení 3D scény pomocí natočení hlavy*

# 10 Řízení aplikace

Již známe všechny součásti aplikace pro rozpoznání natočení hlavy a k rozpoznání jednoduchých gest. Nyní vysvětlíme, jak jsou tyto prvky propojeny a jak probíhá celý proces řízení aplikace. Úroveň abstrakce byla zvolena tak, aby byl pochopen princip činnosti. Stručně lze postup popsat takto:

- Spuštění aplikace 3D scény (volitelné, pouze pro vizuální demonstraci výpočtů úhlů natočení hlavy)
- Spuštění aplikace
- Načtení souborů natrénovaných Haar-kaskád klasifikátorů
- Načtení souboru gest k rozpoznání
- Hlavní smyčka
    - Pokud nyní nejsou sledovány objekty:
        - Pokud byla v předešlém snímku detekovaná hlava, detekuje se hlava v přilehlém okolí, jinak se hlava detekuje z celého snímku
        - Pokud nebyla hlava detekována, vrátí se řízení zpět na začátek hlavní smyčky
        - Ve výřezu obrázku reprezentující nalezenou hlavu jsou detekovány oči a nos
        - Pokud nebyly nalezeny alespoň dvě oči, či nebyl nalezen nos, řízení se vrací na začátek hlavní smyčky
        - Nalezení kandidáti očí a nosu jsou poslány Antropologickému analyzátoru k analýze
        - Pokud Antropologický analyzátor nevrátí platnou trojici, řízení je vráceno na začátek hlavní smyčky
        - Pokud byla vybrána platná trojice, jsou dále sledovány objekty
    - Pokud jsou sledovány objekty:
        - Nalezení nových pozic bodů pro oči a nos

        - Nové pozice předáme Kontroloru správného sledování. Pokud kontrolor určí, že se sledovač ztratil, nastaví aplikace, že nejsou sledovány objekty a vrátí řízení na začátek hlavní smyčky
        - Nové pozice předáme Analyzátoru natočení hlavy, který se již postará o případné zaslání zpráv 3D scéně, či vykreslení natočení do okna se snímaným obrazem
        - Nové pozice předáme Pohybovému analyzátoru
            - Nové pozice uloží Správce historie
            - Je zavolán Analyzátor částečných pohybů
            - Je zavolán Analyzátor pohybů
        - Je zavolán Rozpoznávač gest, je-li rozpoznáno gesto, vypíše se do konzole
- Úklid: uvolnění alokovaných prostředků z paměti
- Ukončení aplikace

# 11 Závěr

Cílem této práce bylo vytvořit aplikaci, která bude schopna detekovat lidskou hlavu, sledovat ji a na základě tohoto pak provádět úkony v počítači. Pro názornou ukázku ovládání počítače hlavou jsem si vybral jednak 3D scénu, která se podle natočení hlavy také natáčí, a potom jednoduché výpisy gest, které uživatel pohybem hlavy provedl. Jako body pro výpočet natočení hlavy jsem si zvolil obě oči a nos. Pro rozpoznávání jednoduchých gest jsem si vybral právě pohyb nosu, protože z těchto tří bodů je víceméně uprostřed hlavy a změny jeho polohy jsou nejznatelnější. K detekci hlavy, očí a nosu z obrazu jsem využil funkce knihovny OpenCV a to Haar-kaskád, které jsou natrénovaným kaskádami klasifikátorů pro detekci objektu v obraze využívající princip metody Viola-Jones. K sledování již nalezených objektů jsem využil princip metody Lucas-Kanade optického toku prostřednictvím KLT trackeru dostupného v OpenCV knihovně. Rozpoznání gest byla již čistě má práce založená na postupném rozpoznávání vlastností pohybů a následném přiřazování sémantického významu těmto pohybům.

Myslím si, že výsledná aplikace splňuje požadavky a je schopna pomocí gest ovládat počítač a cíl této práce byl tímto splněn. Ovšem pro obecné využití je aplikace ještě celkem nedokonalá a to hlavně díky nepřesnosti sledovače, který by si zajisté zasloužil mnohá vylepšení.

Určitě by bylo vhodné gestům, která se prozatím jen vypisují do konzole, přiřadit konkrétní význam a funkci, ať už v oblasti ovládané 3D scény nebo v oblasti celého operačního systému. Tato aplikace je tedy prozatím jen demonstrací postupu, jak je možné gesta a ostatní informace, jako je natočení hlavy, z obrazového toku vyextrahovat.

Aplikace by si jistě zasloužila inovaci i v oblasti obsluhy a ovládání, a to přidáním grafického uživatelského rozhraní, kde by bylo například možno v průběhu aplikace měnit různé konstanty využité v procesu rozpoznávání a sledování. Dále by zde byla možnost přidávat nová gesta, ať už formou definice gesta, či automaticky formou předvedení gesta počítači, nebo gestům přidávat či měnit jejich význam. Gesta by mohla být využita pro navigaci v dokumentech, v internetovém prohlížeči a podobně.

Kromě jiného by aplikace mohla být obohacena i o detekci směru pohledu a rozšířit tím svoje budoucí uplatnění, například jako náhrada za konvenční počítačovou myš, ať už v různých hrách, které by tento přístup mohly využívat, jako pomoc handicapovaným lidem, zejména s postižením horních končetin, nebo také pro obyčejné uživatele, kteří by využívali tuto novodobou myš a zároveň oběma rukama klávesnici.

Potenciál této platformy je opravdu obrovský a dle mého názoru je jen otázkou času, kdy se tento způsob komunikace s počítačem začne masově rozšiřovat.

# Literatura


[1] Viola, P., Jones, M.. Robust *Real-time Object Detection*. In Second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing, and Sampling. Vancouver, Canada, 2001

[2] Freund, Y., Schapire, R.E.. *A decision-theoretic generalization of on-line learning and an application to boosting*. In Computational Learning Theory: Eurocolt `95, stránky 23-37. Springer-Verlag, 1995

[3] Sonka, M., Hlavac, V., Boyle, R.. *Image Processing, Analysis, and Machine Vision*. Toronto: Thomson, 2008. ISBN 10: 0-495-08252-X 13: 978-0-495-08252-1

[4] Lucas, B.D.. *Generalized image matching by the method of differences*. Dizertační práce. Pittsburgh, Computer Science Department Carnegie-Mellon University, 1985

[5] Bradski, G., Kaehler, A.. *Learning OpenCV*. O'Reilly Media, Sebastopol, 2008. ISBN 978-0-596-51613-0

[6] Horn, B.K.P., Schunck, B.G.. *Determining Optical Flow*. In Artificial Intelligence, svazek 17, stránky 185-203, 1981

[7] Davies, E.R.. *Machine Vision*. Boston: Elsevier, 2005. ISBN 0122060938 9780122060939

[8] Jain, R., Kasturi, R., Schunck, B.G.. *Machine Vision*. New York: McGraw-Hill, 1995. ISBN 0070320187 9780070320185 0071134077 9780071134071

[9] Bradski, G.. *OpenCV Monthly*. In OpenCVWiki. Last modify on 15 May 2010. Dostupný z WWW: < http://opencv.willowgarage.com/wiki/OpenCV%20Monthly>

# Seznam příloh

1. Návod pro spuštění aplikace
2. Návod pro manipulaci s gesty
3. Gesta v gestures.txt
4. CD s tímto textem, zdrojovými soubory, spustitelnými soubory, návody, knihovnami, zkušebním video souborem a souborem gest

# Příloha 1: Návod pro spuštění aplikace

Tato práce byla vytvořena jako dva samostatné celky, jako dvě konzolové aplikace.

- Prve spustíme 3D scénu (soubor scena.exe) a to buď klasickým poklepáním, nebo z příkazové řádky bez jakýchkoliv parametrů. Tento krok je nepovinný, protože 3D scéna slouží pouze pro demonstraci detekce úhlů natočení hlavy
- Je potřeba aby dll knihovny (libcv200.dll, libcxcore200.dll, libhighgui200.dll, ) a xml soubory Haar-kaskád (haarcascade_eye.xml, haarcascade_frontalface_alt.xml, haarcascade_mcs_nose.xml) byly ve stejné složce se souborem detektor.exe. Spustíme detektor.exe poklepáním, nebo z příkazové řádky. Pokud jej pustíme poklepáním, tím pádem bez parametrů, bude program snímat obraz z „hlavní“ kamery vašeho počítače. Pokud kameru nemáte, program se po zapnutí ihned vypne. Taktéž při spuštění bez parametrů uvidíte pouze hlavní okno aplikace s obrazem z kamery, ve kterém nebudou vykresleny žádné další informace. Do konzole se budou vypisovat rozpoznaná gesta. Aplikaci jde parametrizovat:
    - Název „avi“ video souboru způsobí, že namísto obrazu snímaného z kamery bude analyzován obrazový tok ze zadaného souboru
    - Přepínač –d způsobí vykreslování detekovaných očí a nosu do obrazu, jejich nedávnou historii kreslí čárou, měnící svou světlost, a také v levém horním rohu ukazuje graficky natočení do jednotlivých úhlů

# Příloha 2: Návod pro manipulaci s gesty

Gesta jsou uložena v souboru gestures.txt a je nutné, aby tento soubor se nalézal ve stejné složce se souborem detektor.exe, který budeme spouštět. Tento soubor obsahuje definici gest, které chceme rozpoznávat.

- Gesto je v souboru uloženo vždy samostatně na jednom řádku a to v tomto formátu:

  Název_gesta;Tolerance_Globální;Tolerance_Lokální;Počet_pohybů_gesta;Orientace_pohybu_1;Normovaná_délka_pohybu_1;Orientace_pohybu_2;Normovaná_délka_pohybu_2;…..Orientace_pohybu_n;Normovaná_délka_pohybu_n;Popis

- V souboru jsou ignorovány řádky kratší, než 14 znaků, proto si zde můžeme vkládat řádky s komentáři. V souboru jsou již takovéto řádky a to s čísly a k nim přiřazenými zkratkami, které symbolizují orientace pohybů
- Normované délky pohybů u gest znamenají délky pohybů v pixelech
- Tolerance globální a lokální byly vysvětleny v kapitole 8.1.2 a 8.1.3 této práce
- Zde jsou čísla orientací pohybů, jejich zkratky i vysvětlení

| | | |
|---|---|---|
| 0 STAND: | Žáden pohyb. Bod zůstává na místě | |
| 1 DRD: | „Direct right down“: | Přímý pohyb doprava dolů |
| 2 DLD: | „Direct left down“: | Přímý pohyb doleva dolů |
| 3 DLU: | „Direct left up“: | Přímý pohyb doleva nahoru |
| 4 DRU: | „Direct right up“: | Přímý pohyb doprava nahoru |
| 5 DR: | „Direct right“: | Přímý pohyb doprava |
| 6 DL: | „Direct left“: | Přímý pohyb doleva |
| 7 DU: | „Direct up“: | Přímý pohyb nahoru |
| 8 DD: | „Direct down“: | Přímý pohyb dolů |
| 11 CRD: | „Concave right down“: | Konkávní pohyb doprava dolů |
| 12 CLD: | „Concave left down“: | Konkávní pohyb doleva dolů |
| 13 CLU: | „Concave left up“: | Konkávní pohyb doleva nahoru |
| 14 CRU: | „Concave right up“: | Konkávní pohyb doprava nahoru |
| 21 XRD: | „Convex right down“: | Konvexní pohyb doprava dolů |

22 XLD: „Convex left down“: Konvexní pohyb doleva dolů

23 XLU: „Convex left up“: Konvexní pohyb doleva nahoru

24 XRU: „Convex right up“: Konvexní pohyb doprava nahoru

100 NDB: „Not determined or break“: Nedeterminovaný pohyb, nebo break: Při deklaraci gesta nepoužitelný

# Příloha 3: Gesta v gestures.txt

Pro testovací účely je v souboru gestures.txt, na přiloženém CD, několik základních gest. Jsou to:

- YES_OK;0.9;0.5;2;7;100;8;100;prikyvnuti ano

  *Jde o gesto složené ze stejně dlouhých pohybů nahoru a dolů*

- YES_AGREED;0.9;0.5;2;8;100;7;100;prikyvnuti ano

  *Jde o gesto složené ze stejně dlouhých pohybů dolů a nahoru*

- YES_WONDER;0.9;0.5;4;7;50;8;100;7;100;8;50;pokyvani ano

  *Jde o gesto složené z pohybů: nahoru, dolů, nahoru, dolů, kde dva prostřední pohyby jsou dvakrát delší než dva pohyby okrajové*

- YES_DEFINITELY;0.9;0.5;4;8;50;7;100;8;100;7;50;pokyvani ano

  *Jde o gesto složené z pohybů: dolů, nahoru, dolů, nahoru, kde dva prostřední pohyby jsou dvakrát delší než dva pohyby okrajové*

- RIGHT;0.9;0.5;2;5;100;6;100;pokynuti vpravo

  *Jde o gesto složené ze stejně dlouhých pohybů doprava a doleva*

- LEFT;0.9;0.5;2;6;100;5;100;pokynuti vlevo

  *Jde o gesto složené ze stejně dlouhých pohybů doleva a doprava*

- NO;0.9;0.5;4;5;50;6;100;5;100;6;50;pokyvani ne

  *Jde o gesto složené z pohybů: doprava, doleva, doprava, doleva, kde dva prostřední pohyby jsou dvakrát delší než dva pohyby okrajové*

  NO;0.9;0.5;4;6;50;5;100;6;100;5;50;pokyvani ne

  *Jde o gesto složené z pohybů: doleva, doprava, doleva, doprava, kde dva prostřední pohyby jsou dvakrát delší než dva pohyby okrajové*

  Obě tyto gesta mají shodný význam, záleží však na uživateli, jestli pro odmítnutí kývá hlavou napřed doprava, či doleva.

- CIRCLE_CLOCKWISE;0.9;0.5;4;11;100;22;100;23;100;14;100;krouzeni hlavou ve smeru hod.rucicek

  CIRCLE_CLOCKWISE;0.9;0.5;4;22;100;23;100;14;100;11;100;krouzeni hlavou ve smeru hod.rucicek

  CIRCLE_CLOCKWISE;0.9;0.5;4;23;100;14;100;11;100;22;100;krouzeni hlavou ve smeru hod.rucicek

  CIRCLE_CLOCKWISE;0.9;0.5;4;14;100;11;100;22;100;23;100;krouzeni hlavou ve smeru hod.rucicek

  *Jde o gesta složená ze čtyř pohybů a to: konkávní doprava dolů, konvexní doleva dolů, konvexní doleva nahoru a konkávní doprava nahoru. Dohromady tyto pohyby o stejné délce tvoří kruh ve směru hodinových ručiček. Jsou zde ve čtyřech variantách proto, že uživatel může začít kroužit hlavou v kterékoliv části kruhu*

- CIRCLE_COUNTERCLOCKWISE;0.9;0.5;4;12;100;21;100;24;100;13;100;krouzeni hlavou proti smeru hod. rucicek

  CIRCLE_COUNTERCLOCKWISE;0.9;0.5;4;21;100;24;100;13;100;12;100;krouzeni hlavou proti smeru hod. rucicek

  CIRCLE_COUNTERCLOCKWISE;0.9;0.5;4;24;100;13;100;12;100;21;100;krouzeni hlavou proti smeru hod. rucicek

  CIRCLE_COUNTERCLOCKWISE;0.9;0.5;4;13;100;12;100;21;100;24;100;krouzeni hlavou proti smeru hod. ručiček

  *Jde o gesta složená ze čtyř pohybů a to: konkávní doleva dolů, konvexní doprava dolů, konvexní doprava nahoru a konkávní doleva nahoru. Dohromady tyto pohyby o stejné délce tvoří kruh proti směru hodinových ručiček. Jsou zde ve čtyřech variantách proto, že uživatel může začít kroužit hlavou v kterékoliv části kruhu*