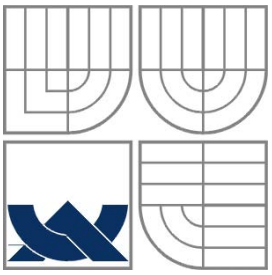
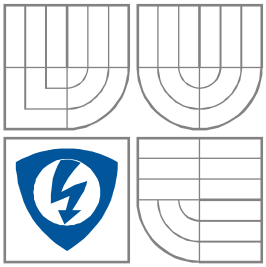


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SLEDOVÁNÍ POHYBU HLAVY VE SNÍMCÍCH
VISUAL HEAD MOTION MONITORING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Lukáš Sieber

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Miloslav Richter, Ph.D.

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Lukáš Sieber **ID:** 119603 **Ročník:** 3 **Akademický rok:** 2010/2011

NÁZEV TÉMATU: Sledování pohybu hlavy ve snímcích

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je navrhnout a implementovat algoritmus, který ze snímku určí možné rozsahy poloh a natočení hlavy vůči pevným ramenům. Předpokládá se snímání běžnou webovou kamerou a určení pohybu hlavy ve třech osách (translace a rotace). Pro co nejpřesnější měření je možné použít vhodně zvolené pomocné značky, které by však uživatele co nejméně omezovaly (nálepky, čelenka, brýle či jiné). Nedílnou součástí práce je provedený experiment s vyhodnocením přesnosti a spolehlivosti algoritmu.

DOPORUČENÁ LITERATURA:

HLAVÁČ, V., ŠONKA, M. Počítačové vidění. Praha: Grada, 1992. HAUßECKER, H., GEIßLER, P. Handbook of Computer Vision and Applications. San Diego: Academic press, 1999. BRADSKI, G., KAEHLER, A. Learning OpenCV: Computer Vision with the OpenCV Library. Cambridge : O'Reilly Media, 2008.

Termín zadání: 7.2.2011 **Termín odevzdání:** 30.5.2011

Vedoucí práce: Ing. Miloslav Richter, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Bakalářská práce se zabývá určením polohy a natočení hlavy v reálném čase ze snímků pořízených webovou kamerou vůči pevným ramenům. Dále poukazuje na výhody využití Open CV. Zahrnut je též podrobnější popis užitých funkcí Open CV, jako například optický tok, hledání korespondujících dvojic 3D-2D bodů a mimo jiné také operace s rotační maticí. V závěru jsou zhodnoceny dosažené výsledky a navržen další možný postup.

Klíčová slova

Webová kamera, Detekce obličeje, Poloha hlavy, Open CV, Natočení hlavy, Optický tok, Sledování pohybu, Rotační matice, Homografie, Open GL

Abstract

This bachelor thesis is based on finding of head rotation and position against stationary shoulders from frames captured by web camera in real time. Furthermore the thesis adverts to benefits of using Open CV. It is also included some Open CV function overview, such as optical flow, 3D-2D points corresponding calculation and among others operations with rotation matrix. The end of this thesis provides outcomes evaluation and future work proposals.

Keywords

Web camera, Face detection, Head position, Open CV, Head rotation, Optical flow, Movement tracking, Homography, Open GL

Bibliografická citace:

SIEBER, L. *Sledování pohybu hlavy ve snímcích*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 48s. Vedoucí bakalářské práce Ing. Miloslav Richter, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Sledování pohybu hlavy ve snímcích jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2011** ……

podpis autora

Poděkování

Děkuji vedoucímu své bakalářské práce Ing. Miloslavu Richterovi, Ph.D. a také Ing. Iloně Kalové, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: ……

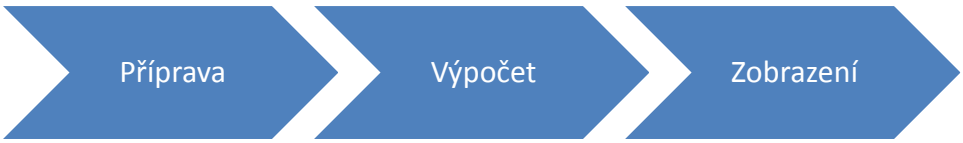
podpis autora

Obsah

1 ÚVOD

Bakalářská práce, kterou právě držíte v rukou, se zabývá sledováním pohybu a natočení hlavy na základě snímků pořízených webovou kamerou. Neklade si za účel podat vyčerpávající přehled navrženého algoritmu, ale ucelený soubor myšlenek vedoucí ke tvorbě funkčního celku. V průběhu čtení této práce se postupně seznámíme s nutnými základy pro pochopení dané problematiky.

Stoje před obtížnou úlohou, je velice moudré ji rozdělit na snadněji řešitelné části. V našem konkrétním případě jsou to postupně části preprocessing, výpočet a postprocessing.



Obrázek 1.1 – členění obtížné úlohy

V důsledku takového dělení jsme schopni se oprostit od komplexnosti celé úlohy a zaměřit se detailně jen na jednu její část. Pro názornost si lze představit opravdu dlouhý seznam úkolů a jejich postupné odškrtávání.

Přípravná část se zaměřuje na inicializaci proměnných, vstupních periferií a také nutné testování jejich korektní funkce. Jednoduše řečeno vytváří funkční základ celého algoritmu.

Část vlastního výpočtu je nejvíce obsáhlá. Zahrnuje v sobě výpočty optického toku, korekční procesy, rozličné převody barevných prostorů až po prahování a aplikaci konvolučního Gausova filtru.

Neméně důležitá je i část zabývající se o správné zobrazení výsledků, která navíc zahrnuje vizualizaci za pomocí grafické knihovny open GL. Její primární činností je však hlavně přehledný výpis vypočtených hodnot a to jak do okna obrazu snímaného kamerou, tak i do hlavního okna programu.

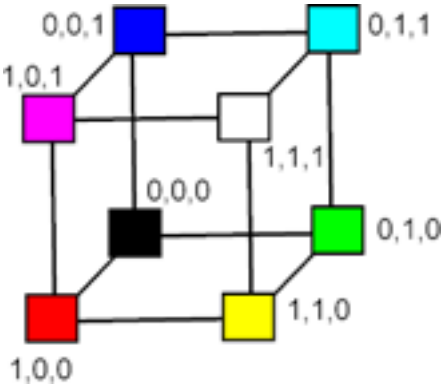
V závěru práce jsou zhodnoceny dosažené výsledky a navrhnuta možná zlepšení algoritmu. Za zmínku také stojí množství referencí na rozličné materiály, kterou mohou čtenáři rozšířit pohled na řešení daného problému.

2 BAREVNÉ MODELY

Zprostředkování informace o vizuální podobě světa nám přináší velmi sofistikovaný lidský orgán zvaný oko. Je schopné rozlišit úzký frekvenční pás elektromagnetického vlnění v pásmu 10^{14} Hz. Přesněji řečeno v tomto pásmu frekvence 4,3 * 10^{14} Hz (červená barva) ÷ 7,5 * 10^{14} Hz (barva fialová). Na základě podráždění tří barevných receptorů vnímajících červenou (R), zelenou (G) a modrou (B) barvu jsou vytvářeny skládáním (adicí) barvy další, kterých je oko na daném frekvenční rozsahu schopno vnímat až 400 000. Tento fakt dal vzniknout různým barevným modelům, více [1] [2] [3] [4].

2.1 Barevný model RGB

Model je založený na tzv. aditivním (sčítacím) přidávání barev. Na tomto principu jsou založeny již zastaralé CRT monitory a i dnešní běžné LCD panely TN. Stavebním kamenem modelu je kombinace tří základních barev červené (R), zelené (G) a modré (B), s jejichž pomocí jsou tvořeny barvy další.

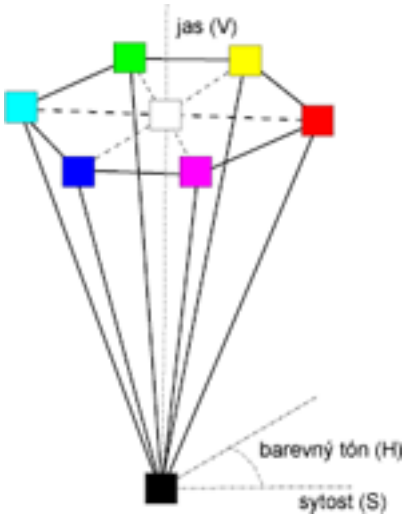


Obrázek 2.1 - převzato z [2]

Popis modelu velmi usnadní tzv. RGB krychle. Jas barev (svítivost) je zde vyjádřen hodnotou 0-1 (v počítači 0-255). Například barva červená (255,0,0), či bílá (255,255,255). K popisu bodu v tomto barevném prostoru nám tedy postačí 3 * 8 = 24 bitů (vzhledem k faktu, že jas barvy je určen 8 bity).

2.2 Barevný model HSV

Tento model se snaží barvy reprezentovat pro člověka přirozeným způsobem. Název HSV je vytvořen z počátečních písmen anglických slov H (hue – barevný tón), S (saturation - sytost), V (value - jas). Jeho prostorová podoba je vytvořena šestibokým jehlanem s vrcholem v počátku soustavy souřadnic.

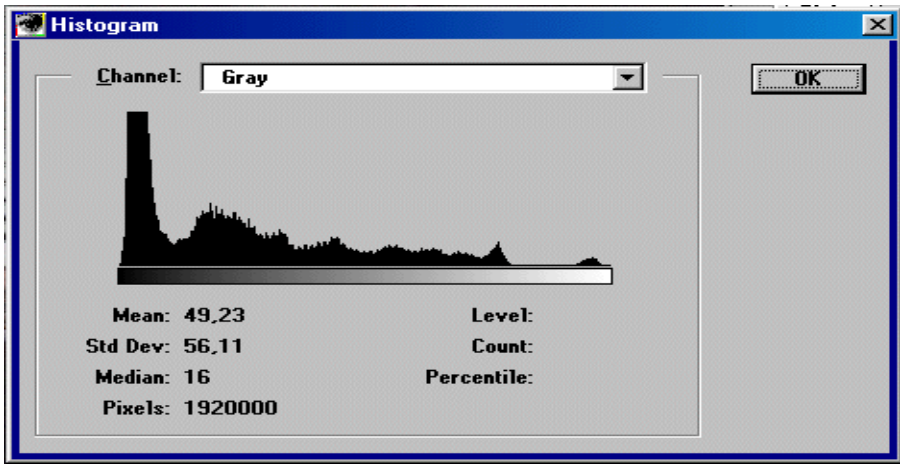


Obrázek 2.2 – HSV jehlan – převzato z [2]

Umístěného tak, že svislá osa představuje změny jasu (od 0 maximálně tmavá ÷ 1 maximálně světlá). Na ose vodorovné pak nalezneme sytost (měnící se od 0 maximální příměs jiných spektrálních barev ÷ 1 barva bez příměsi jiných spektrálních barev). A nakonec barevného tónu, který je určen jako úhel proti směru hodinových ručiček od osy S (0° ÷ 360°).

2.3 Histogram

Histogram neboli „graf obrázku“ lze popsat jako graf, na jehož ose x je vyneseno 256 bodů (body 0-255), které postupně reprezentují úrovně jasu od naprosto černé ÷ po naprosto bílou. Naproti tomu osa y určuje počet pixelů dané úrovně jasu na obrázku. Ve výsledku histogram udává rozložení jasu na analyzovaném obrázku (vpravo nejnižší jas a vlevo nejvyšší jas).



Obrázek 2.3 – histogram – převzato z [5]

Představme si, že máme barevnou fotografii v RGB modelu o rozměrech 640 x 480. V tomto případě je k dispozici 640 x 480 = 307 200 pixelů. Každý pixel nese přesnou barevnou informaci, která je dle pravidel pro model RGB reprezentována jako součet tří základních barev a to červené, zelené i modré. Pozornému čtenáři je v současné chvíli již jasné, že se všechny základní složky podle nějakého vzorce podílejí na výsledném jasu pixelu. Díky vzorci (1.) převedeme barevný obrázek na obrázek ve stupních šedi vyjadřující výsledný jas pixelu.

$$JAS = 0.3 * R + 0.59 * G + 0.11 * B \; ; \; kde \; R, G, B = 0 \div 255 \qquad (1.)$$

3 METODY DETEKCE POHYBU OBJEKTU

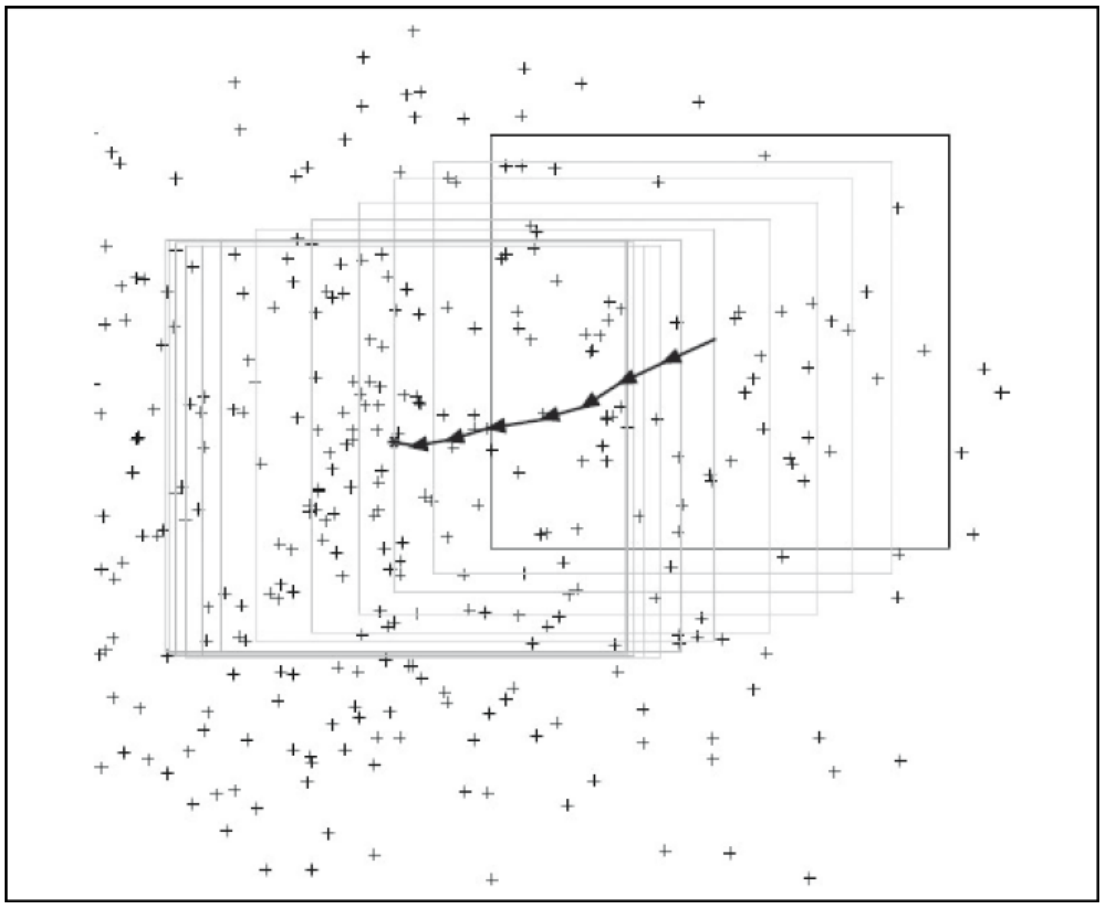
Díky této kapitole společně nahlédneme na princip detekce a sledování pohybu v obraze. Seznámíme se s algoritmem CamShift a samozřejmě také prozkoumáme možnost výpočtu tzv. optického toku.

3.1 Algoritmus camshift

CamShift je název sestavený z anglického „Continuously Adaptive Mean Shift“. Základem a stavebním kamenem toho algoritmu je tedy algoritmus Mean Shift vylepšený o schopnost adaptivně se přizpůsobovat tak, aby nejlépe kopíroval daný objekt. Základní myšlenkou algoritmu Mean Shift je převedení úlohy segmentace na úlohu shlukování bodů v d-rozměrném prostoru.

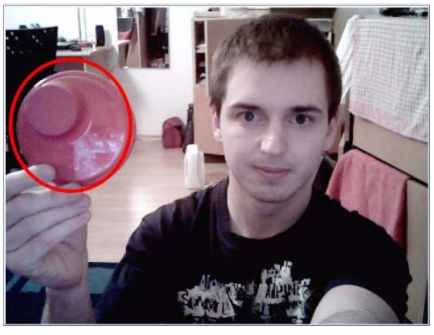
Představme si černobílý obrázek, jehož body můžeme v matici [x, y] reprezentovat jako hodnoty intenzity, nicméně daný obrázek lze také reprezentovat v třírozměrném prostoru [x, y, z], kde x a y udávají polohu a z výšky bodů (hodnotu jasu). Je tedy zřejmé, že objekty s podobným jasem (barvou) budou tvořit „shluky bodů“. Na obrázku 3.1 je příklad shluku bodů a také vymezení pracovní oblasti algoritmu (na obrázku čtverec). Při pohybu objektu (shluku bodů) probíhá neustálé přepočítávání těžiště (středu skupiny bodů uvnitř oblasti) a posun středu oblasti do tohoto bodu. Můžeme si všimnout určité podobnosti v názvu mean (střed, průměr, těžiště) a shift (posun). Algoritmus se vykonává tak dlouho, dokud pohyb objektu neustane.

Velmi důležitou informací je fakt, že pouze při inicializaci CamShiftu dojde k vytvoření histogramu (viz kapitola 2.3), který je následně použit ke sledování (hledání) objektu v obraze. Na základě vytvořeného histogramu je obraz navíc převeden na soubor bodů s intenzitou $0 \div 255$, které reprezentují pravděpodobnost, že daný pixel náleží hledanému objektu. Místo s největším shlukem vysoce pravděpodobných pixelů v dané oblasti je poté označeno jako hledaný objekt. Podrobnější informace nalezne lze nalézt v [6].



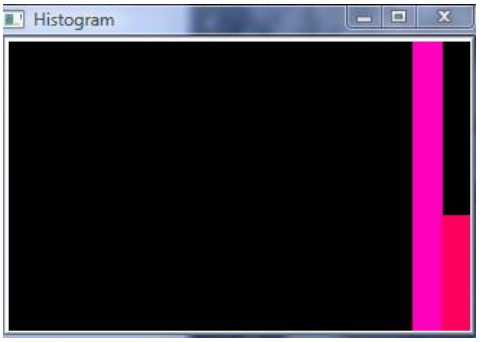
Obrázek 3.1 – Posouvání středu oblasti do těžiště shluku bodů převzato z [7]

Na závěr kapitoly o algoritmu CamShift příkládám ilustrační obrázky z praktického běhu algoritmu. Za povšimnutí stojí, jak pravděpodobnostní rozložení pixelů v obraze tzv. „backprojection“, tak i nastavení parametrů vmin a smin. Tyto parametry způsobí prahování obrazu (odstranění šumu) u pixelů téměř černých a téměř šedých.

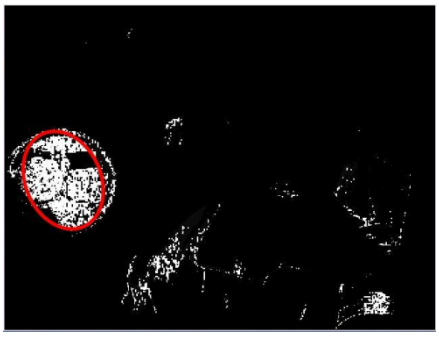


Obrázek 3.2 – ukázka camshift

Obrázek 3.3 – backprojection camshift



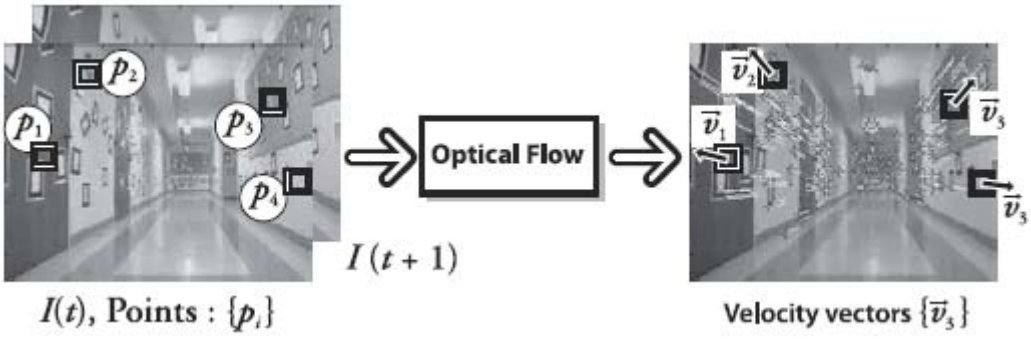
Obrázek 3.4 – histogram zvoleného objektu



Obrázek 3.5 – ladění pomoci smin a vmin

3.2 Optický tok

Optický tok lze rozdělit dle charakteru výpočtu na tzv. sparse (řídky) a dense (hutný). Hlavní rozdíl mezi oběma spočívá v náročnosti výpočtu a sledované oblasti. „Dense“ optický tok se snaží sledovat celou oblast obrázku a nejlépe každému pixelu (v praktickém použití bloku pixelů) přiřadit vektory reprezentující rychlost změny v daných směrech d-rozměrného systému. Naproti tomu „sparse“ optický tok se snaží vymezit pracovní oblast (například čtvercovou) a následně počítat optický tok v této oblasti, což se velice dobře projeví na rychlosti výpočtu. Open CV implementuje obě výše jmenované metody, avšak my se zaměříme na „sparse optical flow“. Tato metoda nejdříve zamýšlená jako „dense optical flow“ spatřila světlo světa roku 1981, ale pro svoji snadnou aplikaci na vybranou oblast pixelu se stala významnou metodou pro výpočet „sparse optical flow“.



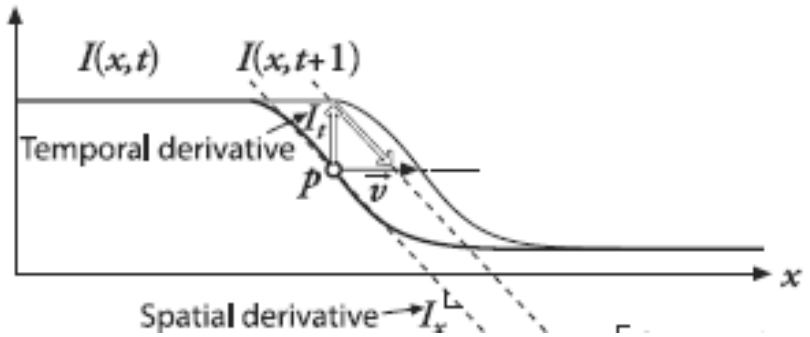
Obrázek 3.6 – Ukázka funkce optického toku a přirazení vektorů rychlosti (kamera se hýbe směrem do místnosti) – převzato z [7]

Aby bylo možné optický tok spočítat, je nutné vycházet z určitých předpokladů:

1. Pixel nemění svůj vzhled (barvu, jas) při pohybu ze snímku do snímku.
2. Změny polohy sledovaného bodu jsou malé, nebo alespoň relativně malé k rychlosti snímání scény.
3. Sousední body patří ke stejné ploše a konají stejný pohyb. Pokud body promítneme, jsou promítnuty blízko sebe (a tedy náleží stejnému objektu).

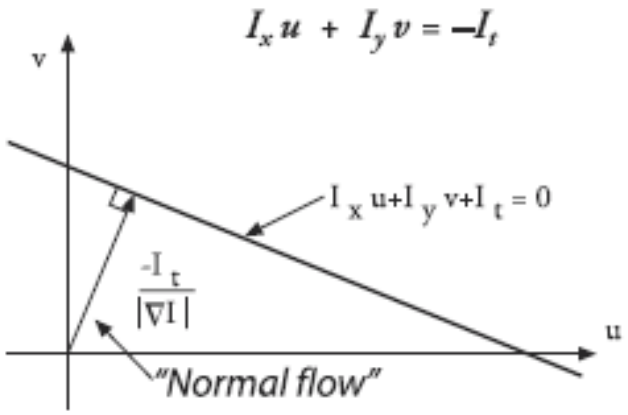
Třetí bod stojí za bližší prozkoumání. Představme si kouli v prostoru, pod kterou leží list papíru. Na kouli kolmo shora svítíme lampou. Stín vytvořený na papíru bude kruh tvořící obrys (průmět) koule. Nyní zvolíme libovolné dva body na kouli a promítneme je do papíru. Promítnuté body budou ležet blízko sebe a nedostanou se mimo obrys (kruh) na papíře a tedy patří k danému objektu (kouli).

Dosažení ideálních podmínek je v reálném světě mnohdy velice obtížné. Například změny světelných podmínek vnášejí do výpočtu chybu, případně změna polohy objektu je velice rychlá ke snímací frekvenci kamery a nedá se označit za velice malou (limitně nulovou).



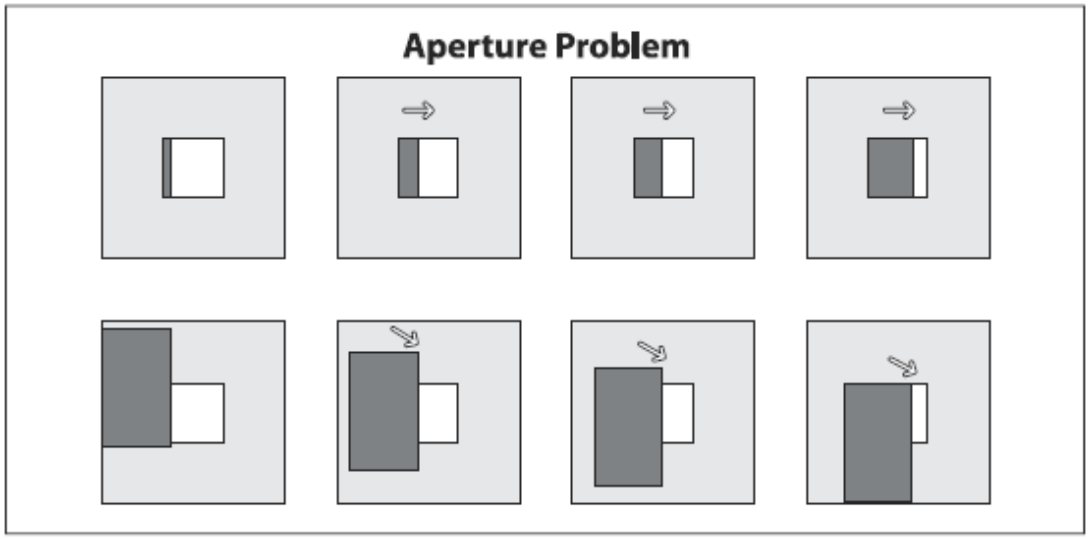
Obrázek 3.7 – sledování pohybu hrany pixelu v 1-rozměrném prostoru – převzato z [7]

Obrázek 3.7 ilustruje výpočet vektoru v z jeho dílčích částí I_t a I_x. Vzhledem k výše jmenovaným faktorům vnášejícím chybu do výpočtu je hodnota vektoru v pouze přibližná. Tato skutečnost nám až tak nevadí, pokud je tento „odhad“ blízko hodnotě skutečné. V takovém případě se dá k výsledku s uspokojivou přesností dojít užitím např. Newtonovy iterační metody (jako počáteční odhad byla volena velikost vektoru v). Pojďme se nyní podívat jak je to s přechodem do prostoru 2-rozměrného.



Obrázek 3.8 – přechod 1D – 2D –převzato z [7]

Na první pohled se může zdát jednoduché rozšířit rovnici *Ixu + It = 0* o jednu další souřadnici *Iy*, což však ve výsledku dá *Ixu + Iyv + It = 0*. A dojdeme tak k výsledku, že máme dvě neznámé, ale jen jednu rovnici. Využitím předpokladu 3 je možné sledovat pohyb středu vetší oblasti sledováním pohybu okolních pixelů v této oblasti. Zvolme tedy oblast 5*5 pixelů, což nám ve výsledku dá 25 rovnic (rovnice pro každý pixel). V tuto chvíli máme zase rovnic zbytečný nadbytek. Metodou nejmenších čtverců je proto počet rovnic redukován na dvě. Na závěr této kapitoly se společně ještě podívejme na tzv. „problém hrany“ (aperture problem).



Obrázek 3.9 – problém hran – převzato z [7]

Komplikace nastávají při volbě příliš malého okna pro sledování pohybu, kdy zachytíme pouze hranu pohybujícího se objektu, ale žádné rohy. Potom není možné určit, jakým směrem se daný objekt pohybuje. Na druhou stranu volbou příliš velikého okna dojdeme do rozporu s předpokladem 3 (dané body patří k jedné ploše) a sledování objektu nebude pracovat dobře. Bližší informace jsou k dispozici v [7].

4 MODEL KAMERY

Při snímání scény webovou kamerou dochází ke ztrátě informace o z-tovou souřadnici. Jednou z hlavních komplikací počítačového vidění je snaha o zpětnou rekonstrukci této ztracené informace. Následující kapitola pojednává o jedné z možností, jak opět tuto informaci získat.

4.1 Homogenní souřadnice

Potřeba homogenních souřadnic vznikla v důsledku snahy o popsání všech základních transformací obrazu s pomocí operací maticového násobení.

Mezi základní operace patří:

- Posunutí
- Otáčení
- Změna měřítka
- Zkosení

Všechny tyto operace mimo posunutí ve 2D prostoru *(x, y)* je možné popsat operací maticového násobení. Z tohoto důvodu vznikla potřeba homogenních souřadnic, které souřadný systém *(x, y)* rozšiřují o další souřadnici *w*. Jedná se tedy o přechod z 2D do 3D prostoru. Souřadnice *w* se standardně volí jako 1.

$$(X, Y, W) \rightarrow \left(\frac{X}{W}, \frac{Y}{W}\right), protože\ platí\ x = \frac{X}{W}\ a\ y = \frac{Y}{W} \qquad (2.)$$

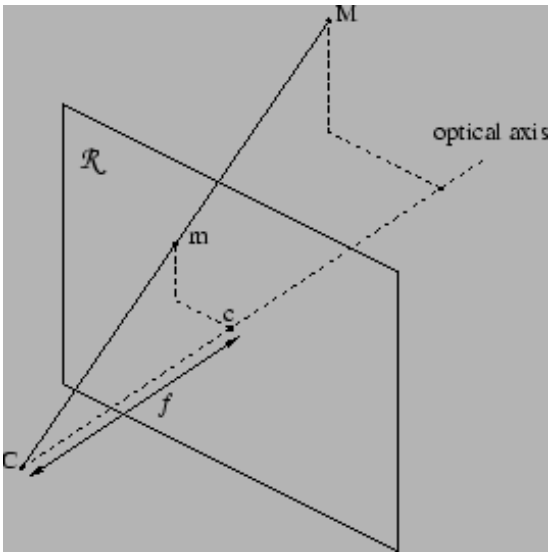
Dle rovnice (2.) je patrný převod mezi body v homogenních souřadnicích (X, Y, W) a body (x, y) kartézského systému. Bližší informace nalezne čtenář v [8].

4.2 Model „dírkové“ kamery

Abychom mohli provádět transformaci mezi souřadným systémem kamery a souřadným systémem okolního světa, je potřeba vytvořit model kamery a tento model matematicky popsat.

V případě modelu dírkové kamery jsou paprsky světla z okolí soustředěny (prochází) jedním bodem. Tento bod je většinou umístěn ve středu souřadného systému kamery. Před jmenovaným bodem se nachází projekční plocha, která je od středu

promítání (středu souřadného systému kamery) vzdálena o ohniskovou vzdálenost dané kamery.



Obrázek 4.1 – model dírkové kamery převzato z [9]

Na obrázku 4.1 je názorně vidět popisovaná souvislost. Mezi bodem *C* (střed souřadného systému kamery a středového promítání) a bodem *M* (bod v prostoru okolí) se nachází projekční plocha *R*. Úsečka mezi středem *C* a středem projekční plochy *c* je ohnisková vzdálenost dané kamery, prodloužení této úsečky je tzn. optickou osou a v tomto případě splývá s osou *Z* souřadného systému kamery. Průsečík přímky mezi body *C* a *M* s rovinou *R* je projekcí bodu *M* z okolního světa do projekční plochy *R* (je označen jako *m*). Daný princip lze zjednodušeně matematicky popsat takto:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \sim \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix},$$

$$kde\ (x, y)\ jsou\ body\ kamery\ a\ (X, Y, Z) jsou\ body\ okolí \tag{3.}$$

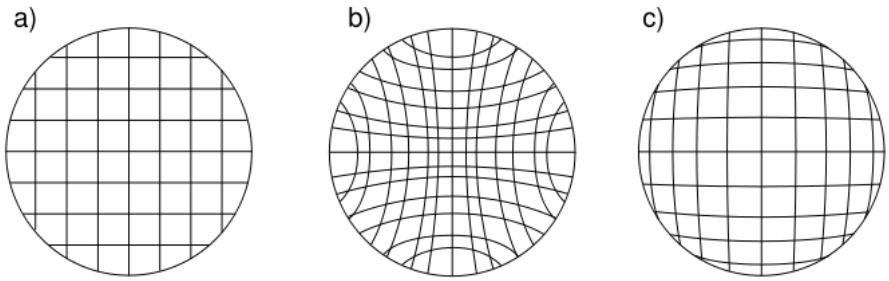
$dalé\ platí\ že\ x = \left(fx * \frac{X}{Z} + cx\right)\ a\ y = (fy * \frac{Y}{Z} + cy)$, cx a cy

jsou souřadnice středu projekční plochy R. fx a fy je ohnis. vzdál.

Je zde uvedena přibližná rovnost a to z důvodu existence poruch, které do snímání vnáší nedokonalosti optického systému kamery. Mezi dvě hlavní poruchy patří Zkreslení obrazu a porucha způsobená nedokonalostí umístění senzoru kamery (není umístěn paralelně s průmětnou plochou *R*).

V prvním případě je zkreslení velice dobře patrné na okrajích obrazu, kdy právě zde jsou přenášeny paprsky přes okraj čočky, kde je zkreslení obrazu největší.

Nejmenšího zkreslení je tedy dosaženo velice blízko optické osy v bodě *c*. V druhém případě dochází vlivem nerovnoběžnosti snímacího senzoru a projekční plochy k přenášení obrazu mimo vypočítaný střed *c*. Podrobné informace je možné najít v [7].



Obrázek 4.2 - názorná ukázka zkreslení obrazu čočkou převzato z [10]

Obrázek 4.2 názorně ukazuje možné chyby zkreslení v důsledku nedokonalosti optické čočky a to v případě:

- a, jde o nezkreslený obrázek
- b, jde o poduškovité zkreslení
- c, jde o soudkovité zkreslení

4.3 Kalibrace kamery

Předmětem kalibrace kamery je získání projekční matice kamery M [1]. Díky znalosti projekční matice jsme schopni za využití operací maticového násobení transformovat body z okolí do souřadného systému kamery.

$$M = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.)$$

Předmětem této práce však není stanovení algoritmu pro výpočet a určení parametrů matice kamery, ale hledání rotačního a translačního vektoru popisujícího změnu polohy hlavy vůči souřadnému systému kamery.

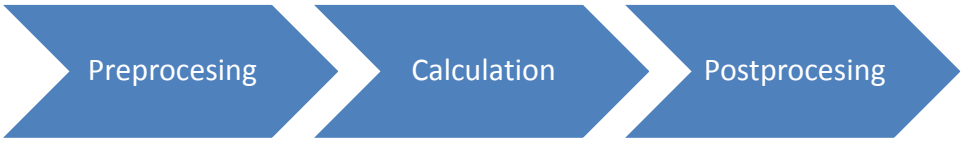
Z tohoto důvodu je k získání těchto „vnitřních“ parametrů kamery použit algoritmus popsaný v [7] str. 378 kapitola Calibration, kde se z velkého počtu snímků bodů nalezených na šachovnici numericky určí hodnota prvků matice *M*.

[1] fx a fy jsou dvě rozdílné ohniskové vzdálenosti vzhledem k nedokonalosti tvaru pixelu u levných kamer (není to čtverec). Tyto hodnoty získáme jako fx = F * sx a fy = F * sy, kde F je ohnisková vzdálenost a s je velikost elementu rastru kamery v daném směru (x nebo y).

5 VLASTNÍ NÁVRH ALGORITMU

Následující kapitola čtenáře seznámí s postupným návrhem algoritmu pro sledování pohybu a natočení hlavy za pomocí webové kamery. V případě nutnosti jsou v kapitole blíže popsány použité funkce knihovny open CV a to hlavně z pohledu praktického přínosu pro daný algoritmus.

Samotný algoritmus lze rozdělit postupně do částí předzpracovaní (preprocessing), výpočetní části (calculation) a části pro zobrazení výsledků (postprocessing).

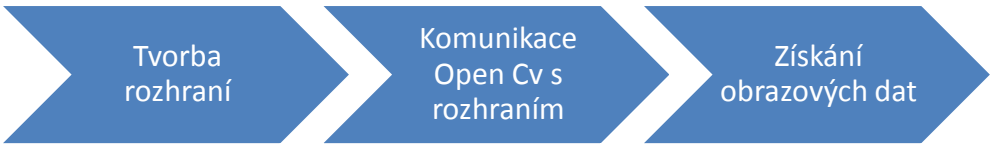


Obrázek 5.1 – postupný průběh algoritmu

5.1 Preprocessing

V souvislosti s touto bakalářskou prací je zde příprava chápana jako načtení obrazových dat za pomocí webové kamery a jejich následná transformace do podoby, která nám usnadňuje operace s nimi.

V první fázi je nutné sestavit komunikační rozhraní mezi knihovnou open CV a webovou kamerou. Toho lze dosáhnout zavoláním funkce *cvCreateCameraCapture(0)* volbou parametru 0 necháváme vybrat první dostupnou kameru. Následným cyklickým voláním funkce *cvQuerryFrame(rozhraní)* získáváme snímky pořízené kamerou.



Obrázek 5.2 – tvorba rozhraní pro komunikaci s kamerou

Dalším důležitým krokem je nastavení přiměřeného rozlišení a to hlavně z pohledu rychlosti a přesnosti algoritmu. Řadou pokusů se jako nejlepší jeví zvolení rozlišení 800*600 (při vzdálenosti hlavy od kamery průměrně 80cm). Volbou rozlišení nižšího (640*480) docílíme rychlejšího běhu algoritmu, za cenu nutného snížení vzdálenosti (body se jeví blíže u sebe => pro bezpečné rozlišení menší vzdálenost

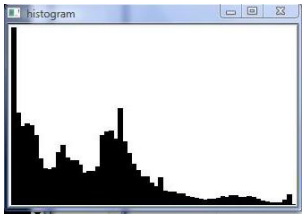
kamery a hlavy), což vede v důsledku také k menším mezním úhlům natočení. Diametrálně odlišné situace dosáhneme volbou rozlišení vyššího (1024*768), kdy na testovaném systému[2] algoritmus neprobíhal již zcela plynule.

Také neméně důležitou úpravou před vlastním výpočtem je normalizace snímků. Tato normalizace je zde zastoupena hlavně z důvodu využití optického toku, který jak bylo popsáno, vychází z předpokladu konstantního jasu sledovaného pixelu (lze také chápat jako konstantní barvu při převodu snímku z RGB na snímek ve stupních šedi). Normalizace je zde dosaženo využitím funkce cvEqualizeHist(cílový, zdrojový), která jako své parametry přebírá pouze zdrojový obrázek a obrázek cílový kam je uložen zdrojový po normalizaci. Normalizace probíhá dle vzorce (5). Více informací na [11].

$$H_e(j) = \frac{q_k - q_0}{M * N} * \sum_{i=0}^{j} H_p(i) + q_0 \qquad (5.)$$

Kde H_e je výsledný normalizovaný histogram q_k,q_0 je interval požadovaných šedotónových úrovní a H_p je histogram původní. Rozložení histogramu téměř odpovídá funkci normálního rozložení.

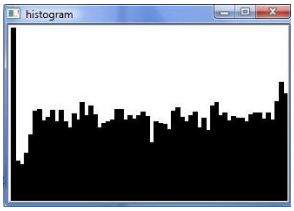
Obrázek 5.3 – před ekvalizací



Obrázek 5.4 – před ekvalizací

Z histogramu neekvalizovaného obrázku je zjevné veliké zastoupení světlých pixelů. Je tedy žádoucí nějakým způsobem obrázek normovat a docílit tak větší robustnosti algoritmu pro výpočet optického toku.

Obrázek 5.5 – po ekvalizaci



Obrázek 5.6 – po ekvalizaci

[2] Intel Core i5 @ 2,8Ghz (s technologii turbo boost = automatické přetaktovaní), Není využito technologie NVIDIA CUDA

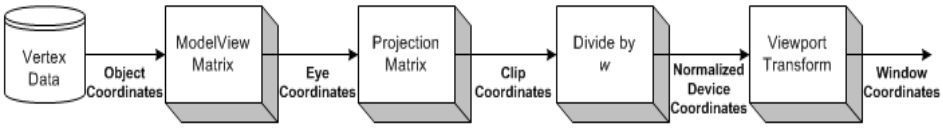
Jak je vidět, rozložení pixelů je zastoupeno v celém spektru odstínu šedi. Použitý obrázek byl dle rovnice (1 viz kapitola 2) převeden na obrázek ve stupních šedi. Tento způsob je zvolen z důvodu zjednodušení a zrychlení výpočtu (pracování s hodnotami intenzity namísto práce se 3 kanály intenzit barevného obrázku).

Poslední důležitou částí preprocesingu je inicializace stavového automatu pro open GL. Tento stavový automat postupně opakuje spouštění zaregistrovaných funkcí a díky tomu se cyklicky překresluje vizualizace vytvořená za pomocí nástavby open GL rozhraní GLUT. Celá aplikace poté běží v novém vlákně mimo vlákno hlavního programu.

Hlavní části pro správnou funkci jsou:

- Funkce pro zobrazení *display*
- Funkce pro překreslení okna *resize*
- Funkce pro čekání *idle*
- Funkce pro spuštění stavového automatu *start_opengl*

V prvním kroku je nutné nastavit projekční matici open GL (udává druh projekce 3D scény pravoúhlá/perspektivní a také stanovuje, které části objektu jsou ještě viditelné) tato matice nese název *Projection Matrix.* Následně se nastaví matice *Model View*, díky které lze nastavit místo pohledu oka (kamery) a navíc násobením této matice prvky z matice rotační upravenými pro použití s open GL, dosáhneme vizualizace pohybu (translace a rotace).



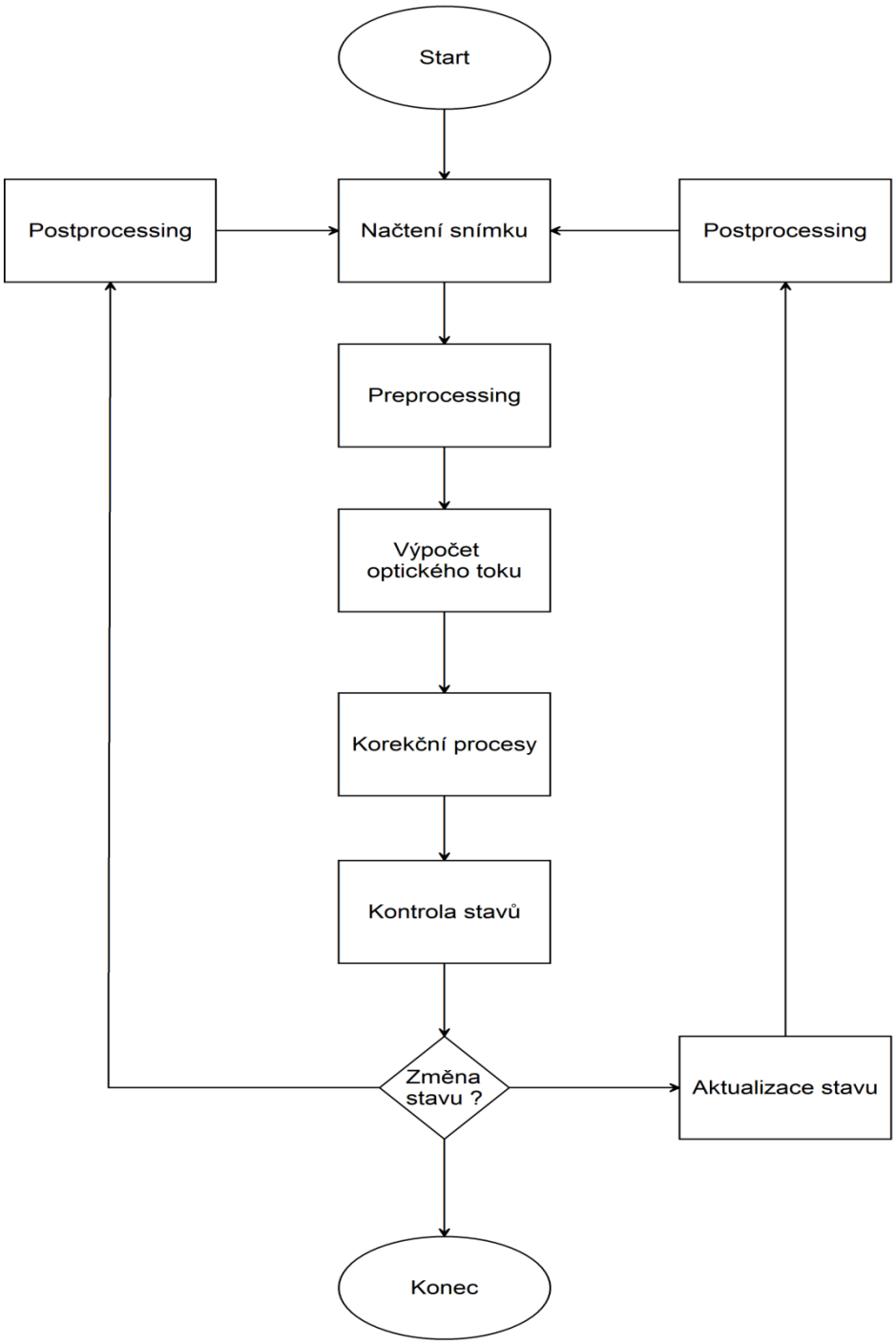
Obrázek 5.7 – řetězec postupného výpočtu pro open GL převzato z [12]

Za zmínku také stojí využití techniky tzn. „double bufferingu“, kdy je scéna nejdříve vytvořena v paměti a teprve poté až vykreslena uživateli. Výše uvedená technika znatelně vylepší vizuální dojem (scéna nepoblikává).

Nicméně problematika Open GL je natolik rozsáhlá, že značně přesahuje rámec této bakalářské práce. Pokud se i přesto čtenář rozhodne do ní hlouběji proniknout, je možné čerpat z [12], [13], [14] a případně také množství kvalitní odborné literatury.

5.2 Výpočetní část

V průběhu kapitoly Výpočetní část je nejdříve důraz kladen na dva bloky níže uvedeného a zjednodušeného blokového diagramu. Těmito bloky jsou výpočet optického toku i korekční procesy. A následně výpočtu úhlu, rotaci a dalšího.



Obrázek 5.8 – Zjednodušený blokový diagram

5.2.1 Výpočet optického toku

Nyní společně krok po kroku projdeme jednu ze dvou hlavní částí algoritmu pro sledování pohybu hlavy ze snímků pořízených webovou kamerou. Následující text si neklade za úkol vysvětlit princip optického toku (ten je popsán v kapitole 3), ale osvětlit soubor úkonů vedoucí k algoritmizaci dané problematiky.

Důležitým faktorem robustnosti při výpočtu optického toku je dodržení tří hlavních zásad:

1. Pixel nemění svůj vzhled (barvu, jas) při pohybu ze snímku do snímku.
2. Změny polohy sledovaného bodu jsou malé, nebo alespoň relativně malé k rychlosti snímání scény.
3. Sousední body patří ke stejné ploše a konají stejný pohyb. Pokud body promítneme, jsou promítnuty blízko sebe (a tedy náleží stejnému objektu).

Z prvního bodu plyne snaha o co nejlepší světelné podmínky celé scény a pokud jich není dosaženo, je snahou co největší přiblížení se tomuto stavu. Tento fakt zohledňuje část preprocessingu prováděním ekvalizace histogramu.

Druhého bodu se týkají, jak softwarová, tak hardwarová omezení. Je důležité si uvědomit, že levné kamery dávají značně omezené možnosti nastavení. Drivery takových webových kamer dělají téměř veškerou činnost za uživatele a možnosti nastavení se zde omezují v lepších případech na zapnuto/vypnuto. Také je dobré vzít na zřetel, že knihovna open CV používá pro záznam z kamery framework VCM (Video Compression Manager), který v současné době již nepatří k nejnovějším (uvolněn už pro windows 3.1). Uvedené zápory lze vyřešit zakoupením lepší webové kamery a případně užitím directshow (pro operační systém Windows). Je na místě také uvést, že knihovna open CV od verze 2.2 podporuje technologii CUDA, která otevírá dveře, rozložením výpočtu mezi procesor a GPU, novým možnostem.

Platnost třetího bodu je v algoritmu zajištěna zvolením velikosti sledovaného objektu dostatečně malého a ve scéně jedinečného vůči celkové ploše snímané kamerou[3].

[3] Logitech C270

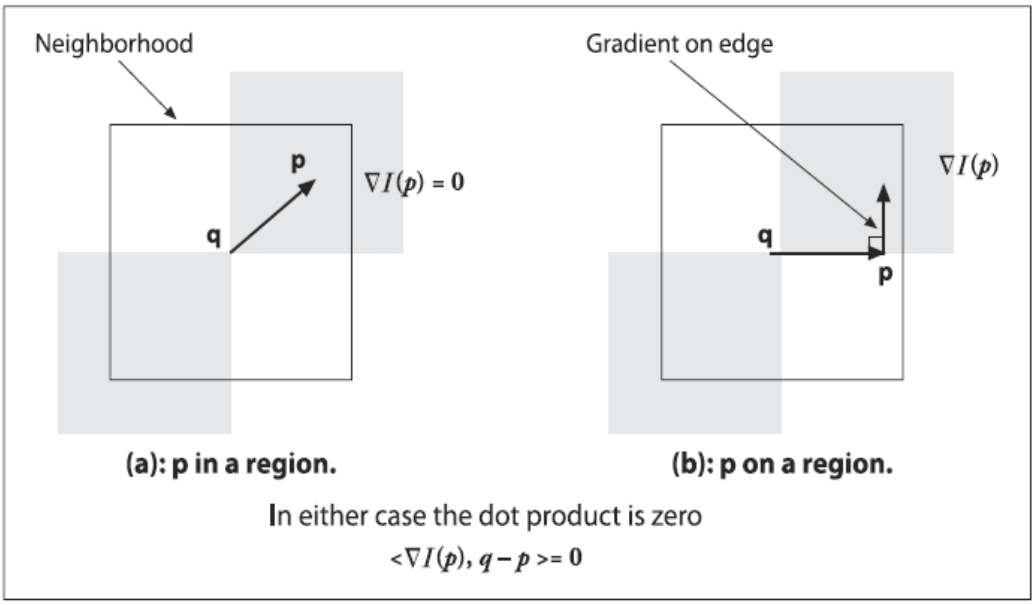
5.2.2 Korekční procesy

V důsledku pohybu sledovaného objektu (bodu) v prostoru dochází v rámci výpočtu optického toku k určitým chybám. Tento stav je zapříčiněn velice malou částí numerickým řešením rovnic a následně podstatnou částí těchto chyb je přítomnost šumu v obraze. Neméně podstatným zdrojem chyb jsou též rušivé objekty zakrývající objekt sledovaný, nedostatečné světelné podmínky a v neposlední řadě prudké pohyby, které nesplňují předpoklad dva pro optický tok.

Snahou korekčních procesů je dopad těchto rušivých vlivů eliminovat, a pokud tohoto není možno dosáhnout, tak přinejmenším korigovat dopad těchto vlivů.

5.2.2.1 Korekce pozice sledovaného bodu

Zde je s výhodou využito funkce Open CV *cvFindCornerSubPix().* Princip funkce si demonstrujeme na obrázku 5.9.



Obrázek 5.9 – algoritmus pro korekci pozice bodu převzato z [7]

Na obrázku jsou znázorněny dva body *q* a *p*. Předpokládejme, že bod *q* leží téměř v rohu (místu vhodném pro sledování) dané oblasti, nicméně protože používáme celočíselné souřadnice, je zde určitá chyba. A právě algoritmus pro hledání „subpixelu“ se snaží tuto chybu minimalizovat.

Vyjdeme z několika předpokladů:

- Stejnorodá oblast má nulový gradient
- Skalární součin dvou kolmých vektorů je nula
- Pro oblast více pixelů (např. čtvercovou) jsme schopni sestavit soustavu rovnic odrážejících 1 a 2 předpoklad.

Určitou představu o práci tohoto algoritmu již tedy máme. Nyní si přiblížíme parametry důležité pro jeho fungování. Dle funkčního předpisu *cvFindCormerSubPix (zdrojový obrázek, detekované hrany, počet předaných hran, velikost okna okolních pixelů, mrtvá zóna, ukončující podmínka).* Funkci předáváme v první řadě obrázek, se kterým má pracovat. Dále nalezené hrany na předaném obrázku. Také je potřeba definovat oblast okolí pixelu, pro kterou budou sestaveny rovnice a také mrtvou zónu chránící algoritmus před zamrznutím (pro velice blízké body již nelze sestavit dostatek rovnic). Protože se jedná o algoritmus iterační, ukončující podmínka umožňuje výpočet ukončit (dosažení přesnosti, dosažení počtu iterací).

Otázkou však zůstává, kdy takové korekční procesy provádět. Je velice nerozumné realizovat takový proces stále. Nejen z pohledu výpočetní náročnosti, ale také v zájmu správné funkce celého algoritmu. Vzhledem k výše zmíněným faktům je korekce prováděna jen v případě přímého pohledu do kamery. V takový moment je totiž plně viditelná maska nasazená na sledované hlavě. Při ztrátě sledovaného bodu je okamžitě změněna stavová proměnná a dochází k opětovnému vyhledání bodů.

5.2.2.2 Výpočet reprojekční chyby

Jsme-li schopni získat rotační a translační vektor, který popisuje způsob transformace bodu z okolí do souřadného systému kamery, jsme také schopni spočítat reprojekční chybu (rozdíl mezi bodem sledovaným a bodem spočítaným za pomocí bodů modelu násobených rotačním a translačním vektorem). Znalost této informace přidává algoritmu možnost kontrolovat kvalitu sledování bodů a případně reagovat na její nízkou hodnotu (vysoká hodnota reprojekční chyby).

Celý algoritmus výpočtu lze provést ve dvou krocích. Nejdříve je nutné spočítat aktuální polohu i natočení modelu a následně ve druhém kroku převést souřadnice z prostoru 3-rozměrného do prostoru 2-rozměrného (souřadnice bodu na projekční ploše). Převod posléze probíhá dle rovnic (6.), (7.), (8.)

$$X_{tr} = (X * Rxx + Y * Rxy + Z * Rxz) + Tx \quad (6.)$$

$$Y_{tr} = (X * Ryx + Y * Ryy + Z * Ryz) + Ty \quad (7.)$$

$$Z_{tr} = (X * Rzx + Y * Rzy + Z * Rzz) + Tz \quad (8.)$$

Kde X_{tr}, Y_{tr} *a* Z_{tr} jsou postupně transformované souřadnice modelu za pomocí prvků z rotačního vektoru *R* a translačního vektoru *T*. Následující rovnice (9.) a (10.) slouží k transformaci výše uvedených souřadnic ze systému 3-rozměrného do systému 2-rozměrného (homogenní souřadnice, kde *w* odpovídá *Z*).

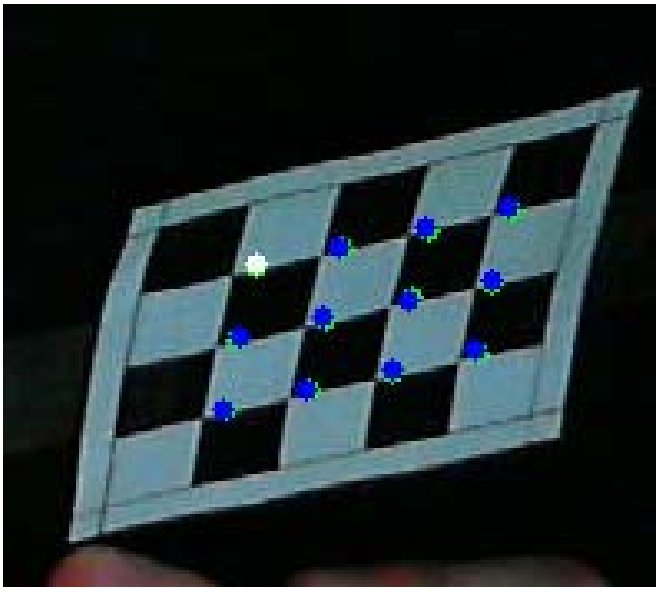
$$x = cx + (fx * \frac{X_{tr}}{Z_{tr}}) \quad (9.)$$

$$y = cy + (fy * \frac{Y_{tr}}{Z_{tr}}) \quad (10.)$$

Kde *x* a *y* jsou již souřadnice, v souřadném systému kamery na průmětně (viz kapitola 4). Také si lze všimnout prvků matice kamery *M* a to *cx*, *cy*,*fx a fy*. Tyto prvky udávají pozici středu průmětny (*cx a cy*) a také ohniskovou vzdálenost (*fx a fy*).

Vlastní reprojekční chybu nakonec snadno spočítáme dle vzorce (11.).

$$\delta = \left(x_{sledované} - x_{spočítané}\right)^2 + \left(y_{sledované} - y_{spočítané}\right)^2 \quad (11.)$$



Obrázek 5.10 – chyba reprojekce

Na obrázku 5.10 je názorně vidět chyba reprojekce. Jsou zde postupně body modré (body zpětné projekce), které jsou získané za pomocí funkce open CV *cvProjectPoints2 (body objektu, rotační vektor, translační vektor, matice kamery, korekční koeficienty, místo uložení spočítaných bodů).* Dále body světle modré (sledované), které jsou získané pomocí funkce *cvFindChessboardCorners (pracovní obrázek, velikost šachovnice, místo pro uložení bodů, proměnná pro uložení počtu bodů, stavové proměnné)* a poté sledované technikou optického toku. Posledním zástupcem obsažených bodů je zde bod bílé barvy. Tento bod je získán dle rovnic obsažených v kapitole 5.2.2.2.

5.2.3 Výpočet rotačního a translačního vektoru

Rotační a translační vektor jsou pro nás v této práci velice důležité. Právě tyto dva vektory v sobě nesou informace potřebné k určení polohy a natočení 3D objektu v prostoru. Abychom byli schopni tyto dva vektory najít je potřeba dle [7] vyjít z rovnice:

$$q = s * M * W * Q \tag{12.}$$

Kde *q* je vektor hodnot bodu v souřadném systému kamery na její průmětně, *s* je faktor zmenšení vzhledem ke středovému promítání (promítnutý objekt je menší). *M* je matice obsahující parametry kamery, *W* je matice slučující rotační a translační vektory a na závěr *Q* je vektor hodnot bodů okolí.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s * \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r1 & r2 & r3 & t \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z = 0 \\ 1 \end{bmatrix} \tag{13.}$$

Rotační vektory *r1,r2* a *r3* obsažené v matici W jsou na sebe kolmé (osy rotace jsou zároveň osami souřadného systému). Díky tomuto faktu lze zvolit souřadnici *Z* jako 0 a tím vyloučit *r3*. Taková úprava umožní vyřešení této rovnice a zároveň nezapříčiní ztrátu informace. Jsme totiž schopni získat *r3* jako vektorový součin *r1* a *r2*.

Nyní označme *s * M * W* jako matici *H.*

$$H = s * M * W \tag{14.}$$

$$q = H * Q \tag{15.}$$

Body v souřadném systému kamery q (body na průmětně kamery) a také body v prostoru okolí Q jsou známé ve chvíli detekce bodů na šachovnici. V takovém případě jsme schopni spočítat matici H (transformační matice bodu je velikosti 3*3). Dle rovnice (14.) můžeme vyjádřit výpočet vedoucí k rotačním vektorům a vektoru translačnímu jako:

$$H = [h1 \quad h2 \quad h3] = s * M * [r1 \quad r2 \quad t] \tag{16.}$$

$$r1 = \frac{1}{s} * M^{-1} * h1 \tag{17.}$$

$$r2 = \frac{1}{s} * M^{-1} * h2 \tag{18.}$$

$$t = \frac{1}{s} * M^{-1} * h3 \tag{19.}$$

$$r3 = r1\ x\ r2 \tag{20.}$$

Kde *t,h1,h2,h3,r1,r2* a *r3* jsou sloupcové matice o velikosti 3*1.

5.2.4 Získání velikosti úhlu rotace z rotační matice

Ke zdárnému vyřešení této komplikace je nutné se nejdříve společně podívat, jak taková rotační matice vůbec vypadá. Předpokládáme 3 různé rotace a to postupně kolem osy *X*, osy *Y* a nakonec osy *Z*. Natočení hlavy v prostoru je posléze popsáno právě kombinací těchto rotací a také translačním vektorem (vyjadřuje posun).

$$Rx(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{21.}$$

$$Ry(\beta) = \begin{bmatrix} \cos\alpha\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{22.}$$

$$Rx(\gamma) = \begin{bmatrix} \cos\gamma & \cos\gamma & 0 \\ 0\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{23.}$$

Provedením rotací (operace maticového součinu) získáme finální podobu rotační matice. Její tvar uvedeme v rovnici (24.)

$$R = \begin{bmatrix} \cos\beta * \cos\gamma & \sin\alpha \sin\beta \cos\gamma - \cos\alpha \cos\gamma & \cos\alpha \sin\beta \cos\gamma + \sin\alpha \sin\gamma \\ \cos\beta * \sin\gamma & \sin\alpha \sin\beta \sin\gamma - \cos\alpha \cos\gamma & \cos\alpha \sin\beta \cos\gamma - \sin\alpha \sin\gamma \\ -\sin\beta & \sin\alpha \cos\beta & \cos\alpha \cos\beta \end{bmatrix} \quad (24.)$$

Dokonce z takto roznásobené kompletní rotační matice, která obsahuje rotace postupně okolo každé osy, lze získat velikost jednotlivých úhlů rotace ve stupních. Nejjednodušší je získání úhlu β, ten je obsažen na třetím řádku prvního sloupce. Je důležité si uvědomit, že goniometrické funkce se periodicky opakují, a proto je možné získat dvě řešení, viz rovnice (25.), (26.) a (27.).

$$\beta_1 = (-\sin R_{31})^{-1} \quad (25.)$$

$$\beta_2 = \pi - \beta_1 = \pi + (\sin R_{31})^{-1} \quad (26.)$$

$$R_{31} \neq \pm 1 \quad (27.)$$

Dalším adeptem na získání velikosti úhlu z rotační matice se stane úhel α. Veškeré zde vypočítané úhly jsou v radiánech a je nutné posléze na stupně převést. Rovnice (28.) a (29.) zachycují získaní úhlu α.

$$\frac{R_{32}}{R_{33}} = \frac{\sin\alpha \cos\beta}{\cos\alpha \cos\beta} = \frac{\sin\alpha}{\cos\alpha} = \tan\alpha \quad (28.)$$

$$\alpha = \operatorname{atan}\frac{R_{32}}{R_{33}} \quad (29.)$$

Vyloučením *cos* β jsme přišli o informaci, v jakém kvadrantu se řešení nachází, proto je důležité na základě jeho hodnoty určovat, zdali je úhel α kladný nebo záporný (leží v rozmezí *(-π,π)*). Východiskem vzniklé situace je využití funkce atan2 (dostupná ve většině programovacích jazyků), která na základě znamének parametrů určí, v jakém kvadrantu se prvky nacházejí. Po takové úpravě lze psát viz rovnice níže.

$$\alpha_1 = \operatorname{atan2}\left(\frac{R_{32}}{\cos\beta_1}, \frac{R_{33}}{\cos\beta_2}\right) \quad (30.)$$

$$\alpha_2 = \text{atan2}(\frac{R_{32}}{\cos\beta_2}, \frac{R_{33}}{\cos\beta_2}) \quad (31.)$$

$$\cos\beta \neq 0 \quad (32.)$$

Analogickým způsobem zjistíme, že vhodnou kombinací prvků z rotační matice lze získat i úhel γ a to dle rovnice (33.). Řešení je potom za dosazení patřičných hodnot stejné jako v rovnicích (30.) a (31.).

$$\gamma = \frac{R_{21}}{R_{11}} = \tan\gamma \quad (33.)$$

Pojďme se společně blíže podívat na případ, kdy $cos\,\beta = 0$. V takové případě jsme ve stavu, kdy se $\beta = \pi/2$ nebo $-\pi/2$ a zvolené prvky matice v rovnicích výše již nadále neobsahují informaci o rotaci. Řešením je získat potřebnou informaci z jiných prvků rotační matice. Nejdříve vezmeme v potaz případ kdy $\beta = \pi/2$.

$$\beta = \frac{\pi}{2} \quad (34.)$$

$$\cos\beta = 0 \quad (35.)$$

$$\sin\beta = 1 \quad (36.)$$

$$R_{12} = \sin\alpha\ \cos\gamma - \cos\alpha\cos\gamma = \sin(\alpha - \gamma) \quad (37.)$$

$$R_{13} = \cos\alpha\ \cos\gamma + \sin\alpha\sin\gamma = \cos(\alpha - \gamma) \quad (38.)$$

$$R_{22} = \sin\alpha\ \sin\gamma + \cos\alpha\cos\gamma = \cos(\alpha - \gamma) \quad (39.)$$

$$R_{23} = \cos\alpha\ \sin\gamma - \sin\alpha\cos\gamma = -\sin(\alpha - \gamma) \quad (40.)$$

Pokud společně nyní vydělíme prvky matice R_{12} a R_{13} jsme schopni získat z nich úhel α-γ.

$$\frac{R_{12}}{R_{13}} = \tan(\alpha - \gamma) \quad (41.)$$

$$\alpha - \gamma = a\tan\frac{R_{12}}{R_{13}} \tag{42.}$$

$$\alpha = \gamma + a\tan\frac{R_{12}}{R_{13}} \tag{43.}$$

Úhel γ zvolíme 0, případně jakékoliv jiné číslo, které může posloužit jako offset (posun celého výpočtu do hodnot, které chceme) a úhel α dopočítáme. V tomto konkrétním případě je úhelβ = π/2. Analogickým způsobem lze získat hodnoty případ kdy β = -π/2.

Ve chvíli, kdy jsme získali veškeré úhly je nutné je převést na stupně podle jednoduchého vzorce uvedeného rovnicí (44.). Informace byly čerpány z [15].

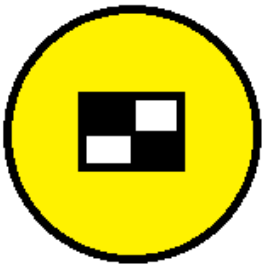
$$\text{úhel ve stupních} = \frac{\text{úhel v radiánech} * 180}{\pi} \tag{44.}$$

Obrázek 5.11 – ukázka funkce algoritmu pro výpočet úhlů

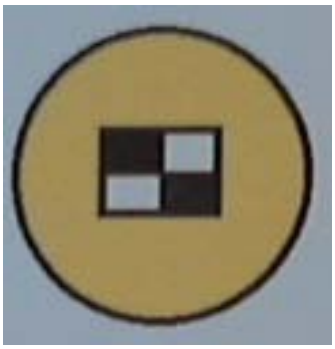
Na obrázku 5.11 si lze všimnout vypočítaných úhlů a to postupně v pořadí alfa (rotace kolem osy x), beta (rotace kolem osy y) a konečně gamma (rotace kolem osy z). Také je zde obsažen výpis složek translačního vektoru (X, Y a Z), počet snímku za sekundu (FPS) a rovněž i hodnota reprojekční chyby.

5.2.5 Zajištění podmínky pevných ramen

Dodržení podmínky zadání o pohybu vůči nehybným ramenům je zde dosaženo za pomocí přídavných značek, jejichž poloha je sledována. Při pohybu těchto bodů (nalezených díky přídavným značkám) mimo toleranční pásmo (je nastaveno na 20 bodů v každém směru) je zahlášen chybový stav a uživatel požádán o novou manuální inicializaci.



Obrázek 5.12 – navržená značka v pc



Obrázek 5.13 – stejná značka po tisku a vyfocení pomocí kamery

Na obrázcích výše si můžeme všimnout markantního rozdílu mezi značkou navrženou v počítači a její vyfocené[4] kopie po tisku. Pozice značky je získána prahováním spodní poloviny snímaného obrazu, který je ještě rozdělen na levou a pravou část (v každé části se nachází jedna značka). Celý výpočet je posléze proveden v barevném prostoru HSV, což nám umožní definovat žlutou barvu a její odstíny (světlejší a tmavší žlutá) jako rozsahy hodnot daného barevného prostoru. Rozsahy jsou získány z obrázku 5.13.

5.2.5.1 Hledání středu pomocné značky

Pro nalezení středu pomocných značek je s úspěchem využito tzn. obrazových momentů nultého a prvního řádu. Před vlastním výpočtem momentů prováděným nad prahovaným obrazem, jsou obrazová data filtrována Gausovým filtrem s jádrem 3x3 ve snaze o odstranění šumu (vnáší chybu do výpočtu středu značky).

Předpokládáme kartézské souřadnice a diskrétní (digitální) podobu obrazu. Potom podle [16] můžeme vzorec pro výpočet momentů obrazu psát ve tvaru:

[4] Logitech c270

$$m_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} X^p * Y^q * P_{xy} \qquad (45.)$$

Kde m_{pq} je dvojrozměrný moment, $X^p * Y^q$ je tzv. základní funkcí (může mít mnoho podob a díky tomu přenáší do momentu potřebné vlastnosti), P_{xy} reprezentuje obrazový bod a M i N jsou rozměry obrazu. Z rovnice (45.) je patrné, že nultý moment m_{00} je součtem všech obrazových bodů (plochou) daného digitálního obrazu. Pokud dále provedeme výpočet momentů prvního řádu, tzn. m_{01} a m_{10} a normujeme tyto momenty vzhledem k celkové ploše obrazu (nultý moment). Získáme souřadnice x a y středu (těžiště) rovnice (46.) a (47.).

$$souřadnice\ středu\ x = \frac{m_{10}}{m_{00}} \qquad (46.)$$

$$souřadnice\ středu\ y = \frac{m_{01}}{m_{00}} \qquad (47.)$$

Přesně k nalezení středu značek (na ramenou) jsou momenty v tomto programu využity. Ve snaze najít střed značek. Tím samozřejmě jejich možnosti nekončí, z těch zajímavějších vlastností jmenujme například invariantnosti vůči posunutí, nebo invariantnost vůči změně měřítka. Takových vlastností se dá dosáhnout právě správnou volbou výše uvedené základní funkce.

Obrázek 5.14 – Prahovaná značka na základě rozsahu hodnot v bar. schématu HSV

Obrázek 5.14 zobrazuje výsledek prahování snímaného obrazu za účelem nalezení pomocné značky umístěné na ramenech sledované osoby. Tmavé body na daném obrázku nesou hodnotu 0 a světlé hodnotu 1. Naší snahou je tedy zvolit rozsahu hodnot HSV tak, aby prahovaný obraz obsahoval nejlépe jen body značky a žádné další rušivé, nacházející se mimo oblast značky. Takovéto rušivé body by posléze vnesly do výpočtu středu značky za užití obrazových momentů chybu.

6 POST PROCESSING

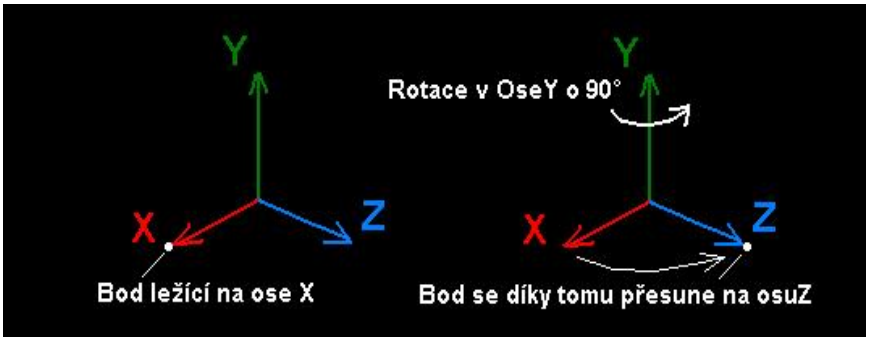
Poslední nedílnou součástí celého algoritmu je zobrazení vypočtených závěrů a případně také jejich vizualizace. K tomuto účelu je zde s úspěchem využito grafické knihovny open GL, o které již bylo pojednáno v kapitole 5.1.

V současné chvíli už tušíme, co je to rotační matice. Je velice pravděpodobné, že i knihovna open GL nějakou svoji rotační matice využívá. Taková informace není sama o sobě nijak převratná, ale když se pokusíme využít získané rotační matice z funkcí open CV nebudou správně fungovat. Důvodem je specifický tvar rotační matice open GL.

RotXx	RotYx	RotZx	0
RotXy	RotYy	RotZy	0
RotXz	RotYz	RotZz	0
MoveX	MoveY	MoveZ	1

Obrázek 6.1 – tvar rotační matice openGL převzato z [13]

Pohledem na obrázek 6.1 si lze všimnout dvou označení *Rot* a *Move*. První označení uvádí prvky rotační matice a druhé prvky translačního vektoru. Matice openGL tedy obsahuje, jak část rotace, tak i část posunu. Velká písmena (X,Y,Z) reprezentují původní osu a malá (x,y,z) osu, na kterou se případně osa původní transformuje. Princip je velice dobře pochopitelný na obrázku 6.2.

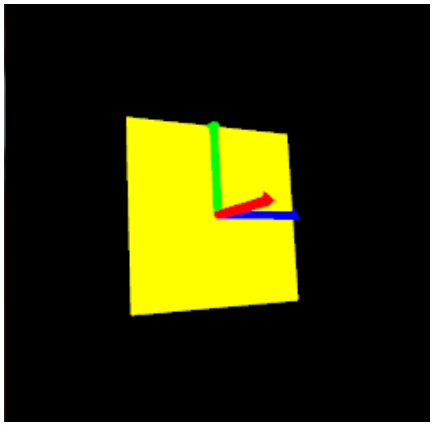


Obrázek 6.2 – princip transformace osy při rotaci o 90° okolo osy Y – převzato z [13]

Takovýchto rotací lze dosáhnout zavoláním openGL funkce *glRotatef(úhel rotace, X,Y,Z).* Kde *X, Y a Z* reprezentují osy, dle kterých se bude daná rotace provádět (taková osa má jako parametr 1 a ostatní 0). Namísto předávání rotačního vektoru, který je vypočítán za pomocí funkcí open CV.

6.1.1 Zobrazení úhlu, FPS, translace a reprojekční chyby

Pro potřeby přesnějšího zobrazení pohybu hlavy než pomocí vizualizace, jsou hodnoty různých výpočtů zobrazeny a to rovnou do okna sledovaného pohybu. Postupně se tak od horního okraje okna můžeme setkat s výpisem úhlů natočení, translačním vektorem, počtem snímků za sekundu a konečně hodnotou reprojekční chyby.



Obrázek 6.3 – grafická vizualizace

Obrázek 6.4 – textová vizualizace

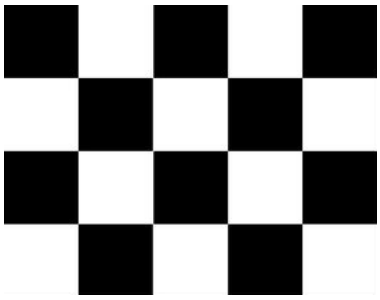
Obrázek 6.3 a 6.4 ukazuje grafický výstup aplikace a to hned ve dvou provedeních. První snímek zachycuje grafickou vizualizaci (osa x je zelená, osa y je modrá, osa z je červená) vytvořenou knihovnou open GL a snímek druhý textový výstup.

K výpočtu FPS byl použit princip popsaný v [17], kdy je na základě počtu průchodů hlavním cyklem programu a potřebného času vypočítána průměrná hodnota FPS. Důležitým zjištěním je, že software kamery Logitech C270 při zapnuté funkci truelight značně omezí počet snímku za sekundu. Pro sledování bodů s využitím optického toku je toto zlepšení FPS citelně přínosné a napomáhá robustnosti celého algoritmu.

7 TESTOVÁNÍ FUNKCE ALGORITMU

V rámci určení míry naplnění zadání a funkčnosti navrženého konceptu je nutné provést sérii testování. Kapitola věnovaná testům ukáže výhody a také nedostatky daného algoritmu. V testování se postupně zaměříme na mezní polohy natočení, necháme vykreslit sérii jednoduchých grafů a také dáme nahlédnout na rychlost pohybu.

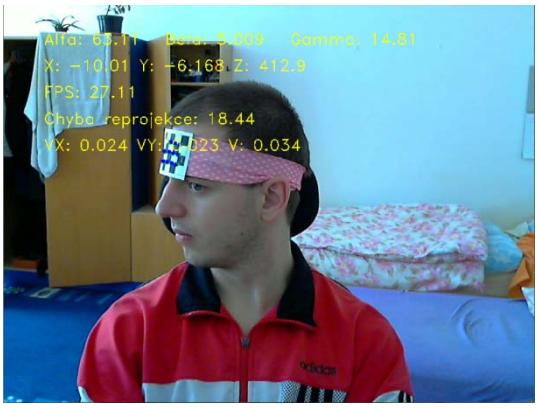
Veškeré grafy jsou tvořeny knihovnou pro openCV dostupnou na [18].



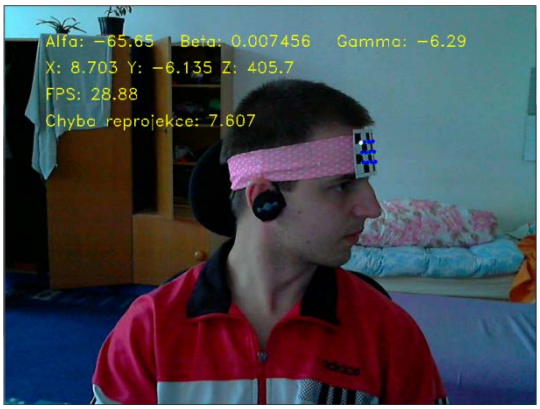
Obrázek 7.1 – sledovaná značka je šachovnice o rozměrech 4*5

7.1 Mezní polohy natočení hlavy

Vzhledem ke snímání obrazu pouze jednou kamerou narazíme brzy na problém mezní polohy a to hlavně při pohledu do strany. Tento stav je efektivně možné řešit použitím vícekamerového systému, kdy po průchodu nějaký mezním úhlem (např. 45°) dojde k přepnutí a snímání obrazu kamerou další.



Obrázek 7.2 – mezní pohled doprava



Obrázek 7.3 – mezní pohled doleva

Na obrázcích výše si můžeme všimnout mezních poloh při otáčení hlavy do strany. Při otáčení doleva (obrázek 7.3) bylo dosaženo mezního úhlu alfa (otáčení

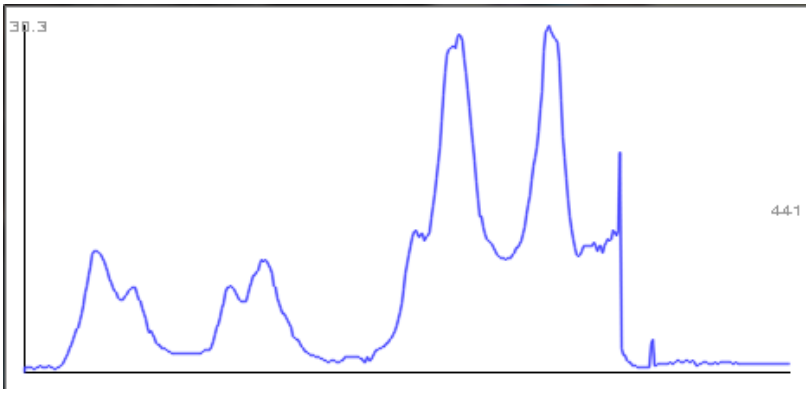
kolem osy x) s hodnotou 65,05°. Pokusem o co největší otočení hlavy do strany vůči pevným ramenům vznikl obrázek 7.2, jedná se o otočení hlavy doprava s mezním úhlem o hodnotě 63,11°.

Obrázek 7.4 – chyba algoritmu

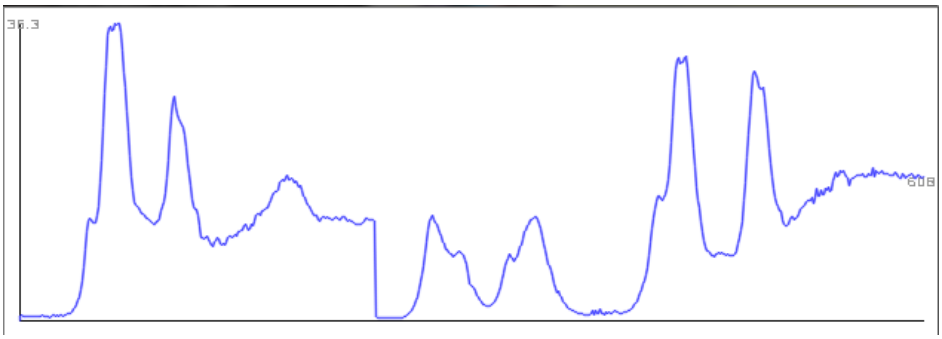
Obrázkem 7.4 je názorně ukázán problém mezních úhlů. Hodnota úhlu alfa je zde špatně interpretována jako -81,6° (tato hodnota odpovídá natočení na druhou stranu). Ze snímku je také patrné, že sledované body leží již téměř v jedné přímce a velikost chyby zde již neumožňuje spoléhat na hodnotu úhlu.

Nyní společně nahlédneme na průběh hodnot reprojekční chyby při různých rotacích a na průběh hodnot samotných úhlů v závislosti na počtu cyklů hlavní smyčky programu (za jeden průběh hlavního cyklu je načten jeden snímek). Osa x zde reprezentuje právě množství průchodů (počtu snímků) a osa y hodnotu úhlu, případně reprojekční chyby. Grafy jsou velice zjednodušené a slouží hlavně k získání přehledu o průběhu dané veličiny.

Pohledem na obrázek 7.5 je možné získat představu o průběhu reprojekční chyby v závislosti na počtu průběhu hlavním cyklem (počtu snímků). Prvním pohybem je otáčení hlavy doleva, této činnosti odpovídají první dvě maxima. Nejdříve prudký vzrůst chyby při začátku pohybu a její pokles, až následně vzrůst chyby v konečné krajní poloze. Situace je shodná s průběhem pohybu na druhou stranu, avšak vlivem osvětlení je chyba značně větší (hlava se nachází v tomto případě proti oknu. V případě menší chyby proti dveřím, které nepředstavují zdroj světla). Poslední významnou částí je prudký pád chyby v závěru grafu, který je způsoben zavoláním korekčního procesu a posunutím bodů na střed rozhraní polí šachovnice.

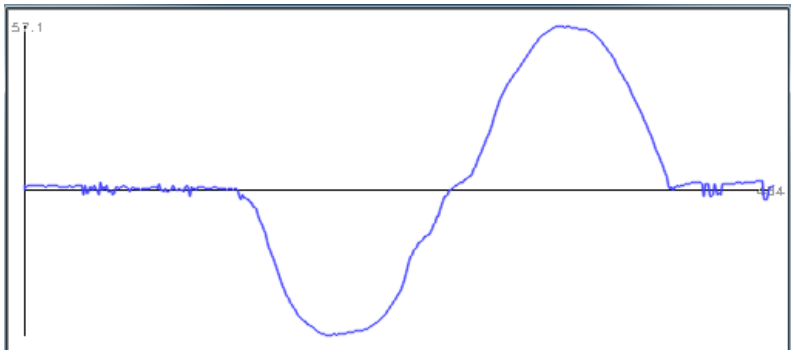


Obrázek 7.5 – průběh chyby při pohledu levá + pravá je prováděna korekce



Obrázek 7.6 – průběh chyby při pohledu levá+pravá kde není prováděna korekce

Z testovacích důvodů jsou vypnuty korekční procesy algoritmu a sledován vliv na výsledný průběh reprojekční chyby. Situaci zachycuje obrázek 7.6, kde si můžeme všimnout ve výsledku ustálené hodnoty reprojekční chyby po ukončení pohybu do stran a návratu do středové polohy. Co už příjemné není, je ustálená hodnota této chyby na poměrně vysoké hodnotě (přibližně 17). Za takových podmínek nelze na přesnost algoritmu spoléhat, což potvrzuje i fakt, že algoritmus přestává být schopen určit, jakým směrem se hlava pohybuje (patrné na vizualizaci při běhu algoritmu).



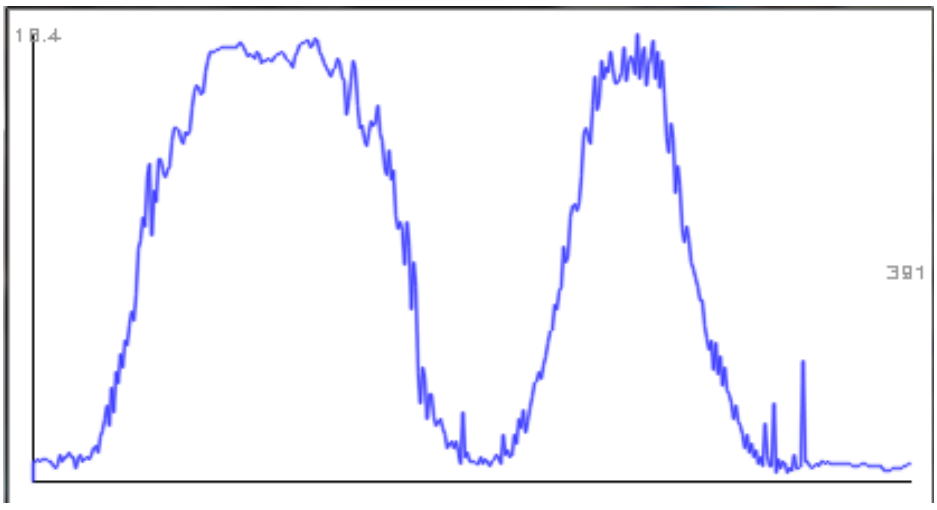
Obrázek 7.7 – průběh velikosti úhlu rotace alfa (rotační osa x)

U obrázku 7.7 je zachycen průběh velikosti úhlu alfa (rotace kolem osy y => otáčení hlavy do stran). Jeho maximální hodnota je 57° a to hlavně z důvodu vyloučení ztráty bodů vlivem vysoké reprojekční chyby a zorného pole jedné kamery (pohyb je veden vůči pevným ramenům).

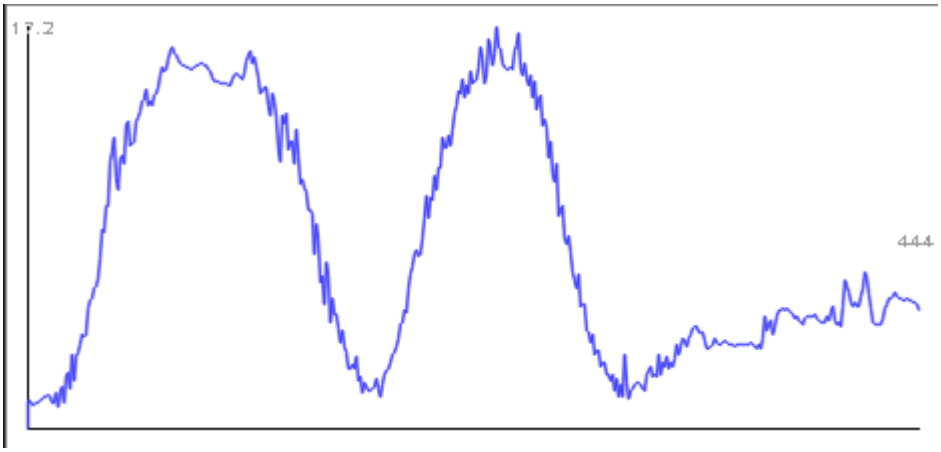
Obrázek 7.8 – mezní pohled nahoru

Obrázek 7.9 – mezní pohled dolů

Další dva mezní pohyby hlavou jsou směrem vzhůru a dolů. Oba pohyby jsou viditelné na obrázcích 7.8 a 7.9. Dosažené mezní hodnoty úhlu beta (rotace kolem osy y => pohyb nahoru a dolů) jsou postupně 55,61° při pohledu nahoru a 43,66° při pohledu dolů. Opět platí, že v mezních krajních polohách je vysoká reprojekční chyba viz obrázky 7.10,7.11.

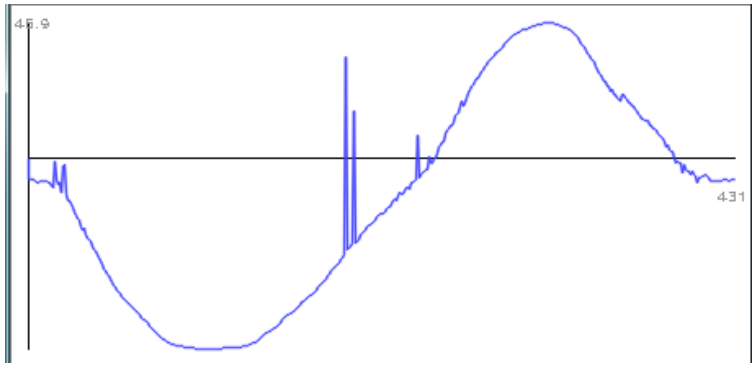


Obrázek 7.10 – průběh velikosti reprojekční chyby při rotaci beta (rotační osa y) + korekce

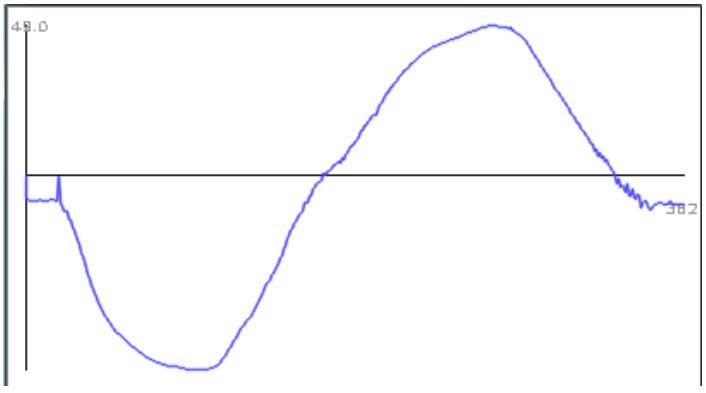


Obrázek 7.11 - – průběh velikosti reprojekční chyby při rotaci beta (rotační osa y) bez korekce

Velikost reprojekční chyby se zde pohybuje kolem hodnoty 17. Teď se společně pomocí obrázků 7.12 a 7.13 přesvědčíme o přínose znalosti hodnoty reprojekční chyby, kdy jsme s její pomocí schopni ušetřit značnou část výpočetního času a také zlepšit fungování algoritmu.



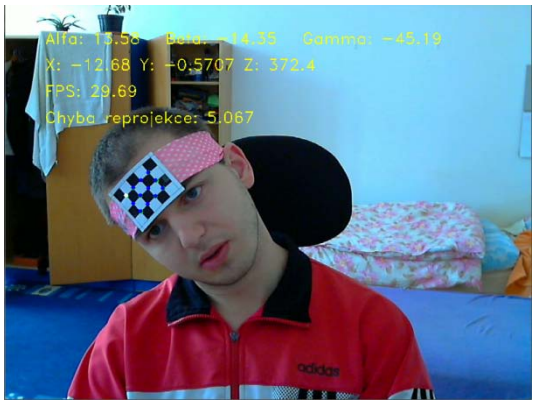
Obrázek 7.12 – průběh hodnoty velikosti úhlu beta s viditelnou korekcí



Obrázek 7.13 – průběh hodnoty velikosti úhlu bez viditelné korekce

Obrovským přínosem znalosti reprojekční chyby je možnost nastavení spuštění korekčního procesu nejen na znalosti hodnoty úhlů, ale také v závislosti na reprojekční chybě (je zbytečné provádět korekci při uspokojivé chybě). Špičky na obrázku 7.12 jsou způsobeny právě spuštěním korekčního procesu (mezní hodnota chyby pro spuštění korekce je zde 4). Naproti tomu obrázek 7.13 ukazuje průběh velikosti úhlu beta při nastavení mezní hodnoty chyby pro spuštění korekcí na 7. Ve výsledku není spuštěn korekční proces a konečná hodnota úhlu beta je prakticky shodná s hodnotou na obrázku 7.12.

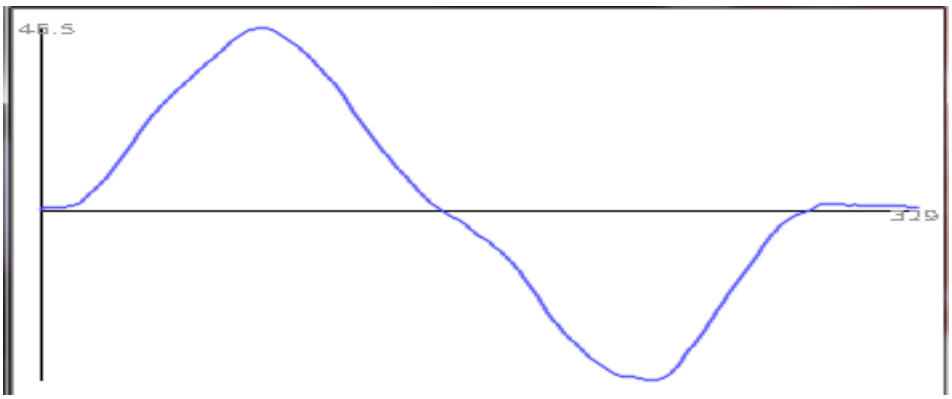
Poslední možnou rotací je rotace kolem osy z (označená jako úhel gamma). Obrázky 7.14 a 7.15 ukazují možné mezní úhly takové rotace při dodržení podmínky pevných ramen.



Obrázek 7.14 – mezní úhel gamma pohyb doleva

Obrázek 7.15 – mezní úhel gamma pohyb doprava

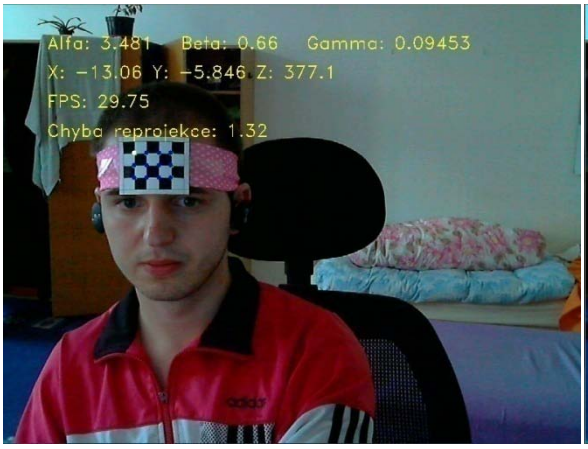
Bylo dosaženo hodnoty mezních úhlů o hodnotách 45.19° (obrázek 7.14) a 46,81 (obrázek 7.15). Vzhledem k dobré viditelnosti na sledovanou značku, je i dosažená reprojekční chyba přijatelná a to průměrně na hodnotě 4.



Obrázek 7.16 – průběh velikosti úhlu rotace gamma

7.2 Funkce translačního vektoru

Velikost translace hlavy (posuvného pohybu) po snímku zachycuje zobrazovaná hodnota translačního vektoru. Místo s nulovou hodnotou (počátek) je určeno souřadnicemi *cx* a *cy* matice kamery *M* (je to střed průmětny kamery). Jinými slovy hodnota translačního vektoru je tedy offset (posun) vztažený k počátku určeného dvěma jmenovanými prvky matice kamery *M*.



Obrázek 7.17 – pohyb do strany bez rotace

Obrázek 7.18 – snaha o nalezení středu

Po bližším prozkoumání obrázků 7.17 a 7.18 si lze všimnout, že hodnoty rotačních úhlů jsou minimální, ale hodnota translačního vektoru (X, Y a Z) se při posuvném pohybu hlavy výrazně mění. Na obrázku 7.18 je snahou nalezení místa s nulovými souřadnicemi translačního vektoru (přibližně střed snímku) a je skutečně vidět i pohledem, že se přibližně o střed snímku jedná.

7.3 Sledování rychlosti pohybu

Rychlost pohybu posuvného i pohybu při otáčení je také důležitým faktorem pro správnou funkci algoritmu. Tuto skutečnost ještě podporuje použití optického toku pro sledování pohybu bodů, který je na možnostech rychlosti snímání kamery závislý z pohledu maximální možné rychlosti pohybu při které je ještě možné body sledovat.

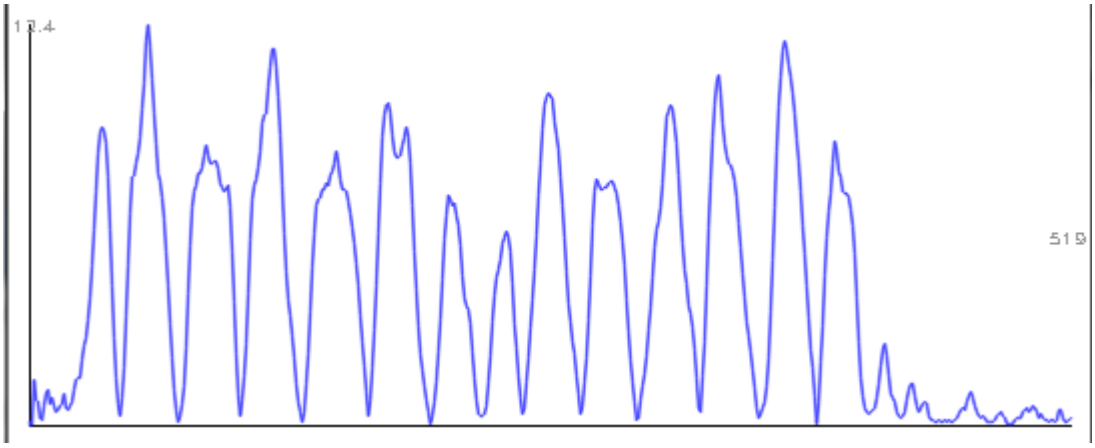
Nejprve se podíváme, jakým způsobem je potřeba rozšířit algoritmus pro umožnění výpočtu rychlosti. Z principu výpočtu optického toku jsou nám při průběhu programu hlavní smyčkou známé, jak body předchozí, tak body současně vypočítané. Využitím jejich hodnot (konkrétně odečtením hodnot x a y souhlasných bodů nových a starých) lze získat výslednou změnu připadající na jeden snímek. Následující rovnice princip osvětlí.

$$V_x = x_{nový} - x_{starý} \quad (48.)$$

$$V_y = y_{nový} - y_{starý} \quad (49.)$$

$$V = \sqrt{V_x{}^2 + V_y{}^2} \quad (50.)$$

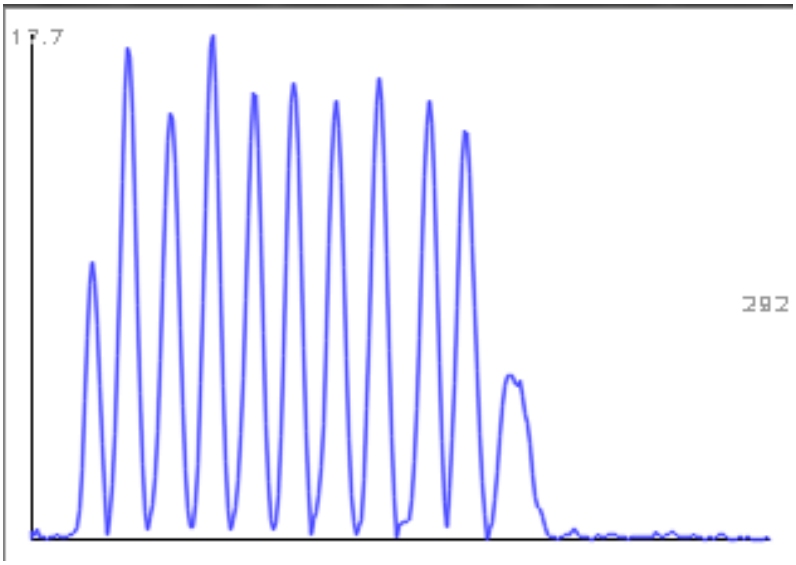
Rovnice (48.), (49.) a (50.) vyjadřují princip výpočtu změny polohy bodu v osách x a y při průběhu jednoho snímku. Z těchto dvou hodnot je posléze možné spočítat jejich výslednici (výslednici rychlosti V). V našem případě je tento výpočet aplikován na všech 12 bodů a výsledná rychlost je určena jako rychlost průměrná ze všech dílčích rychlostí (rychlostí jednotlivých bodů).



Obrázek 7.19 – velikost rychlosti změny polohy za jeden snímek při rotacích do stran a nahoru i dolů

Pořízením obrázku 7.19 si demonstrujeme velikost hodnoty rychlosti při běžném otáčení hlavou. Dosáhli jsme zde rychlosti změny 12,4 bodů na jeden snímek (jeden průběh cyklem hlavní smyčky) a to průměrně při 25 snímcích za sekundu (rychlost snímání kamery). Optimální rychlost při užití kamery Logitech C270 leží tedy někde na maximálně 14 bodech za snímek při 30 snímcích za vteřinu. Tato rychlost dává algoritmu dostatek času při průchodu značky středem k provedení korekčních procesů. Rotace jsou prováděny do 45° ve směru horizontálním a 40° ve směru vertikálním.

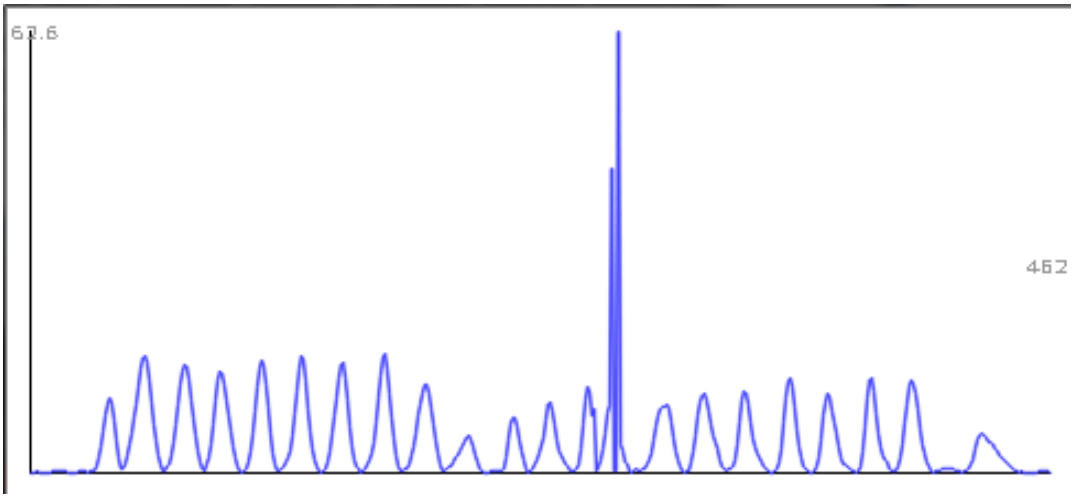
O něco lépe dopadl jen izolovaný pohyb hlavy ze strany na stranu při rotacích do 40° a tuto skutečnost zachycuje obrázek 7.20.



Obrázek 7.20 – otáčení hlavy ze strany na stranu (omezení pohybu do 40°)

Z obrázku lze odečíst hodnotu 17,7 bodů za snímek, ale „bezpečnou“ rychlostí se ukázala hodnota do 16 bodů za snímek. Pokud se zaměříme na poslední maximum na obrázku, lze si všimnou jeho značně menší hodnoty. Je to způsobeno nedostatkem času pro korekční procesy při průchodu středem, což vede k velkému nárůstu chyby a rozpadu geometrického rozložení bodů. Takový stav zachycuje obrázek 7.21.

Obrázek 7.21 – Rozpad geometrického uspořádání bodů



Obrázek 7.22 – velice rychlé změny polohy hlavy

Za zmínku závěrem stojí ještě obrázek 7.22, který demonstruje případ velice rychlých změn natočení hlavy. Průměrně se jedná o rychlost okolo 20 bodů na snímek. Taková rychlost neumožní úspěšné provedení korekčních procesů a velice brzy vede ke ztrátě a okamžitému hledání bodů šachovnice, což je ilustrováno velikým maximem obrázku nad textem.

8 ZÁVĚR

Navržený algoritmus je schopen s dostatečnou přesností (reprojekční chyba do hodnoty 7) sledovat pohyb hlavy a její natočení. Jako „optimální“ se ukázalo sledování pohybu v rozsahu 45° ve směru horizontálním a 35° ve směru vertikálním, je nutné říci, že se jedná o natočení značky připevněné k hlavě. Maximální bezpečnou rychlostí umožňující provádění korekcí polohy bodů je hodnota průměrně 13 bodů za snímek (jeden průběh hlavním cyklem programu). Byla provedena řada testů za účelem zjištění mezních podmínek korektní funkce algoritmu viz kapitola 7. Hlídáním pevné polohy ramen je dodržena podmínka zadání a v případě nutnosti je uživatel požádán o novou manuální inicializaci.

Praktické využití algoritmu již částečně demonstruje samotná vizualizace. Je možné za pomocí knihovny open GL vytvořit virtuální prostředí a na základě snímání polohy a natočení hlavy simulovat pohled po tomto prostředí. V takové případě se navíc značně omezí rozsah pohybu hlavy (uživatel potřebuje vidět na monitor) a stabilita celého algoritmu bude mnohonásobně vyšší než v krajních polohách (bude téměř pořád vidět celou značku => snadná korekce). Dalším možným praktickým využitím je například přibližný odhad směru pohledu sledované osoby.

Z použití optického toku pro sledování bodů plynou dva hlavní omezující faktory algoritmu. Prvním faktorem je schopnost kamery pořizovat snímky určitou maximální rychlostí. Zde si můžeme pomoci použitím (v případě operačního systému windows) knihovny directshow pro komunikaci s kamerou (zrychlení komunikace) a samozřejmě také zakoupením vysokorychlostní kamery. Druhým omezením funkčnosti je zorné pole kamery. Opět lze řešit dvěma způsoby a to lehkým vzdálením se od kamery (možno ovlivnit jen minimálně, protože se vzdalováním klesá přesnost) a konečně použitím vícekamerového systému, kde při dosažení mezního úhlu přejde snímání na kameru další.

Do budoucnosti se předpokládá rozšíření algoritmu o další kamery a vytvoření virtuálního prostředí za pomocí knihovny open GL k demonstraci praktického využití daného algoritmu.

SEZNAM ZKRATEK

- atd. – a tak dále
- např. – například
- apod. – a podobně
- tzn. – to znamená
- obr. – obrázek
- rce. - rovnice

BIBLIOGRAFIE

[1] **Osuchowski, Mgr. Marek.** Zpracování barev. *Výukové materiály.* [Online] [Citace: 29. listopad 2010.] http://www.osuchowski.cz/grafika/obecna/zpracovani_barev.php.

[2] **Krištof, Martin.** Výukový modul do EPO. [Online] SPŠei, Ostrava. [Citace: 29. listopad 2010.] http://www.dmp.spsei.cz/digi/model.php.

[3] **Pihan, Ing. Roman.** Reprezentace barev v PC, RGB a barevný prostor. *Digimanie.* [Online] 15. září 2006. [Citace: 29. listopad 2010.] http://www.digimanie.cz/art_doc-C40D4D3BDB59097EC12571E900639E80.html.

[4] Lidské oko. *Wikipedie.* [Online] 17. duben 2007. [Citace: 29. listopad 2010.] http://panwiki.panska.cz/index.php/Lidsk%C3%A9_oko.

[5] **Neff, Ondřej.** DigiNeff. *Co je to histogram.* [Online] [Citace: 30. listopad 2010.] http://www.digineff.cz/cojeto/ruzne/histogram.html.

[6] **Doubek, Petr.** České Vysoké Učení Technické. *Mean Shift Segmentace.* [Online] 2007. [Citace: 05. 01 2011.] http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv4/meanshift.pdf.

[7] **Bradsky, Gary a Kaehler, Adrian.** *Learning Open CV.* Cambridge : O'Reilly Media, Inc., 2008. ISBN: 978-0-596-51613-0.

[8] **Němec, Martin.** Vysoká škola báňská. *Základy počítačové grafiky, Transformace.* [Online] [Citace: 6. květen 2011.] http://barborka.vsb.cz/nemec/zpg/prednasky/prednaska2.pps.

[9] **Pollefeys, Mark.** A simple model. [Online] 22. listopad 2002. [Citace: 6. květen 2011.] http://www.cs.unc.edu/~marc/tutorial/node36.html.

[10] **Josef, Miroslav.** Čočka. *Wikipedie.* [Online] 20. dubedn 2011. [Citace: 6. květen 2011.] http://cs.wikipedia.org/wiki/%C4%8Co%C4%8Dka_%28optika%29.

[11] **Hlaváč, Václav.** ČVUT v Praze. [Online] [Citace: 15. duben 2011.] http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/21PredzpracObr.pdf.

[12] **Ho Ahn, Song.** OpenGL Transformation. [Online] 2011. [Citace: 6. květen 2011.] http://www.songho.ca/opengl/gl_transform.html#example1.

[13] **Turek, Michal a kol., a.** NeHe. [Online] 2008. [Citace: 06. květen 2011.] http://nehe.ceske-hry.cz/tut_obsah.php.

[14] **Aron a Roy.** More than Technical. *Quick and Easy Head Pose Estimation with OpenCV.* [Online] [Citace: 10. duben 2011.] http://www.morethantechnical.com/2010/03/19/quick-and-easy-head-pose-estimation-with-opencv-w-code/.

[15] **Slabaugh, Gregory G.** Gragslabugh. *Computing Euler Angles from rotation matrix.* [Online] [Citace: 13. květen 2011.] http://www.gregslabaugh.name/publications/euler.pdf.

[16] **Možný, Karel.** *Rozpoznávání znaků pomocí umělé inteligence, bakalářská práce.* [editor] Ing. Luděk Červinka. Brno : Vysoké Učení Technické, 2010.
[17] **Reynolds, Carson.** Benchmarking frames per second when using OpenCV's cvCaptureFromCAM. *Sublimated.* [Online] [Citace: 16. květen 2011.] http://sublimated.wordpress.com/2011/02/17/benchmarking-frames-per-second-when-using-opencvs-cvcapturefromcam/.
[18] **Emami, Shervin.** Shervin Emami. *Graphs in OpenCV.* [Online] [Citace: 19. květen 2011.] http://www.shervinemami.co.cc/graphs.html.

SEZNAM PŘÍLOH

Příloha č. 1 – Základní soupis ovládání programu

Příloha č. 2 – DVD obsahující program, elektronickou verzi práce a další důležité materiály

PŘÍLOHA 1

K základnímu ovládání programu slouží několik funkčních kláves, při jejichž stisku dojde ke změně stavových proměnných a díky tomu i chování programu.

Možné volby jsou:

- **Písmeno a** – slouží ke spuštění běhu algoritmu, případně zavolaní nové inicializace i při běhu algoritmu
- **Písmeno c** – slouží k zastavení běhu algoritmu a vymazaní načtených bodů
- **Písmeno u** – pro zobrazení/skrytí vypisování informace o úhlech
- **Písmeno r** – pro zobrazení/skrytí vypisování informace o reprojekční chybě
- **Písmeno f** – pro zobrazení/skrytí vypisování informace o FPS (snímky za sekundu)
- **Písmeno t** – pro zobrazení/skrytí vypisování informace o translacích
- **Písmeno v** – pro zobrazení/skrytí vypisování informace o rychlosti pohybu