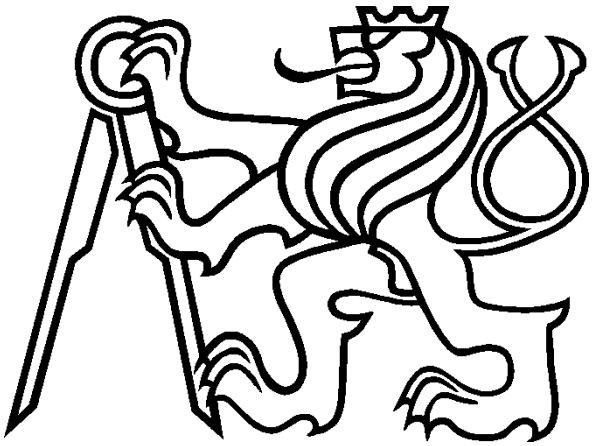
České vysoké učení technické v Praze

Fakulta elektrotechnická

BAKALÁŘSKÁ PRÁCE



Detekce zákrytu při vizuálním sledování algoritmem SSD

Petr Vávra

Vedoucí bakalářské práce: Ing. Pavel Krsek, Ph.D.

2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Petr V á v r a

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Detekce zákrytu při vizuálním sledování algoritmem SSD

Pokyny pro vypracování:

1. Seznamte se se stávajícím systémem vizuálního sledování objektů, který se používá v projektu SVICE. Pro lokalizaci objektu a možnost detekce zákrytů je zásadní především zvolená chybová funkce a způsob odhadu kvality sledování.
2. Navrhněte a realizujte algoritmus pro detekci ztráty vizuálního kontaktu se sledovaným objektem (auto projíždí pod mostem, či za stromem apod.). Je třeba detekovat jak zakrytí tak naopak opětovnou lokalizaci objektu poté, co je vizuální kontakt obnoven. Po dobu zakrytí sledovaného objektu je třeba přerušit vizuální sledování a přejít k odhadu polohy objektu na základě modelu pohybu.
3. Seznamte se s postupy pořizování referenčních testovacích dat a s již dostupnými daty. V případě potřeby dále testovací data doplňujte.
4. Proveďte experimenty potřebné pro realizaci algoritmu a ověření jeho funkčnosti. Věnujte pozornost realizaci experimentů a pečlivému zpracování výsledků. Cílem je dobře dokumentovat výsledky navrženého algoritmu a nastavení jeho parametrů.

Seznam odborné literatury:

[1] Kevin Nickels, Seth Hutchinson: Estimating uncertainty in SSD-based feature cracking. Image and Vision Computing 20, Elsevier 2002, 47-58.
[2] Quiang Zhu, Kwang-Ting Cheng, Hongjiang Zhang: SSD Tracking Using Dynamic Template and Log-polar Transforamtion. In proceedings of IEEE International Conference Multimedia and Expo (ICME 2004) Vol.1, June 2004, 723 – 726.
[3] Jianbo Shi, Carlo Tomasi: Good Features to Track. In proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR94), Seattle, June 1994, 593-600.
[4] Bruce D. Lucas and Takeo Kanade: An Iterative Image Registration Technique with an Application to Stereo Vision. In proceedings of International Joint Conference on Artificial Intelligence, 1981, 674-679.
[5] Milan Šonka, Václav Hlaváč, and Roger Boyle: Image Processing, Analysis and Machine Vision. Thomson, 3rd edition, ISBN 978-0-495-08252, 2007.
[6] Ristic Branko, Arulampalam Sanjeev and Gordon Neil: Beyond the Kalman Filter. Particle Filters for Tracking Applications. Artech House, ISBN 1-58053-631-X, 2004.

Vedoucí bakalářské práce: Ing. Pavel Krsek, Ph.D.

Platnost zadání: do konce zimního semestru 2012/2013

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 19. 12. 2011

Abstrakt

Bakalářská práce se zabývá detekcí zákrytu při vizuálním sledování algoritmem SAD (Sum of Absolute Differences), který je modifikací algoritmu SSD (Sum of Square Differences). Tato detekce je důležitá pro včasné zastavení sledovacího algoritmu a zajištění predikce polohy objektu, aby mohl být objekt opět nalezen za překážkou. Algoritmus pro detekci ztráty cíle budu navrhovat pomocí chybové funkce. Nejprve algoritmus SAD otestuji na připravených videosekvencích. Poté se pokusím navrhnout práh pro detekci ztráty cíle na základě hodnoty chyby z chybové funkce. Nakonec tento práh otestuji na skutečných videosekvencích pořízených z kamery.

Abstract

This thesis deals with the detection of occlusion in visual tracking algorithm SAD (Sum of Absolute Differences), which is a modification of the algorithm SSD (Sum of Square Differences). The detection is important for the early cessation of the tracking algorithm. Early detection and object position prediction ensure that the object could be found again behind the barrier. Algorithm to detect loss targets I propose using the error function. First SAD algorithm was tested on prepared videos. Then I try to design a detection threshold for loss targets based on the value of the value of the error function. Finally, the threshold is tested on real video sequences captured from the real camera.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 25. 5. 2012 .....

.....

podpis

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Pavlu Krskovi, Ph.D. za trpělivost a cenné rady při vytváření tohoto projektu. Děkuji také své rodině, která mě podporovala během studia.

Obsah

Úvod

V dnešní době se kamery často používá k pozorování a sledování za různým účelem. Pozorování je při tom vedeno jak ze staticky umístěných kamer tak z kamer umístěných na pohybujícím se nosiči (automobil, letadlo). Stále převládají vojenské aplikace a aplikace v oblasti zajištění bezpečnosti střeženého území. Větší měrou se však využívá kamer také pro účely hospodářské (sledování dálkových rozvodů, kontrola) či pro řízení dopravy.

S využitím kamer, které umožňují měnit směr pozorování, přichází ke slovu také sledování vybraného objektu. Sledování objektu je pak již nezbytným úkonem pokud se pro pozorování používá mobilního nosiče, kde dochází ke vzájemné změně polohy sledovaného objektu a pozorovatele. Pokud by v takovém případě nebylo zajištěno sledování, nebylo by prakticky možné provádět dlouhodobější sledování. Sledování vybraného objektu může zajistit operátor zařízení manuálně. Jde však o úkol, který částečně odvádí operátora od samotného pozorování. Pokud se podaří zajistit sledování vybraného objektu automaticky, sníží se zatížení operátora a zefektivní proces sledování.

Z uvedených důvodů je potřeba obraz stabilizovat a umět sledovat vybraný objekt. Automatické sledování zajišťují algoritmy sledování, které jsou dnes již součástí nasazovaných zařízení. V průběhu sledování ovšem může dojít k zakrytí sledovaného objektu překážkou (např. auto projíždí pod mostem, za stromem …). Zakrytí sledovaného objektu je potřeba včas detekovat, zastavit algoritmus sledování. Po dobu zakrytí je poloha objektu predikována na základě jeho pohybu před zakrytím, aby mohl být objekt opět zachycen za překážkou. Pokud není sledování včas přerušeno, může se stát, že je zachycen jiný objekt, který bude nadále sledován a původní objekt už za překážkou nebude nalezen.

Hlavním motivací pro zadání mé bakalářské práce byla potřeba správně a včas detekovat zákryt sledovaného objektu v již běžícím projektu SVICE. Mým úkolem tedy bylo navrhnout algoritmus pro detekci zákrytu na základě zvolené chybové funkce. Na základě této chyby, bylo mým úkolem vytvořit statický nebo dynamický práh pro detekci ztráty cíle.

Celá bakalářská práce je programována v programu MATLAB, je sice pomalejší než některé jiný programovací jazyky např. C, C++, ale je velice snadný na práci a mám s programováním v něm velké zkušenosti na rozdíl od ostatních programovacích jazyků, což mi velmi usnadní a urychlí práci.

Kapitola 1

Algoritmy sledování

Základem sledování objektu je detekce objektu v daném snímku. Existuje mnoho algoritmů pro detekci objektů v různých podmínkách. Můžeme je rozdělit do několika skupin podle základního přístupu:

Algoritmy porovnávající model s pořízeným snímkem. Model je definován obvykle jako obraz hledaného objektu. Porovnáním modelu s částí snímku získáme hodnotu chyby. Chyba je definována jako míra rozdílu mezi modelem a snímkem v určitém místě. Vyhodnocením chyby na různých místech obrazu získáme chybovou funkci, kterou vyhodnocujeme pro všechny možné polohy hledaného objektu. Algoritmus hledá místa, kde je chyba minimální. Pozice s nejmenší chybou je pozicí hledaného objektu. Obraz objektu se bohužel v průběhu sledování obvykle mění (jas, nemodelovaná geometrie …). Jednou z možností jak tyto změny začlenit do algoritmu sledování je postupná aktualizace modelu.

Vyhodnocování funkce ve všech polohách je časově náročné. Proto zavádíme omezení prohledávaného prostoru. Hlavním zjednodušením je omezení prohledávaného prostoru jen na okolí předpokládané polohy objektu v následujícím snímku. Další možné zjednodušení je zanedbání natočení či změny velikosti obrazu objektu. Tyto změny jsou obvykle kompenzovány aktualizací modelu objektu. Zástupcem tohoto druhu algoritmu je např. SSD (Sum of Square Differences) algoritmus [1] či základní část algoritmu K-L (Kanade-Lucas) [2] .

Algoritmy popisující objekt jsou založeny na získávání určitých číselných hodnot (příznaků) ze snímku a stanovení sledovací funkce. Cílem je získání malého počtu příznaků, které dostatečně popisují objekt a mohou být nalezeny v následujícím snímku. Existuje mnoho způsobů, jak definovat tyto příznaky a sledovat je. Mean-shift algoritmus dle [3] je založen na porovnávání barevných histogramů objektu a příslušné části snímku. Byla už ale vyvinuta řada obdobných algoritmy, jako například KLT (Kanade-Lukas-Tomasi tracker) [4] nebo SIFT (Scale invariant Feature Transform) [5] .

Algoritmy založené na rozpoznávání využívají algoritmů strojového učení. Používají standardní algoritmy klasifikace objektu na základě učení z příkladů. Jako pozitivní příklady pro učení se používají snímky objektu v jednotlivých snímcích zpracovávané video sekvence. Negativní příklady naopak tvoří části obrazů z okolí objektu. Detekce objektu je provedena příslušným klasifikátorem. Klasifikátor klasifikuje jednotlivé části (vzorky) obrazu odpovídající velikosti objektu do dvou tříd – okolí a objekt. Poloha vzorku klasifikovaného jako objekt je novou polohou sledovaného objektu. Poté je obvykle objekt a zbytek snímku použito k doučení klasifikátoru. Znalosti o objektu se tak zvětšují s postupným doučováním. Příkladem takového algoritmu je On-line Boost popsaný například v [6]

Uvedené algoritmy se liší v možnostech sledování různých typů objektu, požadovaných podmínkách, časové náročnosti a mnoha dalších charakteristikách. Já budu používat algoritmus SAD (Sum of Absolute Differences), který se používá v projektu SVICE. Algoritmus je snadno implementovatelný a umožňuje sledovat i malé objekty. Nevýhodou SAD je časová náročnost. Samotný SAD algoritmus je v mém případě doplněn také aktualizací modelu.

Kapitola 2

Popis algoritmu

Jelikož se v projektu SVICE jako algoritmus pro sledování objektu používá algoritmus SAD, který je modifikací algoritmu SSD. Budu s ním v této bakalářské práci také pracovat. Algoritmus SAD je výpočetně méně náročný než algoritmus SSD, protože se nepočítá druhá mocnina, ale pouze absolutní hodnota. Vstupem do tohoto algoritmu je sekvence snímku z kamery, v níž operátor vybral objekt, který chce sledovat. A výstupem je poloha objektu v jednotlivých snímcích sekvence.

Jako model objektu používáme uspořádanou n-tici intenzit $T(\overrightarrow{x_1}),\ldots,T(\overrightarrow{x_n})$ obrazových bodů $\overrightarrow{x_1},\ldots,\overrightarrow{x_n}$, kterou nazýváme vzorem *T*. V našem případě tvoří obrazové body čtvercovou oblast *X* (oblast zachycení) o rozměrech *t*×*t* obrazových bodů. Její velikost nastavuje operátor podle velikosti objektu, který chce sledovat. Pro lokalizaci objektu se používá chybová funkce vyjadřující podobnost mezi vzorem a pozorovanými intenzitami $I(\vec{u}+\overrightarrow{x_1}),\ldots,I(\vec{u}+\overrightarrow{x_n})$ ve snímku *I* na pozici $\vec{u}$. Chybovou funkci můžeme zapsat takto:

$$f(I,T,\vec{u})=\sum_{i=1}^{n}|I(\vec{u}+\overrightarrow{x_i})-T(\overrightarrow{x_i})| \tag{2.1}$$

Lokalizace objektu je hledáním minima chybové funkce:

$$\overrightarrow{u_j}=arg\min_{\vec{u}\in U} f(I_j,T,\vec{u}) \tag{2.2}$$

kde *U* je množina prohledávaných pozic a *j* je index zpracovávaného snímku. V případě naší implementace představuje *U* čtvercovou oblast, jejímž středem je predikovaná poloha sledovaného objektu. Chyba, se kterou byla nová pozice $\overrightarrow{u_j}$ nalezena se vypočítá:

$$g=f\left(I_j,T,\overrightarrow{u_j}\right) \tag{2.3}$$

kde $\overrightarrow{u_j}$ je nový střed oblasti zachycení *X* na snímku *j*. Právě na základě hodnoty této funkce budeme hledat algoritmus pro detekování ztráty sledovaného objektu. Je to nejjednodušší a výpočetně nejméně náročné řešení. Dále budeme používat pro porovnání průměrnou hodnotu chyby *g* na jeden obrazový bod:

$$e=\frac{g}{n} \tag{2.4}$$

kde *n* je počet obrazových bodů oblasti zachycení *X*.

Z důvodu změny cíle (změna osvětlení, zaostření, zatáčení vozidla…) je potřeba vzor *T* aktualizovat, pro tento účel se používá vzorec:

$$T_{j+1}(\overrightarrow{x_i})=(1-\alpha)\cdot T_j(\overrightarrow{x_i})+\alpha\cdot I_j(\vec{u}_j+\overrightarrow{x_i}) \tag{2.5}$$

kde α definuje rychlost adaptace modelu. α volíme malé od 0,1-0,02, aby se vzor rychle neměnil. Pokud bychom α zvolili příliš velké, obraz vzoru by se měnil příliš rychle a mohlo by se stát, že by algoritmus zachytil jiné objekty. Naopak pokud by bylo α příliš malé, obraz vzoru by se téměř neměnil a mohlo by dojít ke ztrátě sledovaného objektu při běžných změnách.

Kapitola 3

Testování algoritmu

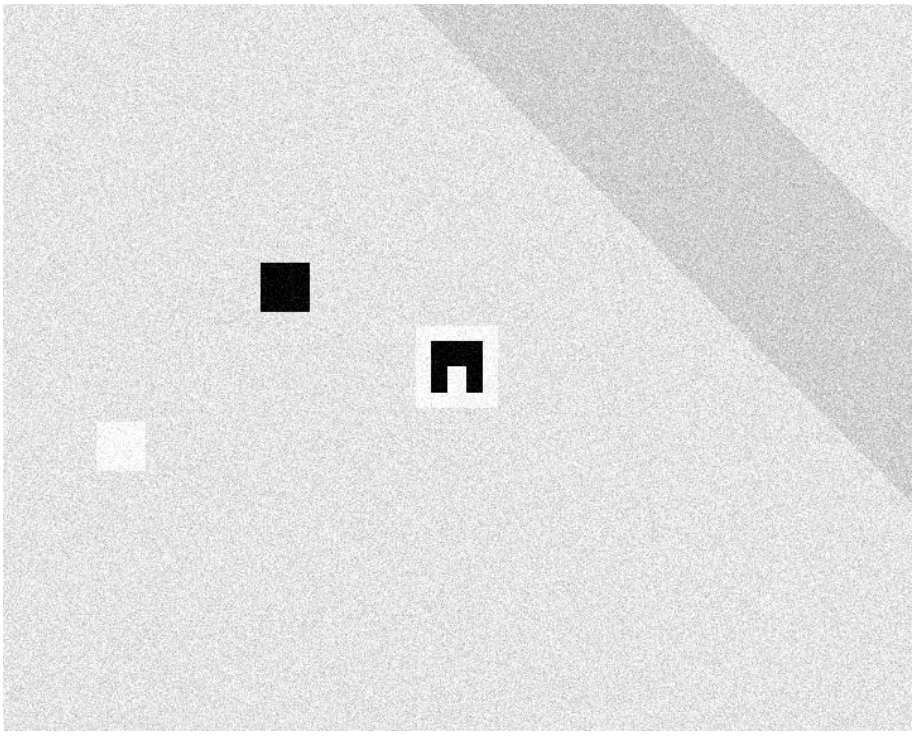
V této kapitole budu testovat funkčnost algoritmu SAD za různých podmínek, abych zjistil jaký na něj má vliv zásah operátora, kvalita kamery či druh sledované scény. Operátor má možnost nastavit velikost oblasti zachycení, proto otestuji, jaký vliv má chybně nastavená velikost na funkčnost algoritmu. Každá kamera poskytuje snímky s rozdílnou velikostí šumu, proto otestuji také, jaký vliv má velikost šumu na funkčnost algoritmu. Jelikož je potřeba včas zastavit sledování cíle, otestuji, zda dochází ke změně predikované polohy již při částečném zakrytí cíle. Nakonec otestuji, jaké výsledky poskytuje algoritmus pro RGB snímky a pro černobílé snímky vzniklé transformací z RGB snímků. Důvodem proč toto testovat je, že zpracování černobílého snímku je výpočetně méně náročné než zpracování RGB snímku.

3.1 Vytvoření testovacích dat

Nejprve bylo potřeba vytvořit vlastní testovací sekvenci snímků pro testování algoritmu a jeho různých modifikací. Jelikož algoritmy sledování transformují RGB snímky na černobílé snímky, kvůli menší výpočetní náročnosti. Vytvořím si testovací snímky pouze ve stupních šedi. Vytvořil jsem n-tici o rozměrech 576x720 (rozměry skutečných testovacích snímků z kamery) s určitým stupněm šedi, jako prostředí. Doprostřed jsem jako sledovaný objekt umístil černobílí obrazec o rozměrech 64x64, který je možno dobře detekoval (s kontrastními přechody). Přes něj přejíždí překážka s jiným stupněm šedi než je okolí, což nám bude simulovat zákryt. V průběhu experimentů se ukázala původní sekvence příliš jednoduchá pro sledování. Proto jsem doplnil různě kontrastní čtverce pohybující se dostatečně blízko sledovaného objektu, simulující např. auta projíždějící okolo sledovaného objektu. Jelikož tyto vytvořené snímky jsou bez jediné vady či rozostření, byla by chyba g z rovnice (2.3) rovna nule. Toto není pro zkoumání kvality algoritmu žádoucí. Proto jsem musel do snímku uměle zavést šum. Podle práce pánů J. Šindeláře a V. Smutného [7] můžu přidat šum s normálním rozdělením, které je dostatečně dobrou aproximací Poissonova rozdělení, které je ve skutečném snímku. Nová intenzita vznikne přičtením šumu s normálním rozdělením k původní intenzitě:

$$I_n = I_p + N(0, k) \tag{3.1}$$

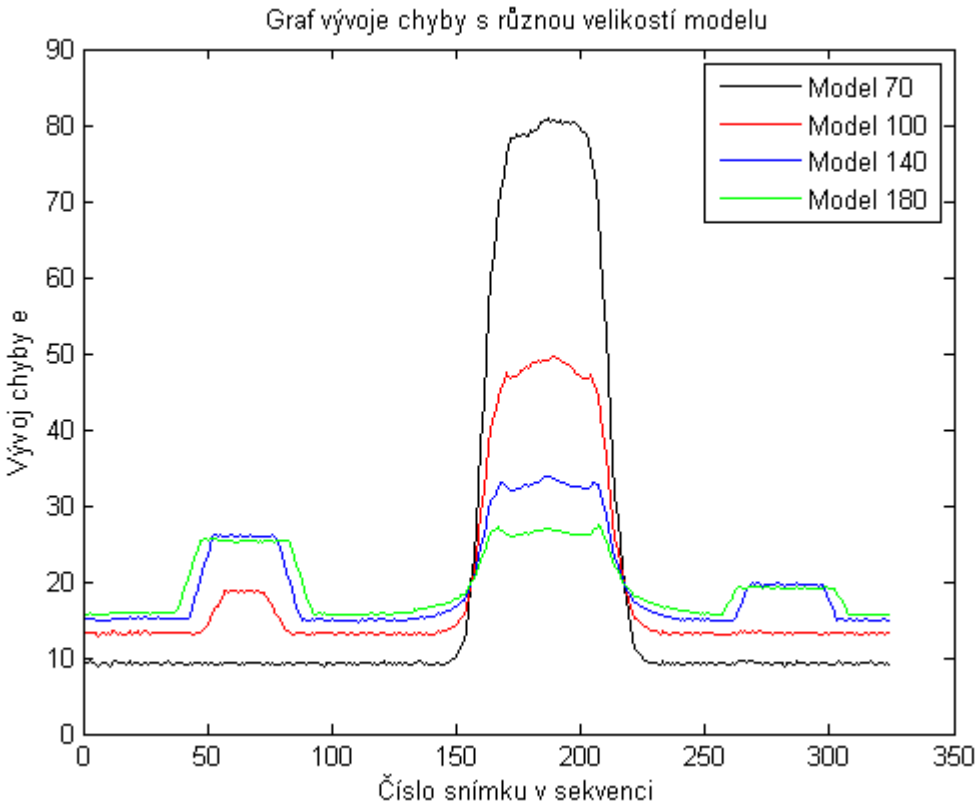
kde I_p je původní intenzita jasu obrazových bodů a N je šum s normálním rozdělením s nulovou střední hodnotou a k je rozptyl. Jelikož mám 255 úrovní jasu (0 pro černou a 255 pro bílou), budu rozptyl volit 5, 15, 25 a 40. Na obrázku 3.1 je pro představu zobrazen 1 snímek z testovací sekvence



Obr. 3.1 Ukázka vytvořeného snímku s rozptylem 15

3.2 Velikost oblasti zachycení

Jelikož má operátor možnost měnit velikost oblasti zachycení (model), je potřeba otestovat, jak chybně nastavená velikost zhorší funkčnost algoritmu a co se stane s hodnotou chyby při této změně. Pokud se nastaví příliš velký model, může se stát, že se v této oblasti budou kromě požadovaného objektu vyskytovat i jiné objekty. Bude nás také zajímat, jaký vliv mají v takovém případě míjející objekty. Jestli dokonce nezpůsobí ztrátu cíle.



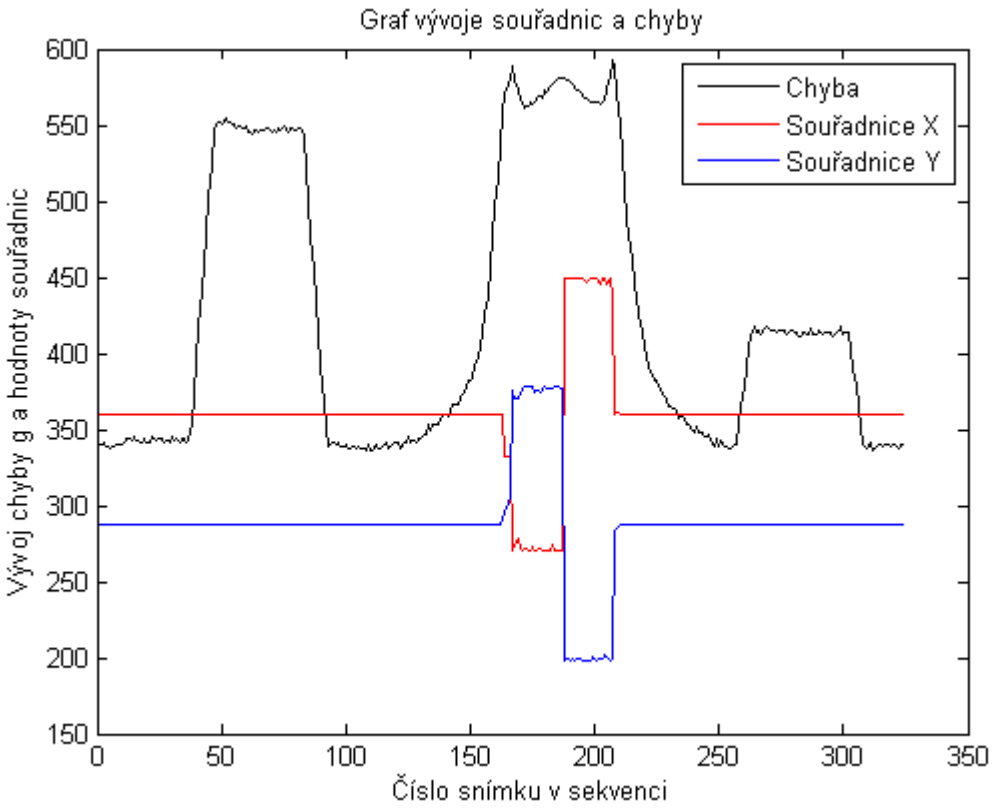
Obr. 3.2 Graf vývoje chyby s různou velikostí modelu pro šum s rozptylem 15

Na obrázku 3.2 vidíme vývoj chyby *e* z rovnice (2.4) s různou velikostí modelu. Jedná se o mnou vytvořenou testovací sekvenci s uměle zavedeným šumem s rozptylem 15. Jelikož se jedná o vytvořený obrázek, chyba by měla být rovna velikosti uměle zavedeného šumu. Jak je vidět na obrázku chyba se pro všechny velikosti modelu při sledování objektu bez míjení blíží k hodnotě 15, což je rozptyl šumu. U modelu 70 je chyby nižší, protože většina modelu tvoří černobílí objekt, dochází zde k saturaci (černá nemůže být tmavší a naopak bílá nemůže být světlejší). Naopak u modelu 180, kde černobílý objekt tvoří pouze malou část modelu, je chyba rovna 15. Z těchto poznatků vyplývá, že funkčnost algoritmu nijak neovlivňuje velikost modelu.

Dále na obrázku 3.2 vidíme kromě oblasti zákrytu další dvě zvýšení hodnoty chyby, které jsou způsobeny tím, že příliš velký model sledované oblasti zasáhnou míjející objekty.

3.3 Chyba určení polohy částečně zakrytého cíle

Jak jsem uvedl, je potřeba otestovat vývoj souřadnic X a Y při částečném zakrytí sledovaného objektu (cíle). Potřebujeme včas zastavit sledovací algoritmus před vychýlením predikované polohy, které by mohlo nastat během částečných zákrytu, buď menšími cizími objekty v oblasti zachycení (např. projíždějící auto) nebo postupným zakrýváním objektu velkou překážkou (např. mostem).

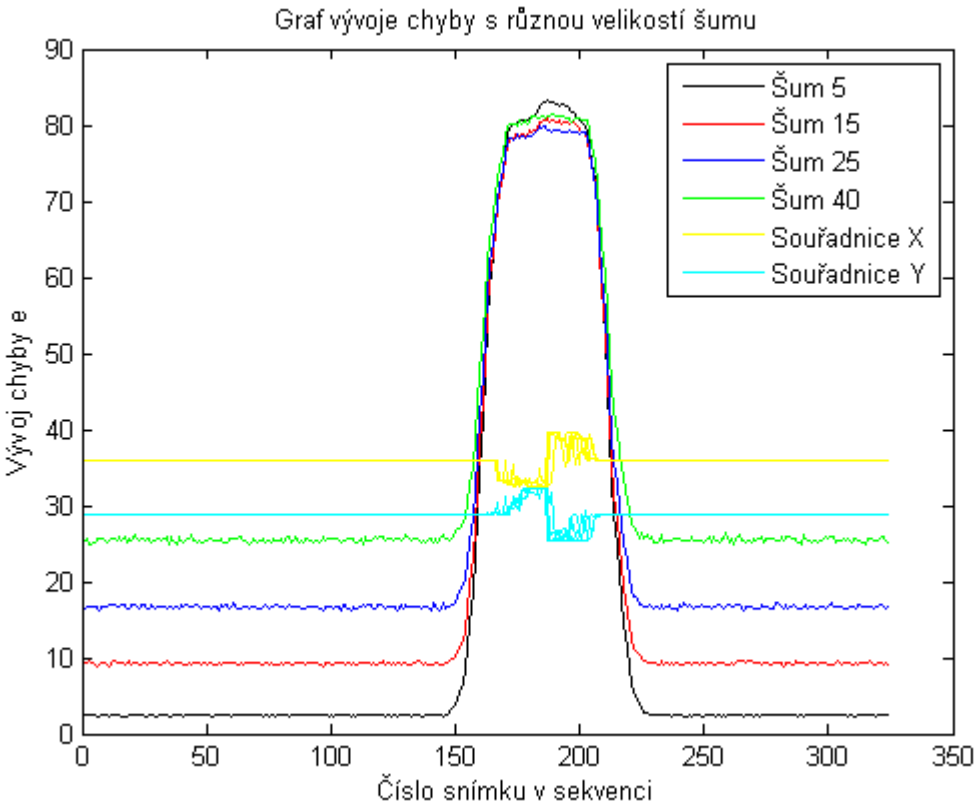


Obr. 3.3 Graf vývoje souřadnic a chyby pro model 180 a šum s rozptylem 15

Na obrázku 3.3 vidíme vývoj souřadnic X a Y, v závislosti na hodnotě chyby *g* z rovnice (2.3), která je vydělena hodnotou 1500 pro přehlednost grafu. Je vidět, že cizí objekty zvýší hodnotu chyby, ale nezmění souřadnice cíle. Souřadnice se mění až při zakrytí většiny cíle překážkou. Z toho vyplývá, že algoritmus sledování stačí zastavovat až při vysokých hodnotách chyby.

3.4 Velikost šumu

Jedna kamera při různých světelných podmínkách a nastavení citlivosti zaznamenává snímky s různým šumem. Proto je potřeba otestovat algoritmus pro různě velký šum. Jak bylo zmíněno v kapitole 3.1, je šum s normálním rozdělením s různým rozptylem dostatečnou aproximací šumu kamery pro účely tohoto experimentu.



Obr 3.4 Graf vývoje chyby s různou velikostí šumu pro model 70

Na obrázku 3.4 jsou do jednoho grafu zaneseny vývoje chyby *e* z rovnice (2.4) pro šum s rozptylem 5, 15, 25 a 40. Dále jsou na obrázku souřadnice X a Y pro všechny hodnoty rozptylu šumu. Hodnota chyby se nerovná velikosti šumu z důvodu saturace jako v kapitole 3.2. Zde je dokonce vidět, že čím je větší rozptyl v rovnici (3.1), tím je větší rozdíl mezi šumem a hodnotou chyby. Ale i přes tyto rozdíly, jak je vidět z vývoje souřadnic X a Y, že ani hodnota šumu nemá na funkčnost algoritmu vliv.

3.5 RGB vs. Černobílý snímek

Nyní již můžeme přistoupit k testování algoritmu na skutečných sekvencích snímků z kamery. Skutečné snímky jsou barevné neboli RGB snímky. Jeden obrazový bod nemá jen jednu intenzitu šedi jako u černobílého snímku, ale 3 intenzity pro každou barvu jednu. Jednou z možností je použít rovnici (2.1) pro intenzitu každé barvy a následně sečíst, což je výpočetně náročné. Jednodušší variantou je transformovat snímek na černobílý za pomoci váhového součtu:

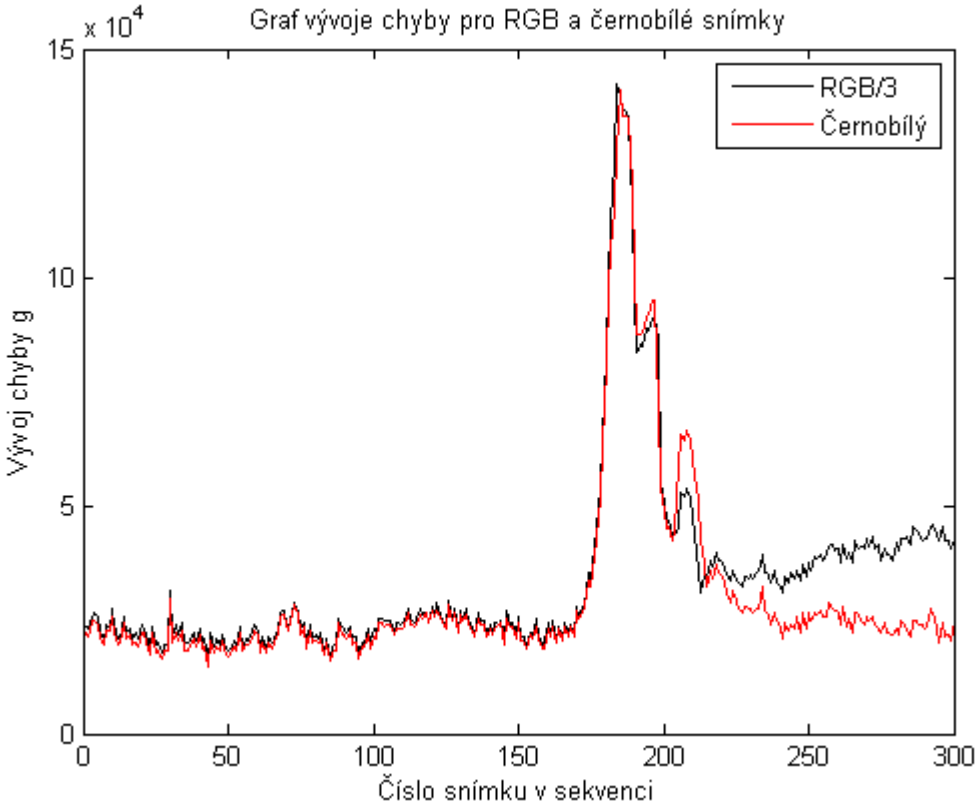
$$I = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B \qquad (3.2)$$

Je potřeba otestovat tento algoritmus pro RGB snímky i pro transformované černobílé snímky a porovnat je, zda dostaneme stejné výsledky.

Obr. 3.5 (a) RGB snímek

Obr. 3.5 (b) Černobílý snímek

Na obrázku 3.5 (a) je pro příklad zobrazen snímek původní RGB sekvence z kamery a na obrázku 3.5 (b) je zobrazen snímek černobílé sekvence transformované z původní RGB sekvence. Nyní porovnáme výsledky algoritmu pro obě tyto sekvence.

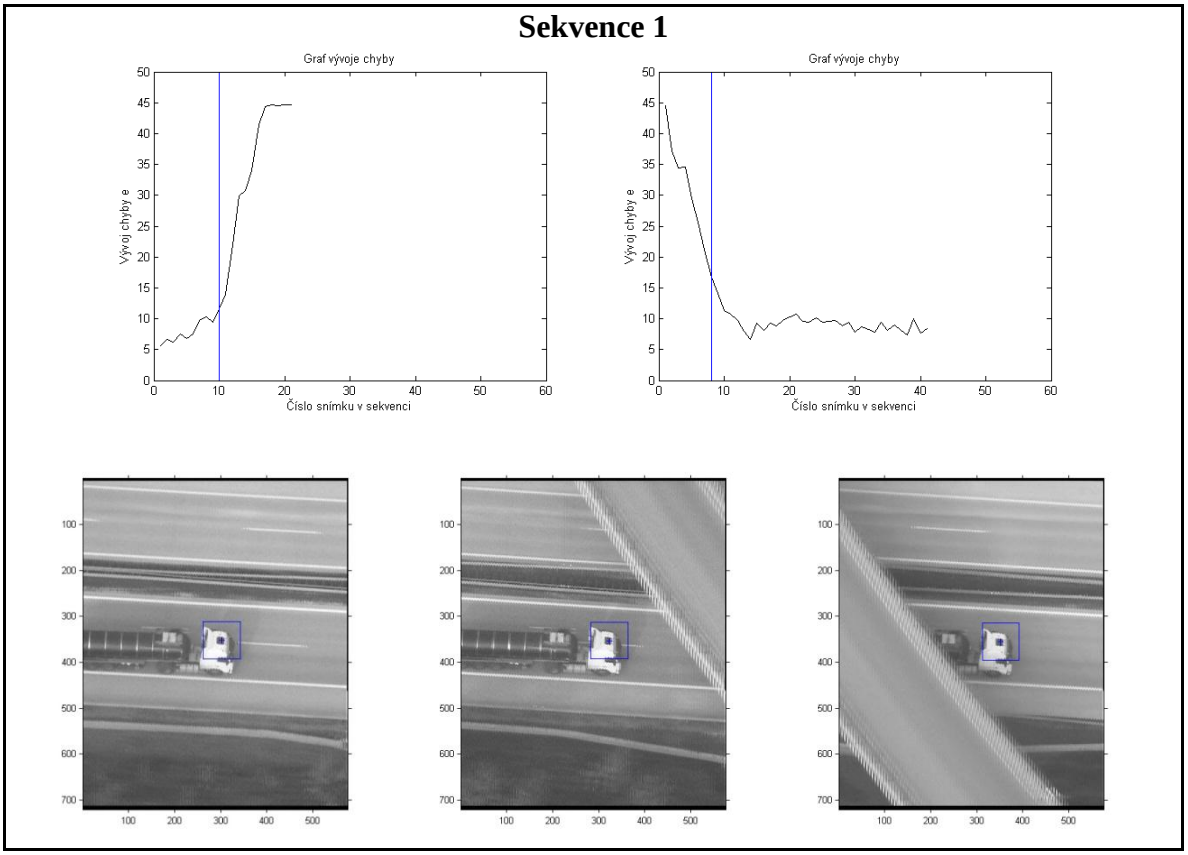


Obr 3.6 Graf vývoje chyby pro RGB a černobílou sekvenci

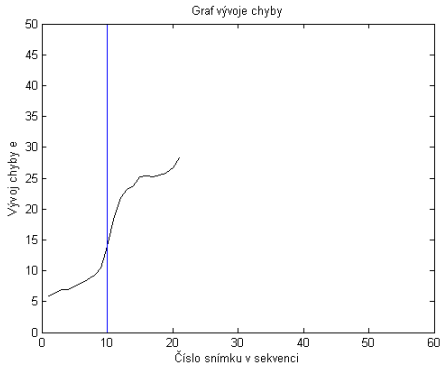
Z obrázku 3.6 vidíme graf vývoje chyby pro RGB snímky a černobílé snímky. Hodnota chyba pro RGB snímky je vydělena hodnotou 3, protože hodnota chyby je součet chyby z rovnice (3.1) pro všech tří barev, teda hodnota chyby oproti černobílým snímkům je 3krát vyšší. Z grafu je vidět že RGB i černobílé snímky mají podobnou hodnotu chyby. Algoritmy detekce zákrytu budou pravděpodobně přenosné. Proto můžeme dále používat transformaci RGB sekvence na černobílou sekvenci a tím snížit výpočetní náročnost algoritmu.

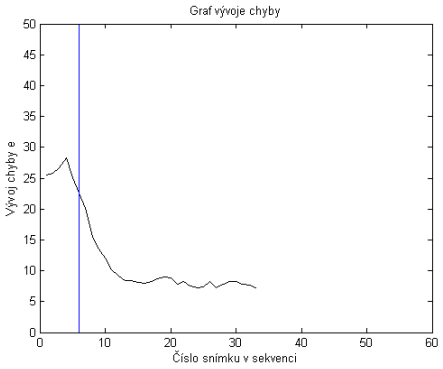
3.6 Zkoumání vývoje chyby při zákrytu

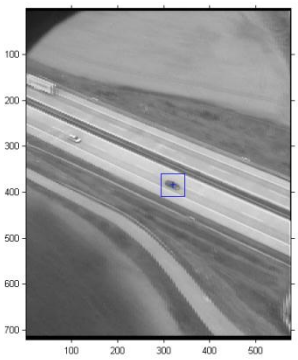
Nyní je zapotřebí prozkoumat jaký je vývoj chyby při zákrytu a při opětovném objevení cíle. Proto jsem si vykreslil grafy hodnoty chyby *e* z rovnice (2.4) pro 9 vybraných sekvencí těsně před zákrytem a těsně po něm. Svislou čarou v grafu jsem označil, kdy začíná zákryt a kdy se cíl znovu celý objeví. V prvním sloupci jsou vloženy grafy náběžné hrany těsně před zákrytem, ve druhém sloupci jsou vloženy grafy sestupné hrany při znovu objevení cíle a na dalším řádku jsou vloženy 3 snímky ze sekvence, abychom viděli, co sledujeme.

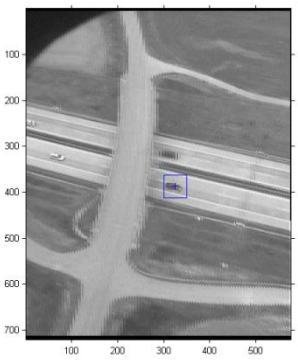


Sekvence 2

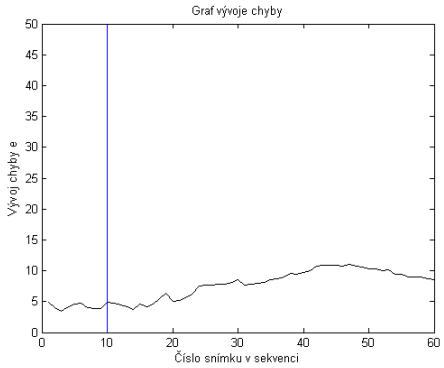


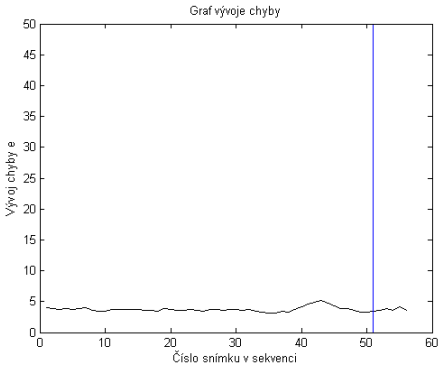


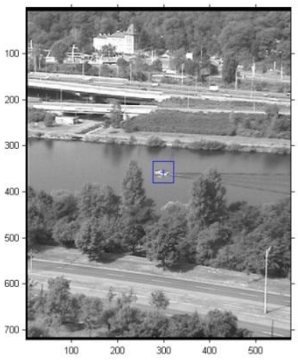


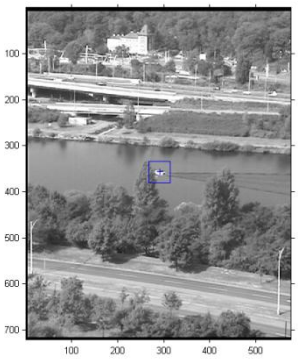


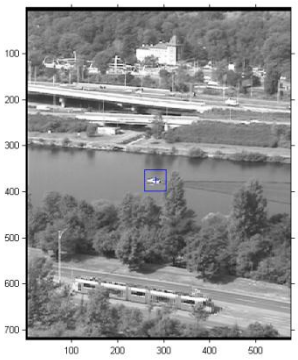
Sekvence 3



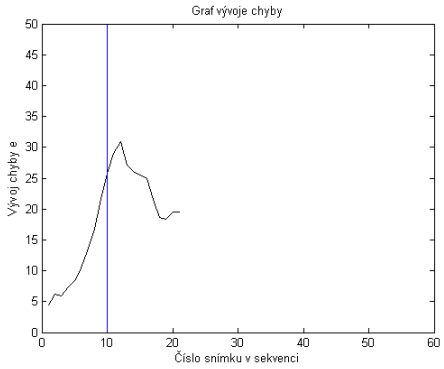


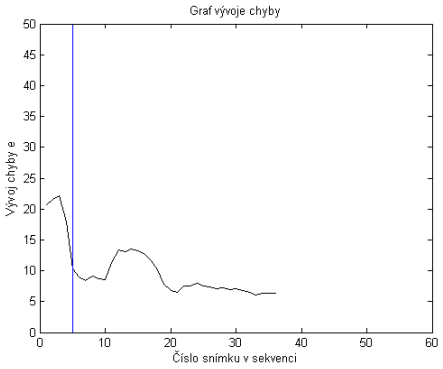


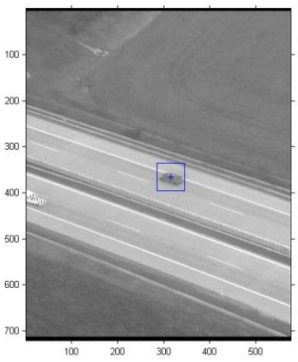


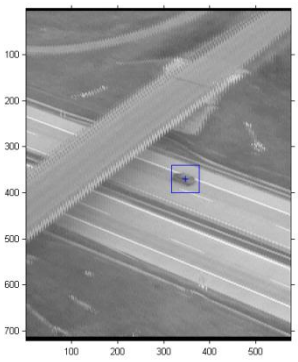


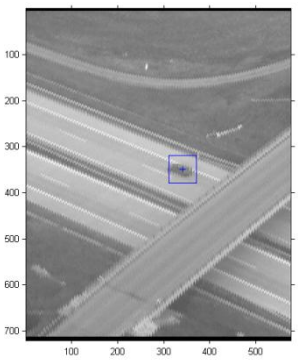
Sekvence 4



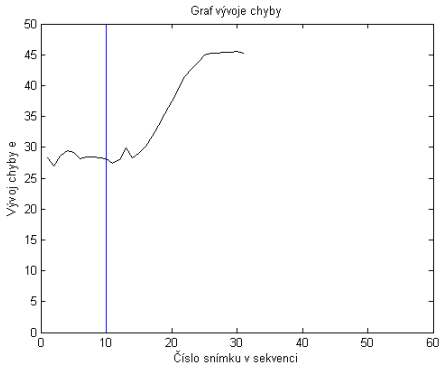


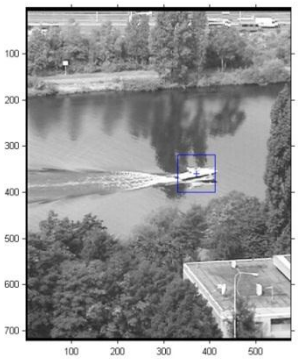


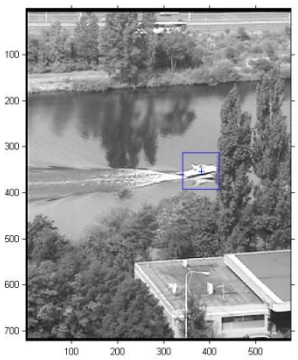




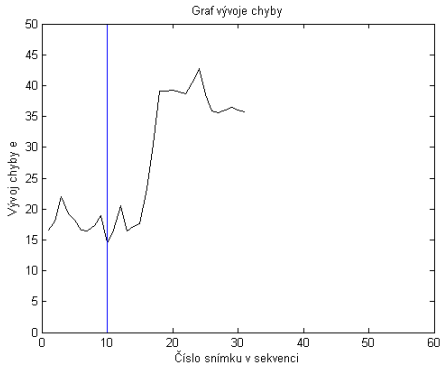
Sekvence 5

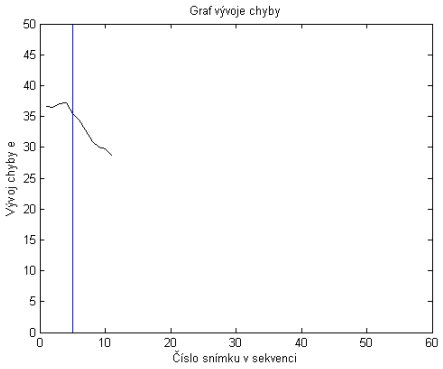




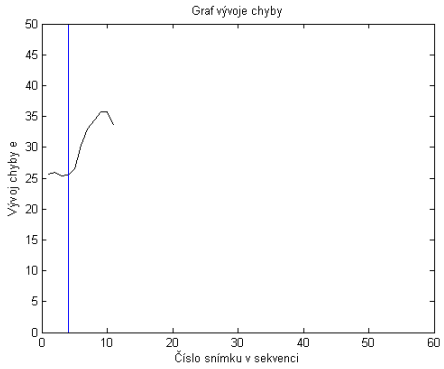


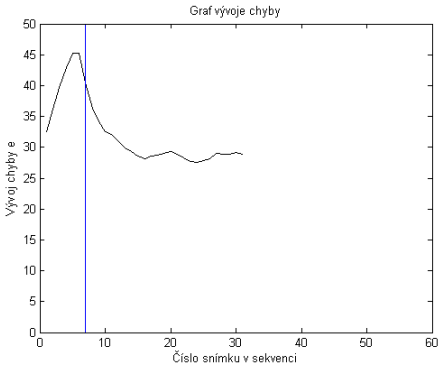
Sekvence 6

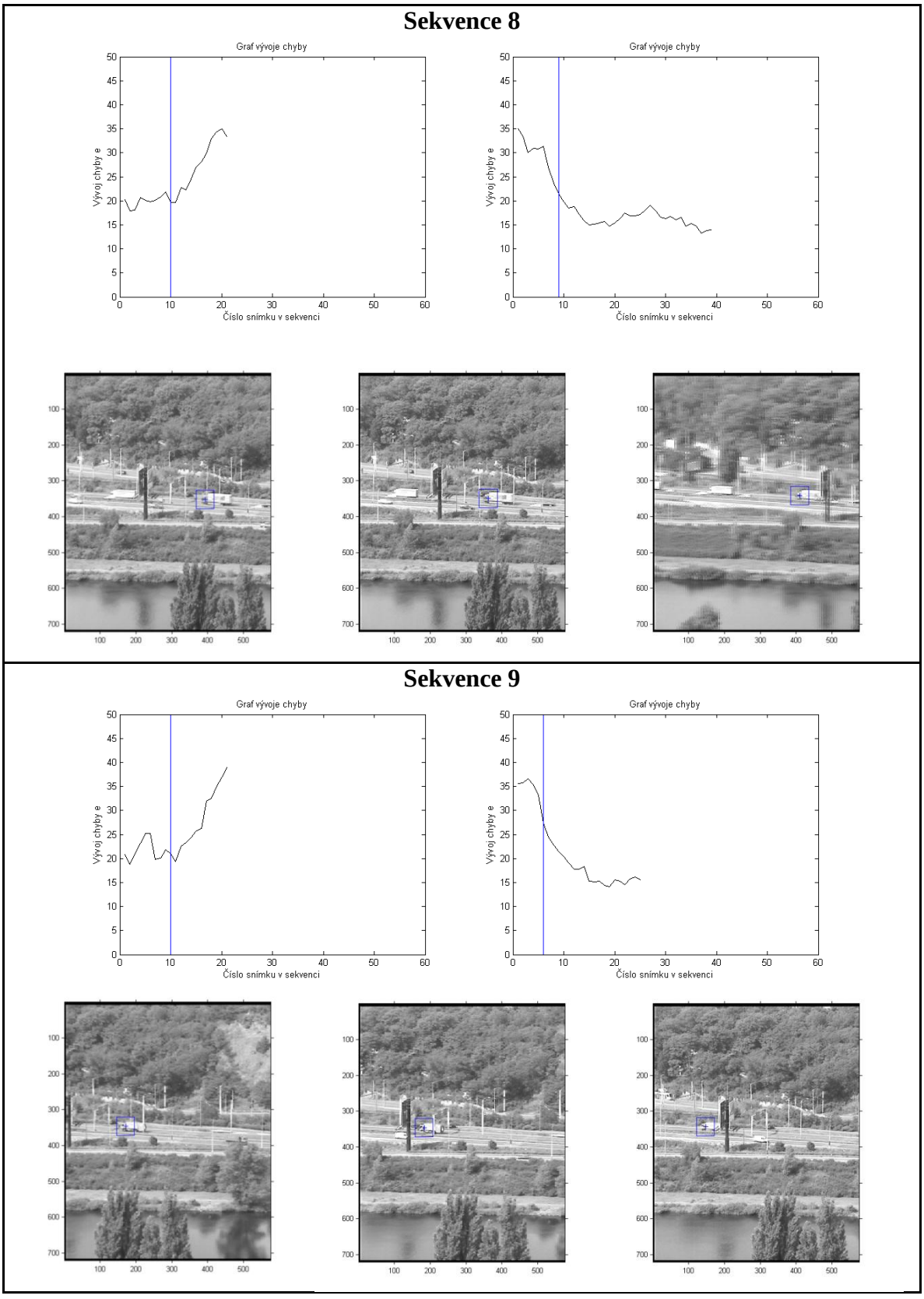




Sekvence 7







Tab. 3.1 Průběhy chybové funkce pro zakrytí a odkrytí cíle

V tabulce 3.1 jsou zobrazeny grafy vývoje chyby *e* z rovnice (2.4) při zakrytí a opětovném nalezení cíle v různých sekvencích. Pod grafy jsou 3 snímky ze sekvence, aby bylo poznat, o jakou sekvenci se jedná. Svislými čarami jsou označeny začátky zákrytu u náběžných hran a úplné opětovné objevení cíle u sestupných hran. U sekvence 5 chybí graf sestupné hrany, protože video sekvence obsahuje jen zakrytí. Zákryty v sekvenci 6 a 7 následují hned po sobě, proto jsou sestupná hrana v sekvenci 6 a náběžná hrana v sekvenci 7 zobrazeny na tak krátkém časovém úseku.

Z porovnání náběžných hran je vidět, že se jejich průběh i průměrný chyba před zákrytem a během zákrytu velice liší. Což je způsobeno různým druhem kamer. Sekvence 1, 2 a 4 jsou natáčeny kamerou Sony FCB EX 480 [8] . Jedná se o kvalitní kameru, ve stabilizovaném závěsu, upevněnou na vrtulníku a ovládanou počítačem. Obraz je tedy ostrý a stabilizovaný. Z tohoto důvodu je průměrná chyba velice malá a náběžná hrana velmi strmá. Sekvence 3, 5, 8 a 9 jsou natáčeny kamerou Sony FCB IX11 [9] . Opět se jedná o kvalitní kameru, která je upevněna v zařízení zajišťující stabilizaci a ovládaná počítačem. Obraz je také poměrně ostrý a stabilizovaný. Ovšem chyba už je o něco větší a náběžná hrana není tak strmá. Sekvence 6 a 7 jsou natáčeny kamerou Canon XM 2 [10] Jedná se o nejkvalitnější tzv. „amatérskou kameru“, což znamená, že má kvalitní obraz i rozlišení, ovšem je držena v ruce. Obraz není stabilní, dochází k častému rozostření a velkým pohybům kamery, které způsobují rozmazání a zhoršení kvality obrazu. Důsledkem je větší chyba.

Kromě druhu kamery je rozdíl v průběhu náběžných hran a průměrné chybě způsoben také druhem scény a osvětlením. Když se podíváme na sekvence 3 a 5, které jsou natáčeny stejnou kamerou, tak je vidět, že i v tomto případě se průměrná chyba a průběh náběžné hrany velice liší.

Kapitola 4

Algoritmus detekce zakrytí objektu

Jelikož už jsme prozkoumali vývoj chyby algoritmu SAD pro různé podmínky (velikost oblasti zachycení, velikost šumu, částečné zakrytí cíle) a dostali jsme uspokojivé výsledky. Můžeme v této kapitole vytvořit algoritmu pro detekci zakrytí sledovaného objektu. Z pozorování v kapitole 3.6 můžeme vyvodit, že bude pravděpodobně nemožné navrhnout funkční statický práh pro všechny kamery a všechny scény. Proto se jeví jako mnohem lepší řešení navrhnout algoritmus pro dynamický práh, který bude pro každou kameru i každou scénu navrhovat speciální práh. Který se bude odvíjet od průměrné hodnoty chyby a její směrodatné odchylky. Nejprve si tedy spočítáme průměrnou hodnotu chyby g z rovnice (2.3):

$$\bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i \tag{4.1}$$

kde N je počet snímků z nichž počítáme průměrnou hodnotu chyby. V našem případě volíme $N = 10$. Dále spočítáme průměrnou směrodatnou odchylku chyby g z rovnice (2.3):

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(g_i - \bar{g})^2} \tag{4.2}$$

Z těchto dvou hodnot navrhneme 2 dynamické prahy. První práh p_1 bude závislý na velikosti směrodatné odchylky a aritmetickém průměru:

$$p_1 = k \cdot s + \bar{g} \tag{4.3}$$

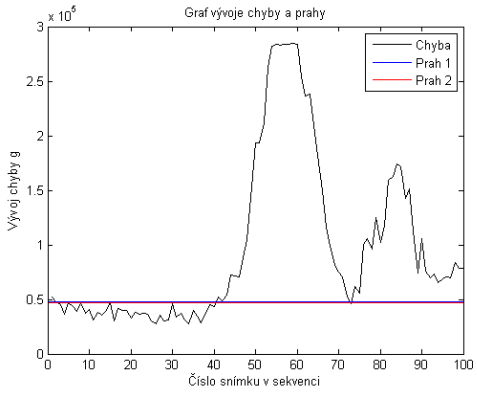
kde k je kladná konstanta, jejíž velikost se pokusíme stanovit na základě experimentů (testujeme k rovno 2; 2,5; 3; 3,5; 4; 4,5 a 5). Druhý práh bude závislý pouze na aritmetickém průměru:

$$p_2 = l \cdot \bar{g} \tag{4.4}$$

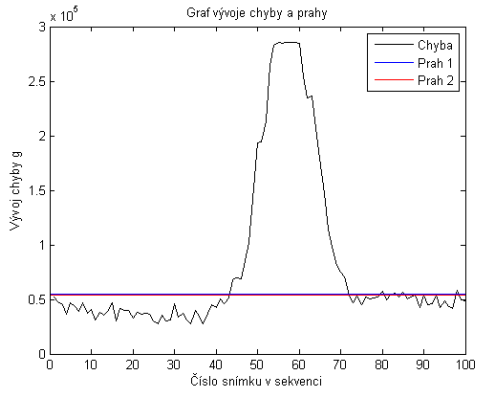
kde l je opět kladná konstanta, kterou stanovíme také na základě experimentů (testujeme l rovno 1,2; 1,25; 1,3; 1,35; 1,4; 1,45 a 1,5). Jednotlivé prahy otestujeme na všech 9 vybraných sekvencích, abychom určili nejvhodnější.

Sekvence 1

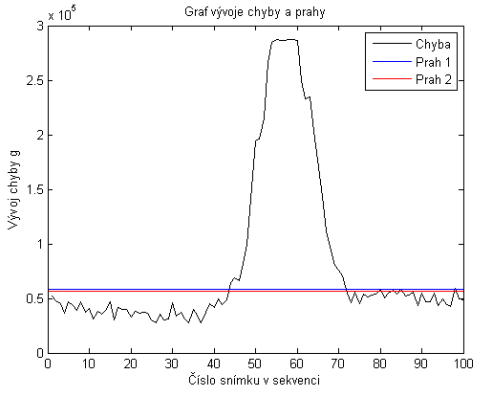
$k = 2;\ l = 1{,}2$



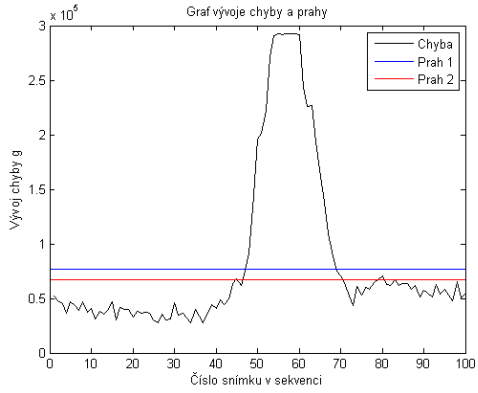
$k = 2{,}5;\ l = 1{,}25$



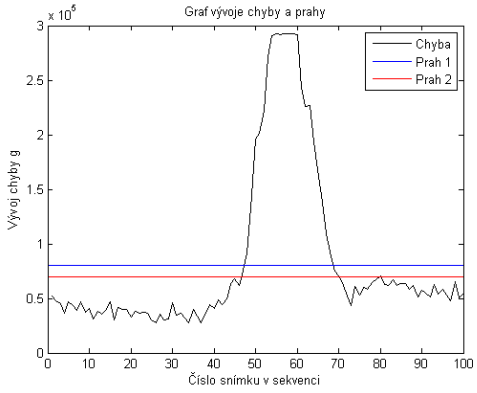
$k = 3;\ l = 1{,}3$



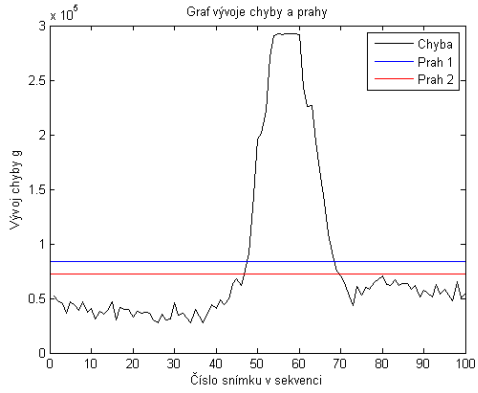
$k = 3{,}5;\ l = 1{,}35$



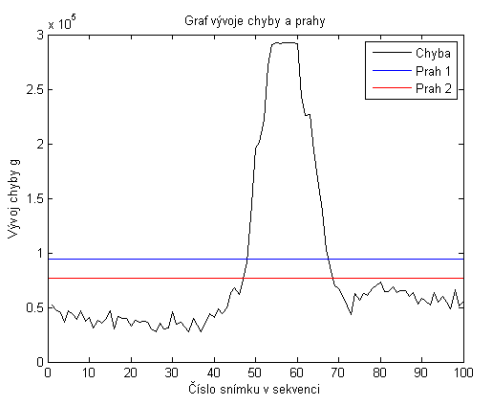
$k = 4;\ l = 1{,}4$



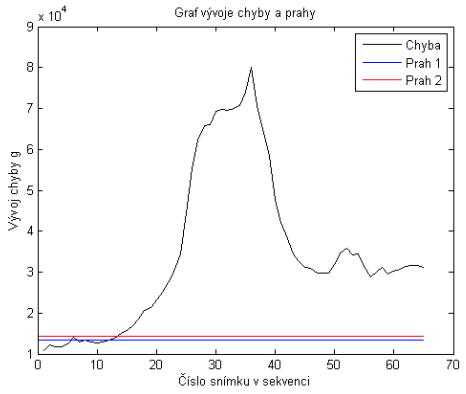
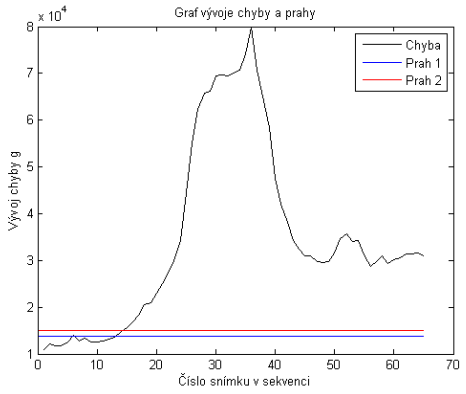
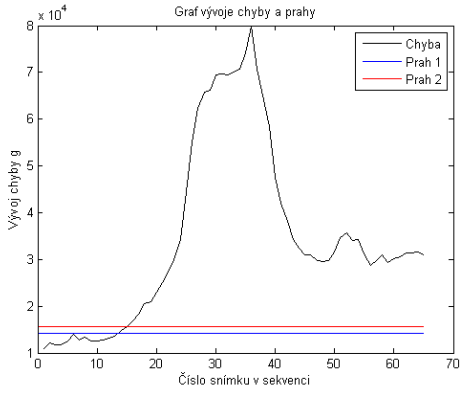
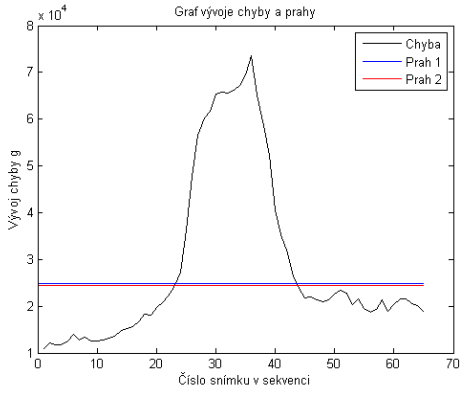
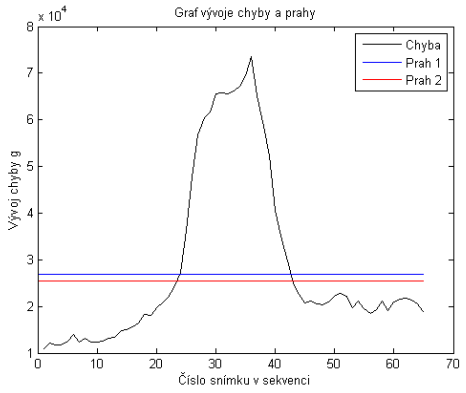
$k = 4{,}5;\ l = 1{,}45$



$k = 5;\ l = 1{,}5$

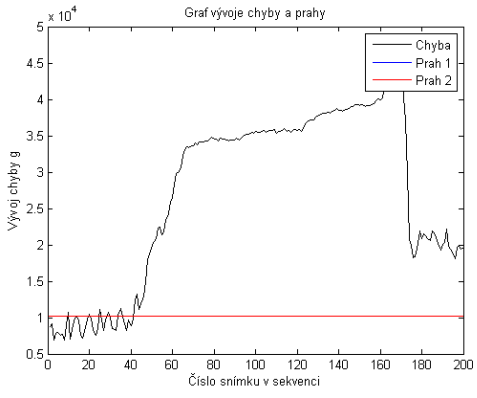
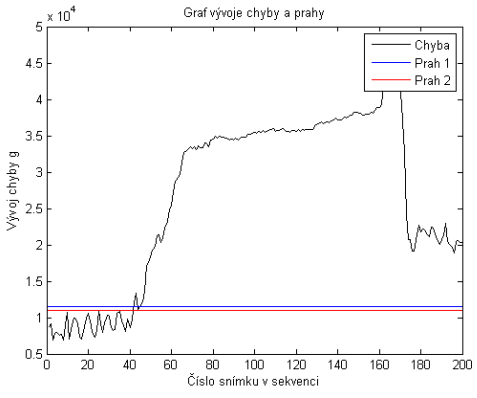
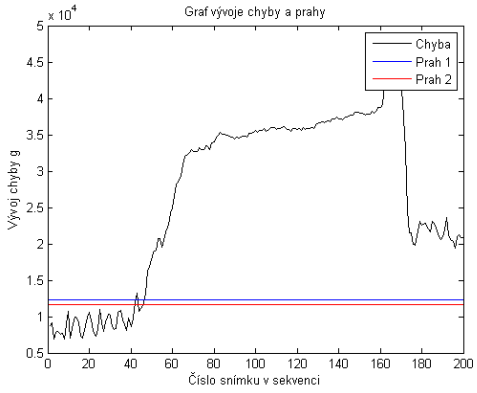
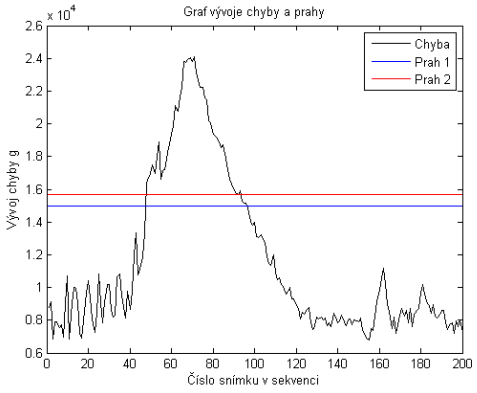
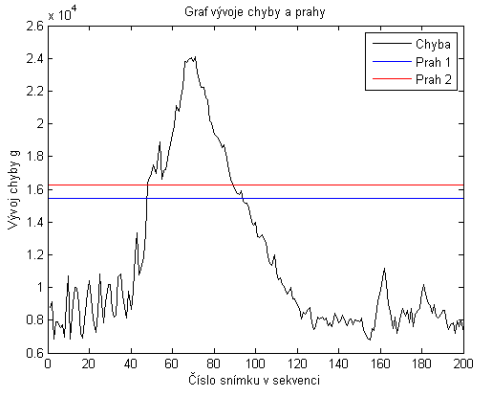
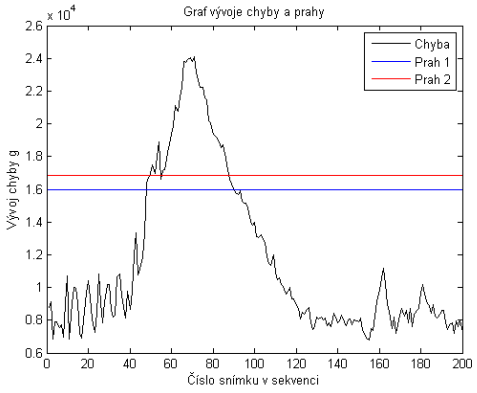
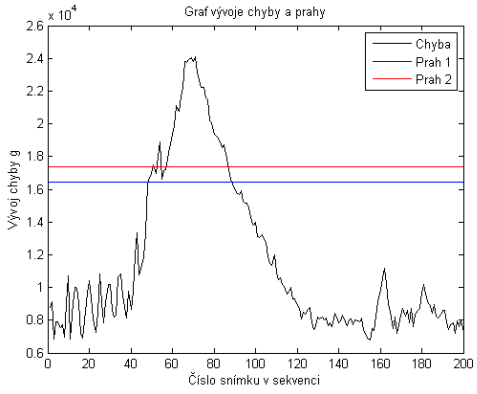


Tab. 4.1 Sekvence 1

Sekvence 2	
$k = 2;\ l = 1{,}2$ 	$k = 2{,}5;\ l = 1{,}25$ 
$k = 3;\ l = 1{,}3$ 	$k = 3{,}5;\ l = 1{,}35$ 
$k = 4;\ l = 1{,}4$ 	

Tab. 4.2 Sekvence 2

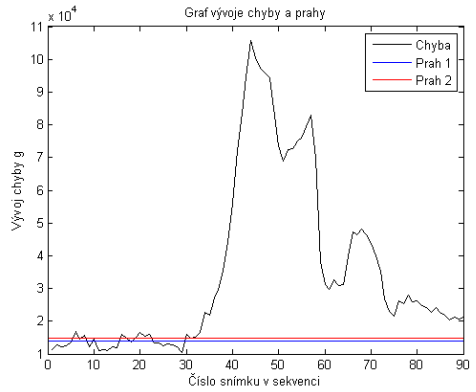
V tabulce 4.2 vidíme výsledky algoritmu pouze pro k = 2; 2,5; 3; 3,5; 4 a l = 1,2; 1,25; 1,3; 1,35; 1,4. Výsledky pro k = 4,5; 5 a l = 1,45; 1,5 nejsou uvedeny, protože algoritmus pro tyto hodnoty ztratil sledovaný cíl, tedy nejsou vhodné pro použití.

Sekvence 3	
$k = 2;\ l = 1{,}2$	$k = 2{,}5;\ l = 1{,}25$
	
$k = 3;\ l = 1{,}3$	$k = 3{,}5;\ l = 1{,}35$
	
$k = 4;\ l = 1{,}4$	$k = 4{,}5;\ l = 1{,}45$
	
$k = 5;\ l = 1{,}5$	
	

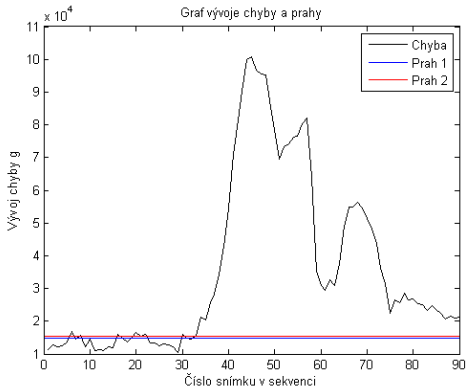
Tab. 4.3 Sekvence 3

Sekvence 4

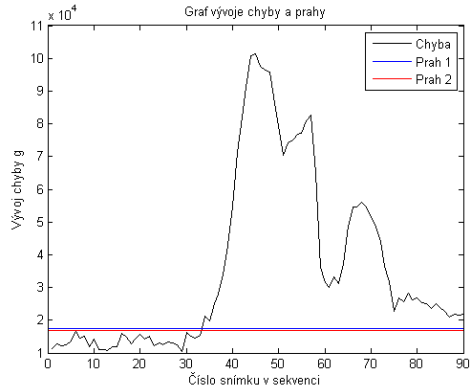
$k = 2;\ l = 1{,}2$



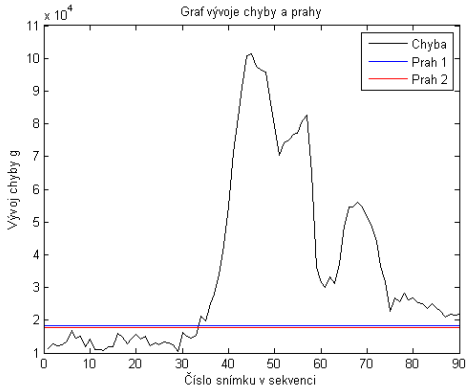
$k = 2{,}5;\ l = 1{,}25$



$k = 3;\ l = 1{,}3$

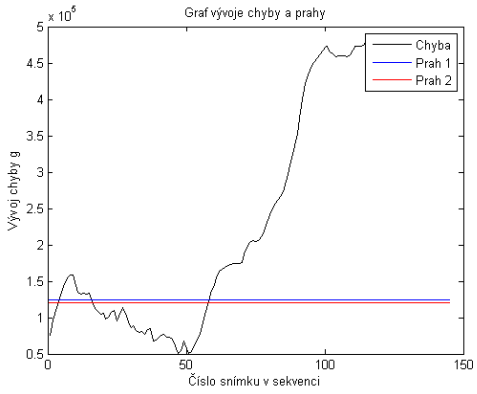
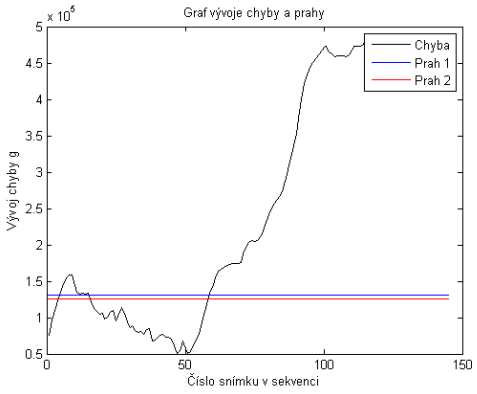
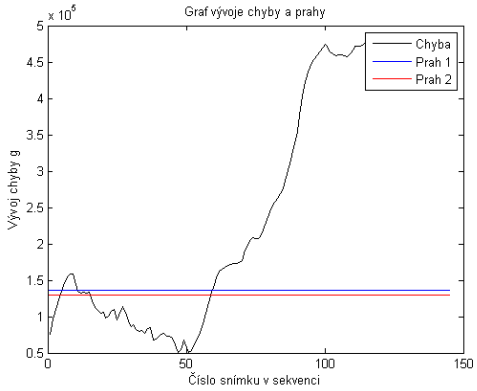
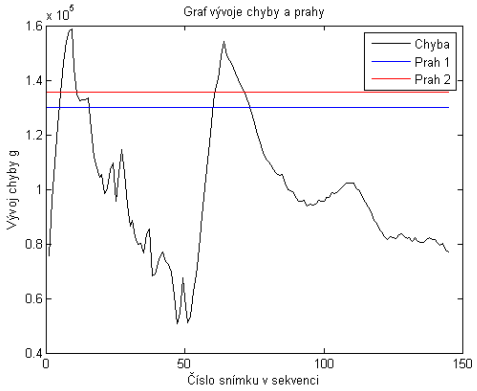
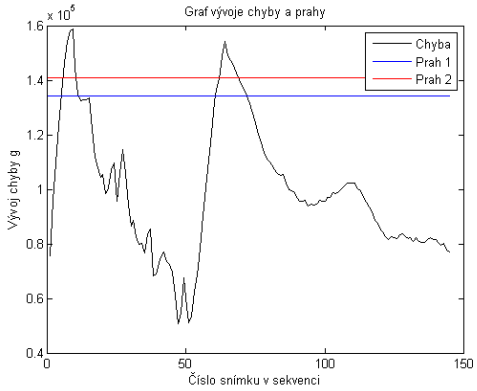
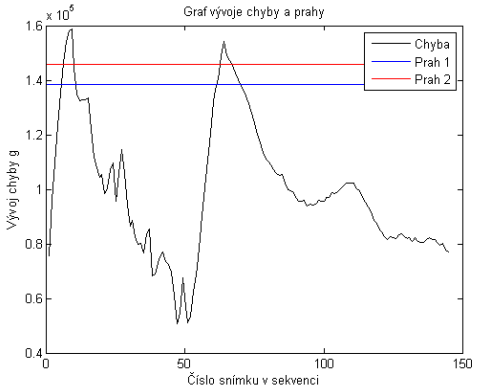
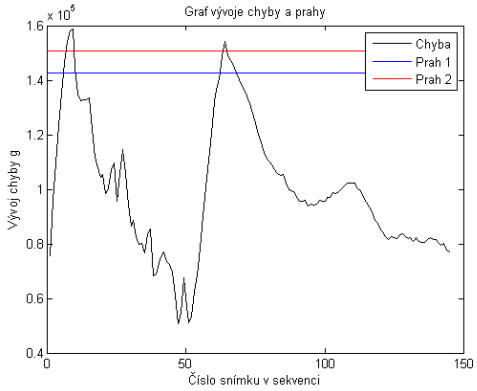


$k = 3{,}5;\ l = 1{,}35$

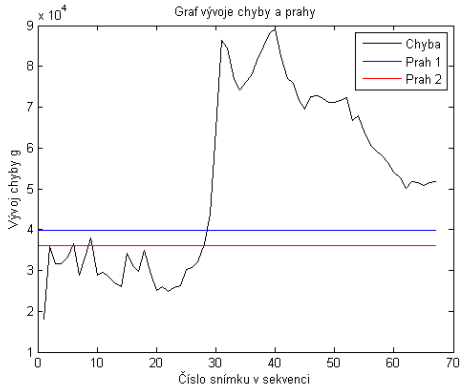


Tab. 4.4 Sekvence 4

V tabulce 4.4 vidíme výsledky algoritmu pouze pro k = 2; 2,5; 3 a l = 1,2; 1,25; 1,3. Výsledky pro k = 3,5; 4; 4,5; 5 a l = 1,35; 1,4; 1,45; 1,5 nejsou uvedeny, protože algoritmus pro tyto hodnoty ztratil sledovaný cíl, tedy nejsou vhodné pro použití.

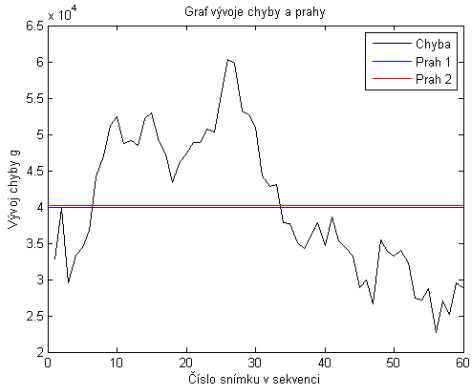
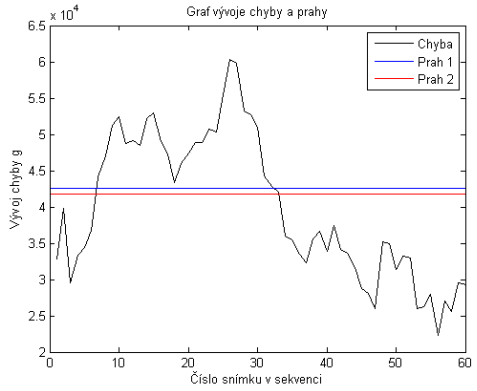
Sekvence 5	
$k = 2;\ l = 1{,}2$	$k = 2{,}5;\ l = 1{,}25$
	
$k = 3;\ l = 1{,}3$	$k = 3{,}5;\ l = 1{,}35$
	
$k = 4;\ l = 1{,}4$	$k = 4{,}5;\ l = 1{,}45$
	
$k = 5;\ l = 1{,}5$	
	

Tab. 4.5 Sekvence 5

Sekvence 6	
$k = 2;\ l = 1{,}2$	
	

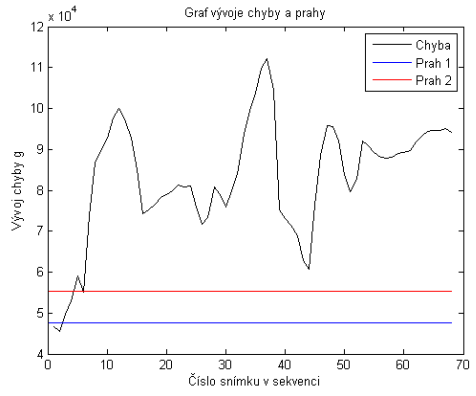
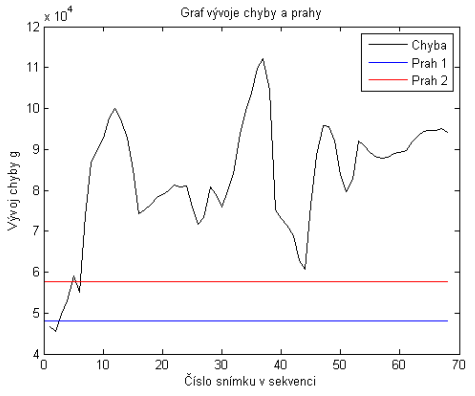
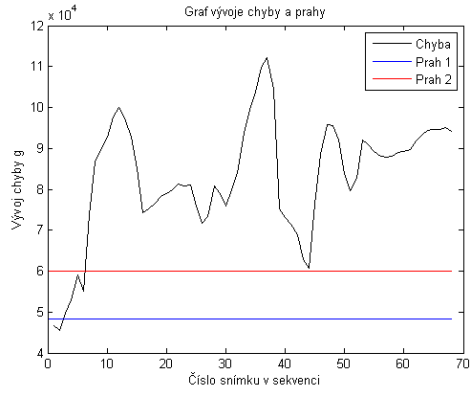
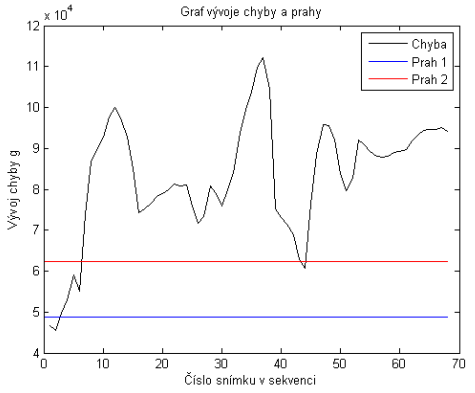
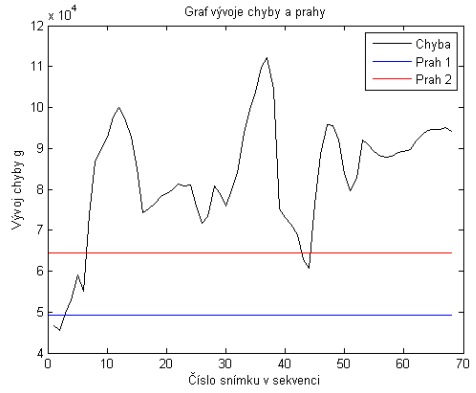
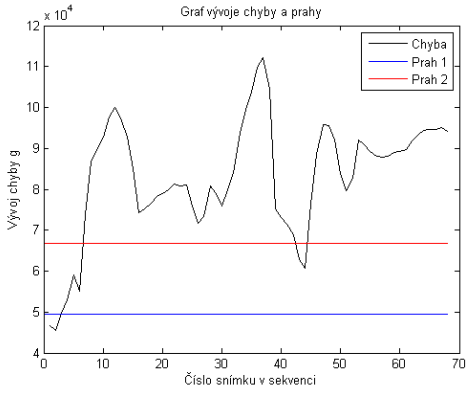
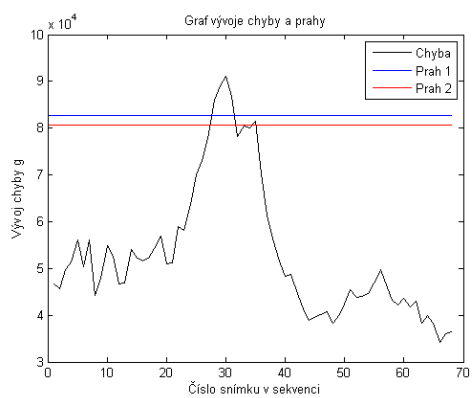
Tab. 4.6 Sekvence 6

V tabulce 4.6 vidíme výsledky algoritmu pouze pro $k = 2$ a $l = 1{,}2$; Výsledky pro ostatní k a l nejsou uvedeny, protože algoritmus pro tyto hodnoty ztratil sledovaný cíl, tedy nejsou vhodné pro použití.

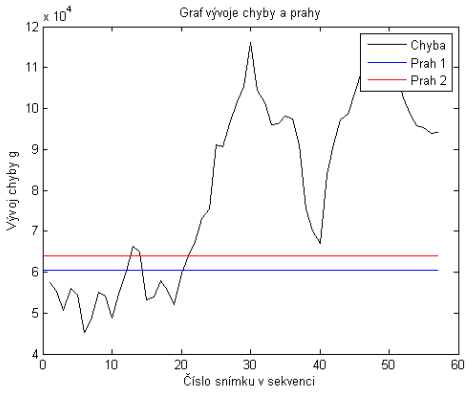
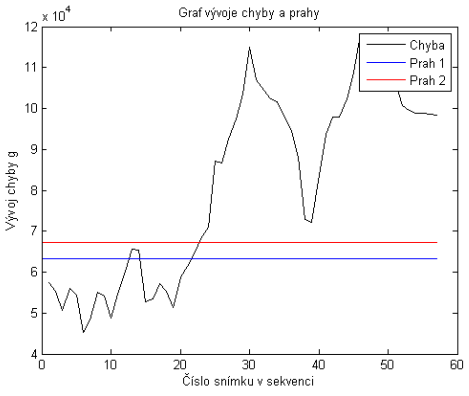
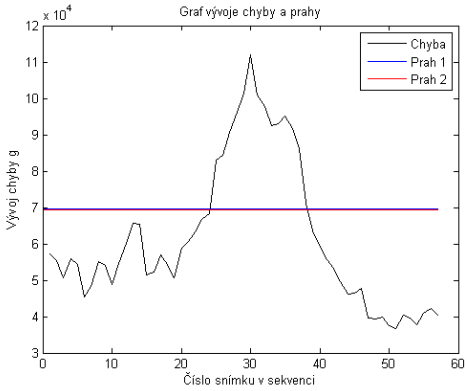
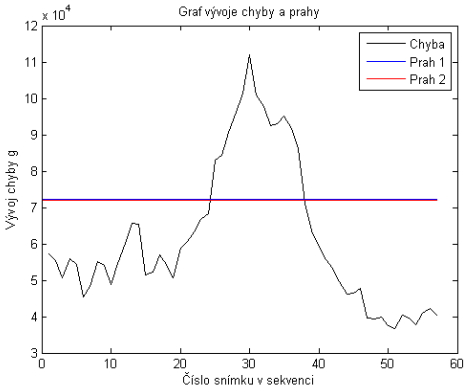
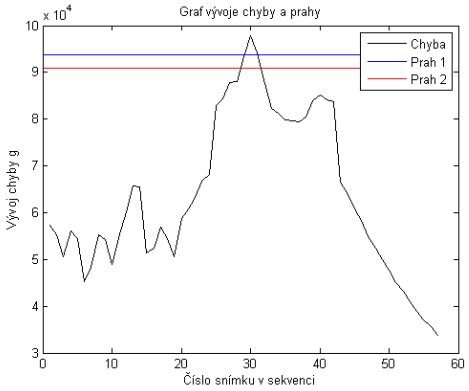
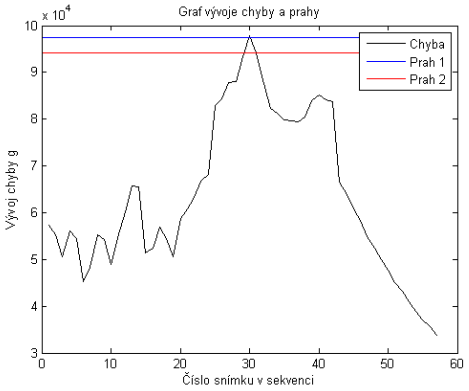
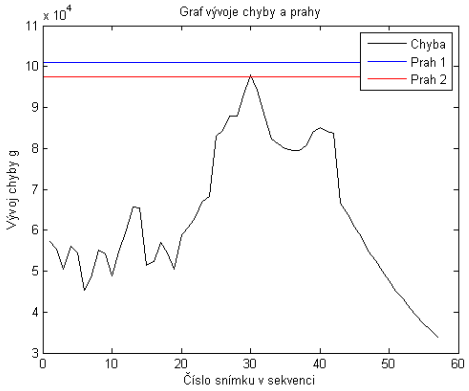
Sekvence 7	
$k = 2;\ l = 1{,}2$	$k = 2{,}5;\ l = 1{,}25$
	

Tab. 4.7 Sekvence 7

V tabulce 4.7 vidíme výsledky algoritmu pouze pro $k = 2;\ 2{,}5$ a $l = 1{,}2;\ 1{,}25$. Výsledky pro ostatní k a l nejsou uvedeny, protože algoritmus pro tyto hodnoty ztratil sledovaný cíl, tedy nejsou vhodné pro použití.

Sekvence 8	
$k = 2;\ l = 1{,}2$ 	$k = 2{,}5;\ l = 1{,}25$ 
$k = 3;\ l = 1{,}3$ 	$k = 3{,}5;\ l = 1{,}35$ 
$k = 4;\ l = 1{,}4$ 	$k = 4{,}5;\ l = 1{,}45$ 
$k = 5;\ l = 1{,}5$ 	

Tab. 4.8 Sekvence 8

Sekvence 9	
$k = 2;\ l = 1{,}2$	$k = 2{,}5;\ l = 1{,}25$
	
$k = 3;\ l = 1{,}3$	$k = 3{,}5;\ l = 1{,}35$
	
$k = 4;\ l = 1{,}4$	$k = 4{,}5;\ l = 1{,}45$
	
$k = 5;\ l = 1{,}5$	
	

Tab. 4.9 Sekvence 9

Z výsledků v tabulkách 4.1 – 4.9 můžeme usoudit, že nelze jednoznačně určit hodnoty k nebo l. Protože když vybereme k nebo l příliš malé, algoritmus pro některé sekvence nenalezne cíl po zákrytu, jako např. sekvence 2, 3, a 4. Naopak pokud zvolíme k nebo l příliš velké, algoritmus označí jako cíle i objekty, které sledovat nechceme, jako např. sekvence 3, 5 a 9 nebo algoritmus přestane fungovat a ztratí cíl úplně, jako např. sekvence 2, 4 a 7, v sekvenci 6, dokonce algoritmus funguje pouze pro hodnoty $k = 2$ nebo $l = 1{,}2$. Z toho důvodu není možné nalézt jedny hodnoty k nebo l, pro které by algoritmus pro všechny sekvence fungoval správně.

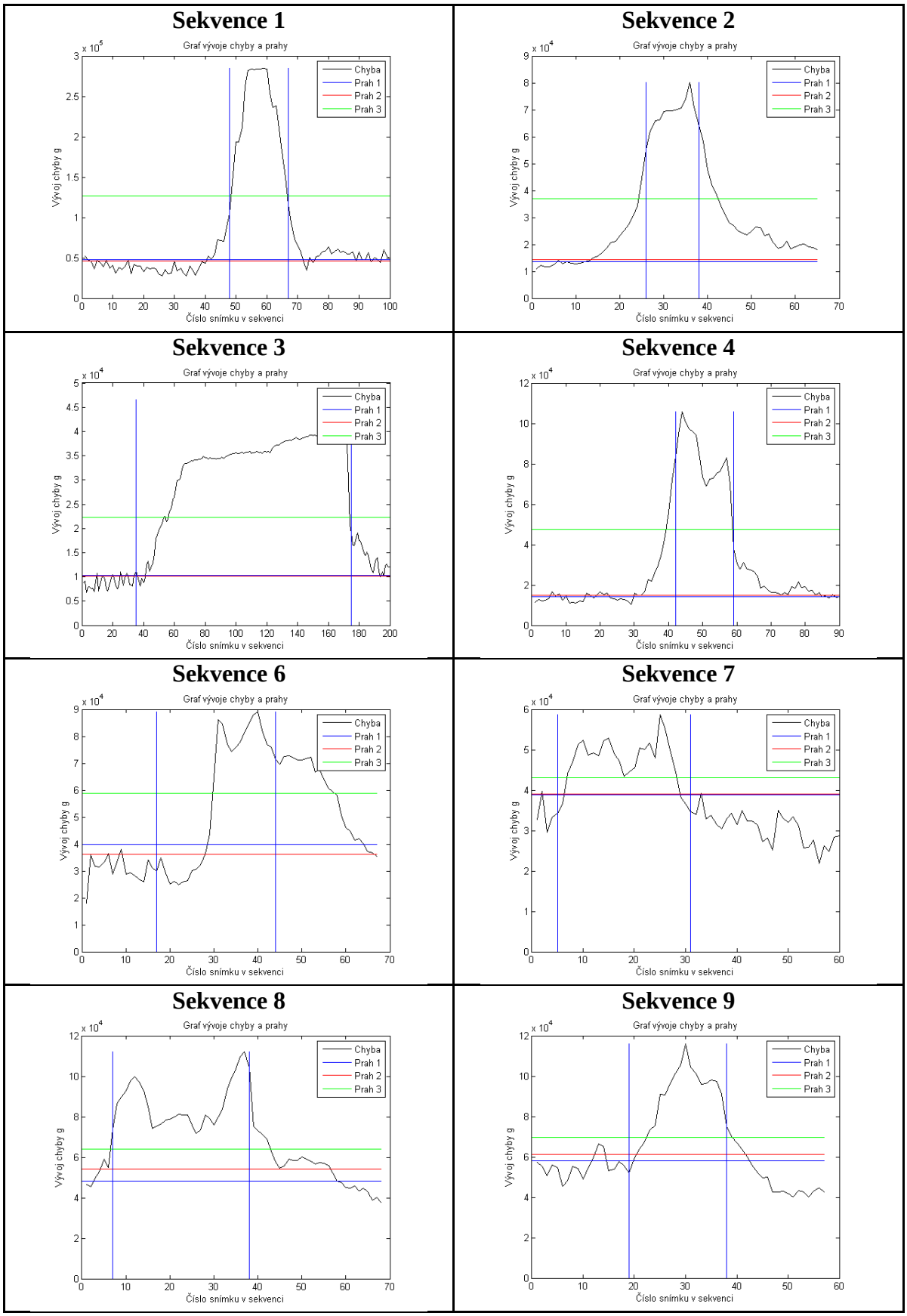
Ovšem při bližším zkoumání si můžeme všimnout, že detekce ztráty cíle na náběžné hraně chybové funkce pracuje správně pro většinu hodnot k nebo l. Tedy že mnohem větší problém je opětovné nalezení cíle, než detekce ztráty. Z grafů je vidět, že práh pro opětovné nalezení cíle na sestupné hraně chybové funkce bude muset být větší než práh pro detekci ztráty. Při výpočtu tohoto prahu, označme si ho práh p_3, využijeme toho, že známe maximální hodnotu chyby, která nastává právě při úplném zákrytu cíle. Práh p_3 by měl ležet někde mezi touto maximální hodnotou chyby g z rovnice (2.3) a aritmetickým průměrem chyby $\bar{g}$, z rovnice (4.1), ale neměl by být příliš velký, abychom neoznačili jako cíle i objekty, které nechceme sledovat. Rovnice pro práh p_3 bude tedy vypadat:

$$p_3 = \frac{\max(g) + \bar{g}}{m} \tag{4.5}$$

kde m jsme experimentálně určili 2,5.

Pokud máme takto určený práh p_3 pro opětovné nalezení cíle, můžeme nastavit hodnoty $k = 2$ nebo $l = 1{,}2$, což jsou hodnoty, které fungují ve všech sekvencích, pro detekci ztráty cíle na náběžné hraně chybové funkce.

Funkčnost algoritmu s prahem p_3 otestujeme na 8 vybraných sekvencích. Na sekvenci 5 není potřeba testovat, protože neobsahuje opětovné nalezení cíle, který řeší nový práh p_3 a funkčnost prahů p_1 a p_2 pro $k = 2$ nebo $l = 1{,}2$ je vidět v tabulce 4.5. Do grafů dále zakreslíme svislými čarami zákryt a opětovné objevení cíle.



Tab. 4.10 Detekce zákrytu s prahem p_3

Kapitola 5

Závěr

Cílem mé práce bylo prozkoumat, navrhnout a ověřit algoritmus pro detekci ztráty cíle, při zákrytu na základě chyby, kterou vrací chybová funkce algoritmu SAD. Nejprve byla představa, že navrhnu statický práh neboli konkrétní hodnotu, která bude určovat, zda ještě sledovat cíl nebo sledování přerušit. Ovšem toto řešení se brzy ukázalo jako nevhodné, protože hodnota chyby je velmi odlišná pro sekvence s různými kamerami a různými scénami a není možné navrhnout práh, který by pracoval správně pro všechny sekvence.

Proto jsem navrhl řešení na základě dynamického prahu, který by zohledňoval právě tuto odlišnost chyby pro různé sekvence. Navrhl jsem dva algoritmy určení prahu, jeden na základě aritmetického průměru chyby a druhý na základě směrodatné odchylky a aritmetického průměru chyby a vyzkoušel na video sekvencích zachycujících sledování různých objektů. Ukázalo se jako obtížnější opětovné nalezení cíle po zákrytu, než detekce ztráty cíle při zákrytu. Proto ani dynamický práh nefungoval.

Z toho důvodu jsem navrhl algoritmus stanovující různou úroveň chybové funkce pro detekci opětovného objevení cíle. To jsem udělal za pomoci znalosti aritmetického průměru chyby a maximální hodnoty chyby, právě během zákrytu. Po vyzkoušení kombinace prahů pro detekci zákrytu a prahu pro opětovné nalezení cíle byly výsledky dostatečně uspokojivé a takto navržený algoritmus splnil cíl mé práce.

Literatura

[1] Q. Zhu, K-T. Cheng, H. Zhang. **SSD tracking using dynamic template and log-polar transformation.** In International Conference on Multimedia and Expo 1. 2004. 723-726 s.

[2] S. Baker, I. Matthews. **Lucas-Kanade 20 Years On: A Unifying Framework.** International Journal of Computer Vision 56(3). 2004. 221-255 s.

[3] D. Comaniciu, V. Ramesh, P. Meer. **Kernel-based object tracking.** IEEE Transactions on Pattern Analysis and Machine Intelligence 25(5). 2003. 564-575 s.

[4] I. Matthews, T. Ishikawa, S. Baker. **The Template Update problém.** IEEE Transactions on Pattern Analysis and Machine Intelligence 26(6). 2004. 810-815 s.

[5] D.G.Lowe. **Object recognition from local scale-invariant features.** International Conference on Compuber Vision 2. 1999. 1150-1157 s.

[6] H.Grabner, J.Sochman, H.Bischof, J.Matas. **Training Sequential On-line Boosting Classifier for Visual tracking.** 19th International Conference on Pattern Recognition. 2008. 1-4 s.

[7] J. Šindler, V. Smutný. **Camera Noise Measurement**. Center for Machine Perception, K13133 FEE Czech Technical University. CTU-CMP-2012-09 (1213-2365). 2012. 5 s.

[8] Sony. **Color Block Cameras FCB-EX Series**. Sony Electronics Inc. Technický list. 2005

[9] Sony. **Color Block Cameras FCB-IX Series**. Sony Electronics Inc. Technický list. 2005

[10] Canon. **XM2-Specifikace produktu**. 2012

Obsah CD

./Bakalářská_práce	Text bakalářské práce ve formátu PDF
./Testovací_sekvence	Vytvořená testovací sekvence s příslušnými M-fily pro vytvoření sekvence, lokalizaci objektu a generaci grafů
./Sekvence	9 vybraných video sekvencí s příslušnými M-fily pro lokalizaci objektu, algoritmy pro návrh prahů a generaci grafů
./Grafy	Veškeré grafy použité v bakalářské práci