VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

SLEDOVÁNÍ VYBRANÉHO OBJEKTU V DYNAMICKÉM OBRAZE

OBJECT TRACKING IN VIDEOFEED

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — BC. MAREK KLVAŇA
AUTHOR

VEDOUCÍ PRÁCE — ING. JIŘÍ KREJSA, PH.D.
SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Marek Klvaňa

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Sledování vybraného objektu v dynamickém obraze

v anglickém jazyce:

Object tracking in videofeed

Stručná charakteristika problematiky úkolu:

Navrhněte a implementujte metodu pro sledování vybraného objektu v dynamickém obraze. Implementace by měla být dostatečně rychlá, aby na běžném PC zvládla při rozlišení 640x480 rychlostí alespoň 5 fps.

Cíle diplomové práce:

1. Vyberte některou z perspektivních metod sledování objektu v dynamickém obraze
2. Metodu implementujte
3. Implementaci otestujte a kriticky vyhodnoťte.

Seznam odborné literatury:

Vedoucí diplomové práce: Ing. Jiří Krejsa, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2010/2011.

V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc. Děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá popisem a implementací algoritmů pro obecné sledování objektu v dynamickém obraze. Jsou zde představeny sledovací algoritmy Mean shift a Continuously adaptive mean shift, které reprezentují skupinu jádrového sledováni. K sestavení modelu sledovaného objektu je použit trojrozměrný barevný histogram, jehož konstrukce je rovněž popsána a zahrnuta do konceptu této práce. Výkony uvedených sledovacích algoritmů jsou porovnány na zkušebních snímkových sekvencích a podrobně vyhodnoceny.

ABSTRACT

The aim of this thesis is a description and implementation of algorithms of the tracked objects in the video feed. This thesis introduces Mean shift and Continuously adaptive mean shift algorithms which represent category based on kernel tracking. For construction of a model is used a three-dimensional color histogram whose construction is described in this thesis as well. The achievements of described algorithms are compared in the testing images sequences and evaluated in details.

KLÍČOVÁ SLOVA

algoritmus, histogram, mean shift, CAMShift, zpětná projekce, adaptace, porovnání

KEYWORDS

algorithm, histogram, mean shift, CAMShift, back projection, adaptation, comparison

Obsah:

1 ÚVOD

S neutichajícím technologickým rozvojem se čím dál více rozléhá pole působnosti výpočetní techniky. V oboru informačních technologií vznikají nové specifičtější obory, které umožňují využití výpočetní techniky takřka ve všech odvětvích. S využíváním počítačů a díky jejich potenciálu zde nastala snaha o to, aby se počítač co nejvíce vyrovnal člověku a v mnoha směrech ho překonal. Tuto myšlenku naplňuje, ať už přímo nebo nepřímo nespočet vědních oborů. Aby se počítač přiblížil chování lidí a tak vzrostla již jeho rozsáhlá možnost uplatnění, je nutné, aby byl schopen simulovat lidské chování. K dosažení tohoto cíle musí být splněno nespočetné množství požadavků.

Jedním z velice důležitých a nejobtížnějších požadavků je schopnost počítače „myslet“, tedy odvětví umělé inteligence, ať už v podobě rozhodovacích algoritmů, známých například ze zábavního průmyslu nebo robustních učících se algoritmů v podobě neuronových sítí či použitím hybridních přístupů. Jako příklad se zde dá uvést kombinace neuronových sítí, fuzzy logiky a genetických algoritmů.

Odlišným požadavkem je umožnit počítači vidět. Odvětví, které se zabývá tímto tématem, se nazývá Počítačové vidění a právě do této problematiky spadá téma této práce. Počítačové vidění (cumputer vision) je relativně nové odvětví informačních technologií, které se zabývá získáváním informací pomocí zpracovávání obrazu (image processing), tedy analyzování předloženého obrazu podle specifických parametrů. Takové analýzy se hojně používají především v medicíně za účelem stanovení diagnózy pacienta. Používají se zejména při rentgenových vyšetřeních, ultrazvukových vyšetřeních nebo vyšetřeních pomocí výpočtové tomografie která je známa jako „CT“. Další široké uplatnění počítačového vidění je například ve strojírenském průmyslu, kde se často pomocí kamerových zařízení kontroluje jakost výrobku. Příkladem z praxe je situace, kdy pomocí aplikace algoritmu detekce hran, jsou kontrolovány tolerance rozměrů výrobku.

Prozatím bylo zmíněno využití počítačového vidění pro zpracování jednoho obrazu, ale také je možné využití pro zpracování videa. Video je v podstatě posloupnost rychle po sobě jdoucích jednotlivých snímků, které na sebe navazují. Tyto snímky je možné zpětně získat, pokud video rozdělíme v určitých časových intervalech. Tímto způsobem je možné získat sekvenci po sobě jdoucích snímků, které mohou být jednotlivě podrobeny potřebné analýze.

V důsledku zpracování videa se v rámci počítačového vidění otevírají nové možnosti. Využití je obrovské, stejně jako v rámci zpracování obrazu. Jednou z využívaných funkcí je rozpoznávání obličeje, kterou jsou dnes vybaveny běžně dostupné digitální fotoaparáty a webové kamery. Tato funkce se obecně nazývá detekce objektu a je často doplňována o funkci rozpoznávání objektu. Detekce objektu může pracovat například na principu rozpoznávání hran nebo barvy, s jehož pomocí je možné odlišit jednotlivé objekty, například od prostředí. Mezi obdobnou metodiku patří algoritmy rozpoznávání objektu, které jsou schopny odlišit specifické objekty od ostatních objektů na snímku. Pokud je možné objekty odhalovat a rozpoznávat, potom je možné v rámci jednotlivých snímků objekty lokalizovat. Metodika zabývající se lokalizací polohy objektů v sekvenci po sobě následujících snímků, se nazývá Sledování objektu (object tracking). A právě rozborem a implementací algoritmů sledování objektu se zabývá tato práce.

Od algoritmů sledování objektů se očekává, že budou schopny označovat polohu sledovaného objektu v sekvenci po sobě následujících snímků tak, že sledovaný objekt nebude ztracen. Tohoto požadavku může být v některých případech velice obtížné dosáhnout. Zejména v případech, kdy sledovaný objekt splývá s jednotvárným prostředím, je cloněn cizími předměty nebo se pohybuje v blízkosti identických objektů. Z těchto důvodů patří vytvoření modelu a implementace robustního, přesného a výpočetně nenáročného sledovacího algoritmu mezi jednu z nejobtížnějších výzev programátorů působících v oblasti počítačového vidění.

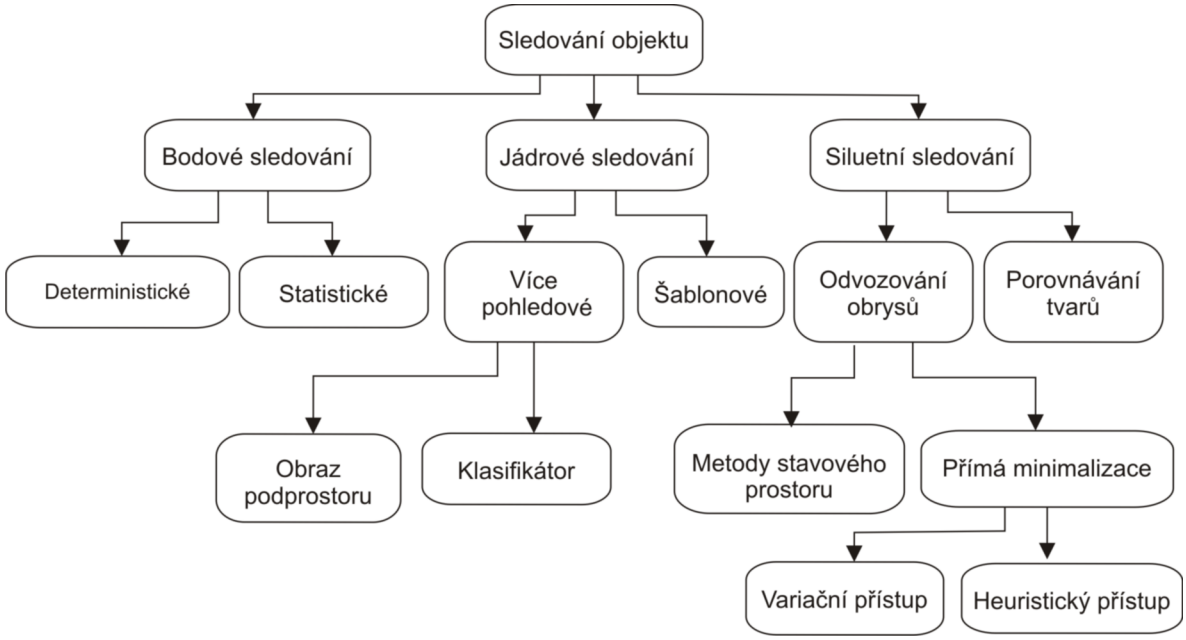
Uplatnění robustních sledovacích algoritmů v praktickém využití je obrovské. Zejména se jedná o využití v medicíně, kde může být sledování objektů zastoupeno v podobě navádění automatizovaných přístrojů na cílový objekt, stejně tak jako v robotice, dopravním průmyslu, letectví a strojírenství. Další uplatnění může být hojně zastoupeno v bezpečnosti a zábavním průmyslu, kde může zastupovat funkci snímače pro uživatelská rozhraní ovládaná gestikulací, jak tváře nebo celého těla. Příkladem může být zařízení Kinect vyvinuté společností Microsoft.

Požadavkem na tuto práci bylo najít a implementovat obecný sledovací algoritmus, který umožňuje sledování objektu na snímcích s rozlišením 640 x 480 pixelů s minimální snímkovou frekvencí 5 snímků za vteřinu. Nicméně algoritmy představené v této práci jsou dostatečně rychlé a tedy nebrání využití pro sledování v reálném čase. Zároveň vykazují dostatečnou přesnost pro sledování komplexnějších objektů, jako mohou být například lidské tváře.

2 PŘEHLED METOD PRO SLEDOVÁNÍ OBJEKTU

Aby bylo možné splnit předpoklady diplomové práce v podobě implementace dostatečně rychlého algoritmu pro obecné sledování, je nejprve nutné zjistit jaké jsou v tomto směru možnosti. Z tohoto důvodu byla vypracována rešerše, která obsahuje stručný pohled na sledovací techniky.

Technik pro sledování objektu v dynamickém obraze je celá řada. Podle principu se dají rozdělit do třech základních kategorií. Jedná se o kategorii Bodového sledování (Point tracking), Jádrového sledování (Kernel Tracking) a Siluetové sledování (Silhouette tracking). Tyto kořenové kategorie se dále větví dle specifičtějšího přístupu zpracování. Hierarchie rozdělení sledování objektu je znázorněna na obrázku číslo 1. Obsah celé této kapitoly vychází z [6], kde jsou jednotlivé techniky popsány podrobněji.



Obr. 1 Rozdělení metod sledování objektu. [6]

Bodové sledování

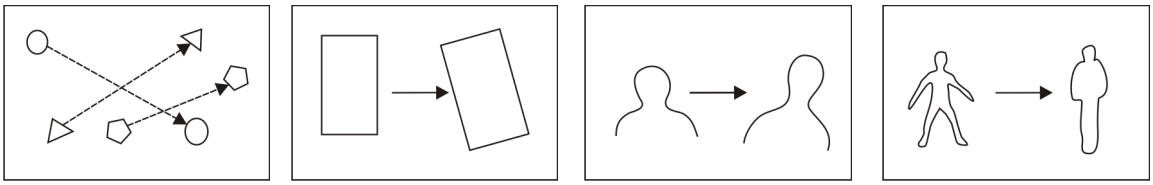
V případě Bodového sledování jsou objekty na jednotlivých snímcích reprezentovány pomocí bodů. Sdružování jednotlivých bodů vychází z předchozího stavu objektu. Stav objektu může reprezentovat pohyb objektu a polohu ve které se na snímku nachází. Tato technika vyžaduje externí mechanismus, který na každém snímku provede detekci objektu. Ukázka navzájem odpovídajících si objektů, je zobrazena na obrázku 2(a). [6]

Jádrové sledování

U jádrového sledování je jádro chápáno jako šablona, která zastupuje tvar sledovaného objektu. Šablona bývá nejčastěji obdélníkového nebo eliptického tvaru. K této šabloně je vztažen odpovídající model sledovaného objektu, který může být například v podobě barevného histogramu. Sledování objektů je prováděno pomocí výpočtu pohybu jádra, v sekvenci po sobě jdoucích snímků (obr. 2(b)). [6]

Siluetové sledování

Sledování siluet je založeno na odhadu oblasti sledovaného objektu v sekvenci po sobě jdoucích snímků. Tyto metody pro sledování pracují s informací, která je obsažena uvnitř oblasti objektu. Tyto informace bývají nejčastěji v podobě mapy hran. Sledování modelu objektu neboli jeho obrysu, je prováděno porovnáváním tvaru nebo odvozováním (evolucí) obrysů objektu, což je znázorněno na obrázcích 2(c) a 2(d). [6]



(a) (b) (c) (d)

Obr. 2 Rozdílné sledovací techniky. [6] (a) více-bodová korespondence, (b) parametrická transformace, (c, d) příklady odvozování obrysů.

2.1 Bodové sledování

Bodové sledování lze chápat jako korespondenci detekovaných objektů, které jsou vyjádřeny pomocí bodů v posloupnosti po sobě následujících snímků. Obecně při sledování objektu nastávají komplikace v případě okluze. Tento problém setrvává i v případě bodového sledování, kdy bodová korespondence může být složitým problémem. Rovněž mohou nastat komplikace v případě vstupů a výstupů objektů v průběhu sledování. Metody bodové korespondence mohou být rozděleny do dvou kategorií. První kategorie se označuje jako deterministické metody pro korespondenci. Druhou kategorií jsou statistické metody pro korespondenci.[6]

2.1.1 Deterministické metody pro korespondenci

Deterministické metody pro korespondenci vymezují cenu sdružující každý objekt ve snímku v čase t – 1 k jednotlivému objektu ve snímku t, užitím skupiny omezujících pohybových podmínek. Minimalizace ceny korespondence je formulována jako kombinatoricky optimalizační problém. Korespondenční hodnota je obvykle definována užitím kombinace z následujících omezujících podmínek.[6]

1) Přiblížení (proximity) - v případě omezující podmínky přiblížení se předpokládá taková poloha objektu, která by neměla být významně změněna při přechodu z jednoho snímku sekvence na druhý (obr. 3(a)).

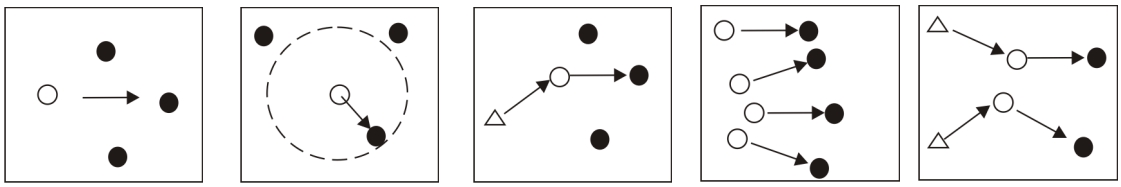
2) Maximální rychlost (maximum velocity) – omezení maximální rychlosti udává horní hranici rychlosti sledovaného objektu a omezuje eventuální korespondence ke kruhové oblasti kolem objektu (obr. 3(b)).

3) Malá změna rychlosti (small velocity change) – u tohoto omezení se předpokládá směr a rychlost objektu v případech, kdy nedochází k prudkým změnám (obr. 3(c)).

4) Společný pohyb (common motion) – dochází k omezování rychlostí objektů, které se nacházejí v blízkém sousedství v závislosti na jejich podobnosti. Tímto omezením jsou ovlivňovány objekty, které jsou reprezentovány větším počtem bodů (obr. 3(d)).

5) Tuhost (rigidity) – tato podmínka vyplývá z předpokladu, že objekty, které se nacházejí v trojrozměrném prostoru jsou tuhé. Z tohoto důvodu vzdálenost mezi dvěma jakýmikoliv body na tomto objektu zůstane beze změny (obr. 3(e)).

6) Blízká jednotnost (proximal uniformity) – tato omezující podmínka je kombinací dvou omezení. Jedná se o omezení přiblíženosti a také malé omezení rychlosti objektu.



(a) (b) (c) (d) (e)

Obr. 3 Pohybové omezující podmínky. [6] (a) přiblížení, (b) maximální rychlost, (c) malá změna rychlosti, (d) společný pohyb, (e) podmínky tuhosti

2.1.2 Statistické metody pro korespondenci

V případě statistických metod pro korespondenci se jedná o využívání stavového prostoru k vyjadřování vlastností objektu. Mezi tyto vlastnosti objektu může patřit poloha, rychlost pohybu a zrychlení objektu. V případě měření se obvykle vychází z polohy sledovaného objektu na jednotlivém snímku sekvence. Poloha je získána užitím detekčního mechanismu. Mezi metody, které se používají pro odhad stavu jednotlivého objektu patří následující. [6]

1) Kalmanův filtr

2) Částicový filtr (Particle filter)

Pro více objektovou datovou asociaci a odhad stavu se používají metody:

1) Společně pravděpodobnostní asociační datový filtr (Joint probability data association filter)

2) Vícenásobné hypotézové sledování (Multiple hypotesis tracking)

2.2 Jádrové sledování

Algoritmy spadající do jádrového sledování pracují na principu výpočtu pohybu sledovaného objektu, který se provádí na každém snímku sekvence. Sledovaný objekt je vyjádřen pomocí oblasti, která reprezentuje tvar a vzhled cíle sledování. Tato oblast bývá nejčastěji obdélníkového nebo eliptického tvaru. Algoritmy spadající do této kategorie se od sebe liší způsobem, jakým je vzhled objektu zastoupen. Dále se liší počtem sledovaných objektů a podle metodiky kterou je pohyb sledovaného objektu odhadován. Tyto metody mohou být rozděleny do dvou podkategorií. Jedná se o kategorii Šablon a modelů vzhledu založených na hustotě (Template and Density-Based Appearance Models) a kategorii Více pohledového vzhledu modelu (Multi-view appearance model). [6]

2.2.1 Sledování pomocí šablony a modelu vzhledu

Technika pro sledování objektu této podkategorie se používá z důvodů, že je relativně jednoduchá a patří mezi výpočetně nenáročné. Tato podkategorie se může dále rozdělit do dalších dvou skupin a to podle počtu sledovaných objektů a podle způsobu zda se objekty sledují individuálně

anebo společně. [6]

2.2.1.1 Sledování jednotlivých objektů

Nejběžnější technikou sledování objektu v tomto odvětví jádrového sledování, je technika porovnávání šablon. Jedná se o metodu hrubé síly, kterou se hledá oblast obrazu, která je podobná šabloně objektu, jenž je definovaná v předchozím snímku. Poloha šablony v současném snímku je vypočítána pomoci míry podobnosti. Obvykle je pro vytváření šablon použita intenzita obrazu nebo jeho barevné vlastnosti. Omezením pro techniku založenou na porovnávání šablon, je její velká výpočetní náročnost v důsledku použití přístupu hrubé síly pro hledání objektu. Za účelem snížení výpočetní náročnosti se nejčastěji používá omezení prohledávacího prostoru na okolí bodu, ve kterém byl objekt nalezen na předchozím snímku. Nemusí se však striktně jednat o reprezentaci objektu pomocí šablon. Pro reprezentaci sledovaného objektu může být použit barevný histogram nebo směs modelů, které mohou být vypočítány z barevných hodnot jednotlivých pixelů uvnitř obdélníkové nebo elipsoidní oblasti, ohraničující sledovaný objekt. [6] Do této kategorie také spadá sledovací algoritmus nazývaný Mean shift, který bude detailně popsán v další části této práce.

Metoda sledování vzhledu (Appearance Tracking) popsaná Allanem D. Jepsonem, která je zmíněná v [6], sleduje objekty na základě tří složek. Tato skupina složek je sestavena ze složky stálých vzhledových vlastností, složky přechodných vlastností a složky šumu. Stálá složka rozlišuje nejreálnější vzhled pro odhad pohybu. Jedná se o takové oblasti objektu, které se rychle nemění. Oproti tomu u složky přechodných vlastností se rozlišují a vyhodnocují takové pixely na obrazu, které se rychle mění v čase. Poslední z trojice složek - složka šumu zpracovává hodnoty, které jsou odlehlé od vzhledu objektu. Tyto hodnoty pravděpodobně vznikly v důsledku rušení. V metodě se využívá online verze EM algoritmu (expectation-maximization algorithm), který slouží k určování výše popsaných složek. Tvar sledovaného objektu je zastoupen eliptickým tvarem. Výpočet pohybu sledovaného objektu se provádí v závislosti na deformaci tohoto eliptického tvaru, při přechodu mezi jednotlivými snímky. [6]

Odlišným způsobem sledování objektu, který je reprezentován obdélníkovým nebo eliptickým tvarem, je posun užitím metody optického toku. Tyto metody jsou použity pro generování hustoty oblastí, pomocí výpočtu vektoru toku každého pixelu, v rámci konstantního omezení jasu. Takový výpočet je prováděn v sousedství pixelu a to buď algebraicky nebo geometricky. Jinou metodou je KLT sledování (Kanade-Lucas-Tomasi Feature Tracker), které opakovaně počítá pohyb oblasti cíle, která je vystředěna k určitému bodu. [6]

2.2.1.2 Sledování více objektů

Při modelování jednotlivých objektů není v důsledku sledování zakomponováno vzájemné působení mezi dvěma nebo více objekty. Také se zde nebere v úvahu vzájemné působení mezi objekty a pozadími objektů. Příkladem je, kdy vzájemné působení mezi objekty může vést k částečnému nebo úplnému clonění ostatních objektů. [6] Metoda představena v [6], řeší tento problém pomocí vrstvení obrazu. Toto řešení spočívá v užití jedné vrstvy, která je určena výhradně pro pozadí a dále užití jednotlivých vrstev pro každý objekt. Jednotlivé vrstvy se skládají z předchozího tvaru, dále z pohybového modelu a popisu daného vzhledu hladiny.

2.2.2 Vícepohledové vzhledy modelu

Ve výše popsaných metodách jsou použity takové modely objektu, které jsou většinou vytvářeny online. Jednalo se zde o histogramy, šablony a podobně. Sestavování takových modelů většinou zastupuje vzhled daného objektu, který vyplývá z nedávného pozorování. Je zřejmé, že komplexnější objekty mohou mít různorodý vzhled v závislosti na úhlu, z kterého je daný objekt

pozorován. Například se může jednat o různorodé tvary objektu nebo odlišné barevné odstíny objektu. V případě, kdy se objekt v průběhu sledování intenzivně mění, je vysoce pravděpodobné, že původně sestavený model objektu, který například zahrnoval pouze jednu stranu, nemusí být nadále validní. V takovém případě, může snadno dojít ke selhání sledovacího algoritmu. Takovému problému je možné předejít v případě, kdy jsou rozdílné náhledy na objekt naučeny offline. [6]

2.3 Siluetové sledování

Při sledování objektu se nemusí vždy jednat o objekty, které mohou být reprezentovány jednoduchým tvarem, nýbrž tvar objektů může být daleko komplexnější. Například tvar lidské ruky není jednoduché popsat primitivním geometrickým tvarem. Sledováním a popisem složitějších geometrických tvarů se vyznačují metody, které jsou založeny na sledování siluet. Cílem těchto metod je nalezení oblasti daného objektu v každém snímku sekvence za pomoci modelu, který byl vygenerován v rámci sledování na předchozím snímku. Takový vygenerovaný model bývá nejčastěji ve formě barevného histogramu, hran objektu nebo v podobě obrysů sledovaného objektu. Siluetové sledování může být rozděleno do dvou kategorií. První kategorie se nazývá Porovnávání tvaru a druhou kategorií je Sledování obrysu. V případě metod porovnávání tvaru se v každém snímku hledá silueta cílového objektu. Oproti tomu při technikách sledování obrysu dochází k vyvíjení počáteční kontury objektu za účelem nalezení nové polohy na každém snímku sekvence. Za tímto účelem je využíváno modelů stavového prostoru nebo přímé minimalizace určité silové funkce. [6]

2.3.1 Porovnávání tvaru

V případě metody porovnávání tvaru může být sledování objektu prováděno obdobně jako v případě metody porovnávání šablon, která byla zmíněna v oblasti jádrového sledování. Zde je silueta objektu s přidruženým modelem vyhledávána na každém snímku sekvence, kdy se samotné prohledávání vyhodnocuje podle míry podobnosti. Jedná se zde o podobnost mezi objektem a modelem objektu, který je vygenerován z předpokládané siluety objektu, jež je odvozena na základě předchozího snímku sekvence. Model sledovaného objektu bývá obvykle v podobě mapy hran. Tento model je znovu inicializován ihned po té, co je nalezena nová poloha cíle. Opětovná inicializace se provádí z důvodu zachycení maximálních možných změn vzhledu objektu, při přechodu mezi jednotlivými snímky. Průběžné aktualizování modelu umožňuje sledovacímu algoritmu, vypořádat se s problémy, které mohou nastat při změně osvětlení nebo při změně zorného úhlu. Odstavec vychází z [6], kde jsou rovněž popsány jednotlivé metody zastupující tuto kategorii.

2.3.2 Sledování obrysu

Metody založené na principu sledování obrysu opakovaně odvozují počáteční obrys objektu z předchozího snímku a to za účelem nalezení nové lokace na snímku současném. K odvození nového obrysu objektu je nezbytná informace o oblasti sledovaného objektu na současném snímku, která se překrývá s oblastí sledovaného objektu ve snímku předchozím. Sledování pomocí odhadování obrysu může být provedeno užitím dvou odlišných technik. První technikou je použití modelu stavového prostoru k vytváření modelů tvaru obrysu a modelů pohybu. Druhá technika je založena na odhadování obrysu za pomocí minimalizace důraznosti obrysu. Toho je dosaženo užitím přímých minimalizačních technik, jako může být například gradient sestupu. Obě uvedené techniky sledování obrysů jsou detailněji popsány v [6].

3 BAREVNÉ HISTOGRAMY

Aby bylo možné vybraný objekt sledovat v dynamickém obraze, je nejprve nutné zvolit jeho reprezentaci. Tato reprezentace se označuje jako model objektu, který charakterizuje jeho vlastnosti a umožňuje sledovaný objekt identifikovat a odlišit od ostatních potenciálních objektů na snímku. Těmi mohou být například objekty, kterými je sledovaný objekt cloněn nebo také pozadí sledovaného objektu. Jednou z technik pro vytváření modelu objektu je výpočet barevného histogramu.

Pod pojmem barevný histogram se rozumí struktura, která umožňuje zaznamenávat frekvenci výskytu hodnot jednotlivých barevných složek pixelů na analyzovaném obrazu. Histogramy mohou být jedno či více rozměrné, v závislosti na tom, jaký barevný prostor je pro model použit a jaký počet jeho barevných složek je potřeba uchovat. V tomto projektu je použit barevný prostor RGB, který je sestaven ze složek červené (red), zelené (green) a modré (blue) barvy, odtud je odvozen název barevného prostoru RGB. Pro výpočet modelu objektu (histogramu) jsou použity všechny barevné složky, což bylo důvodem volby třírozměrného histogramu.

3.1 Porozumění barevnému prostoru RGB

Jak je uváděno v [9], barevný prostor RGB je barevný model, který je vytvořen kombinací červeného, zeleného a modrého světla, za účelem získání rozsáhlého barevného spektra. Každý barevný odstín, je tedy reprezentován mohutností tří základních barev. Mohutnost jednotlivých barev může být reprezentována pomocí barevné hloubky, kde je pro jednotlivou barevnou složku vyhrazen určitý počet bitů. Složka o počtu 8 bitů pojme 256 (2^8) hodnot, jedná se o rozsah hodnot v intervalu 0 až 255. Míra mohutnosti udává intenzitu s jakou se barva dané složky zobrazí. [9]

3.2 Zpracování obrazu

Každý obraz se skládá z pixelů. Počet pixelů, které daný obraz obsahuje je udáván rozlišením obrazu. Snímané obrazy z kamerového zařízení jsou nejčastěji ve formátu RGB. Jedná se o jeden z barevných prostorů, který se v počítačovém vidění používá. U 24-bitového RGB obrazu nabývají jednotlivé složky R, G a B, hodnot v rozsahu intenzity od 0 do 255 a může tedy reprezentovat 2^{24} odlišných barev. [1]

3.3 Princip histogramu

Barevné histogramy fungují na principu zapisování barevných hodnot každého pixelu na odpovídající pozici histogramu. 24-bitový obraz disponuje počtem 2^{24} různých barevných odstínů. Aby byl histogram schopen pojmout takové množství hodnot, musel by být sestaven z 256-ti sloupců pro každou barevnou složku. Sestavení takto rozlehlého histogramu je možné, nicméně pro účely modelu sledovaného objektu a manipulaci s ním není z hlediska výpočetní časové náročnosti efektivní.

Vyšší efektivnosti lze dosáhnout v případě, kdy je definován histogram s pevnými rozměry. Přesněji se jedná o takové rozměry, jejichž hodnoty jsou výrazně menší než je maximální hodnota barevné složky. U 24-bitového RGB obrazu je tedy pro každou složku maximální hodnota 255. Rozměry histogramu se zadávají v počtu sloupců na jednu barevnou složku, což znamená, že se histogramu definuje počet sloupců, do kterých budou zaznamenávány hodnoty v rozsahu 0 až 255.

Například v případě histogramu o rozměrech 8 x 8 x 8 se rozumí, že každá barevná složka R, G a B v intervalu rozmezí hodnot 0 až 255, bude zastoupena v osmi sloupcích histogramu. Tímto systémem se zajistí, že barevné hodnoty podobné intenzity budou shlukovány do stejných sloupců. To má za následek, že hodnoty podobné intenzity se budou chovat tak, jako by byly identické. S větší rozlohou histogramu se získává přesnější model objektu, ovšem za cenu nárůstu doby výpočtu.

3.4 Výpočet histogramu

Jednorozměrné nebo více rozměrné histogramy je možné implementovat pomocí polí. V tomto projektu je využíván třírozměrný histogram, který je intuitivně implementován pomocí třírozměrného pole podle [1] , nicméně je možné užít i jednorozměrné pole pro třírozměrný histogram, jako popisuje autor [1].

Předpokládá-li se histogram o velikosti 8 x 8 x 8 sloupců, je nejprve nutné získat počet barevných hodnot na jeden sloupec histogramu. Tento počet se získá podílem maximální hodnoty barevné složky ku počtu sloupců reprezentujících tuto barevnou složku. V tomto případě to tedy bude 32 (256 / 8 = 32) barevných hodnot na jeden sloupec. Jelikož je nastaven stejný počet sloupců pro všechny tři (R, G, B) barevné složky, budou mít sloupce všech tří složek stejnou kapacitu, tedy každý ze sloupců pojme škálu 32 hodnot jedné barevné složky. Pokud by byl zvolen histogram o velikosti 8 x 16 x 8, bude kapacita R a B sloupců stejná a pro G bude kapacita 16 (256 / 16 = 16).

Nyní se pro každý pixel na obrazu provede následující. Z každého pixelu se získají jeho hodnoty RGB složek a zapíší se do příslušného sloupce v histogramu. Na ukázku uvažujme například pixel, jehož RGB složky nabývají hodnot: R = 120, G = 5, B = 255. Jestliže se předpokládá histogram o velikosti 8 x 8 x 8, výpočet pozice příslušného sloupce se provede následovně. Složka R bude zapsána do sloupce 4 (120 / 32 = 4), složka G bude na pozici 0 (5 / 32 = 0) a složka B bude umístěna v sloupci 8 (255 / 32 = 8). Z důvodu, že pole se indexují celočíselným datovým typem, je nutné vypočítané pozice sloupců zaokrouhlovat na celá čísla směrem dolů. Nyní jsou zjištěny pozice sloupců pro analyzovaný pixel a jeho barevné složky se tedy mohou zaznamenat do histogramu. V případě třírozměrného pole se inkrementuje hodnota v poli o +1 na pozici [4, 0, 8]. Tímto způsobem se zapíší do histogramu údaje o barevných složkách každého pixelu na analyzovaném obrazu. Obecný vztah pro sestavení histogramu je uveden jako (1) [10]. Je dán histogram s počtem *m* sloupců. Je definováno *n* pozic pixelů $\{x_i\}_{i=1..n}$ s histogramem $\{q\}_{u=1..m}$. Definuje se funkce $c: x^2 \rightarrow \{1..m\}$, která přiřazuje danému pixelu na lokaci x_i určitý index sloupce v histogramu, do kterého aktuálně zkoumaný pixel spadá. Proměnná δ zde představuje Kroneckerovu delta funkci. [1, 4, 5, 10]

$$q_u = \sum_{i=1}^{n} \delta [c(x_i) - u] \qquad (1)$$

3.5 Porovnávání histogramů

Jednotlivé histogramy je možné navzájem porovnávat, respektive porovnávat míru jejich podobnosti. Jedna z technik určená k získání podobnosti histogramů podle [1], je výpočet Bhattacharyya koeficientu.

$$\rho = \sum_{u=1}^{m} \sqrt{p_u q_u} \qquad (2)$$

Bhattacharyya koeficient je možné aplikovat na normalizovaný histogram s identickým počtem sloupců. Nechť jsou dány dva normalizované histogramy *p* a *q*, se shodným počtem sloupců, potom jejich Battacharyya koeficient je dán vztahem (2). Kde *u* udává index aktuálního sloupce a *m* je celkový součet všech sloupců. Normalizace se provádí již u vyhotoveného histogramu takovým způsobem, že hodnota každého sloupce se vydělí celkovým počtem pixelů, z kterého byl histogram sestaven. Například pokud se stavuje histogram z obrazu o velikosti 10 x 10 pixelů, poté do hodnoty každého sloupce histogramu přiřadíme aktuální hodnotu sloupce vydělenou plochou analyzované oblasti, která je v uvedeném případě tedy 100. [1]

4 MEAN SHIFT ALGORITMUS

Pro účely této práce byl vybrán a implementován algoritmus Mean shift, který spadá do oblasti jádrového sledování. Algoritmy této oblasti se vyznačují svou nízkou výpočetní náročností a relativní jednoduchostí. Algoritmy pracují s barevnou reprezentací cíle a nevyžadují žádný externí mechanismus pro určení polohy. Barevnou reprezentací cíle se rozumí, že objekt, který má být algoritmem sledován, je vymodelován pomocí barevného prostoru. Algoritmus již nepracuje s vybraným obrazem jako takovým, nýbrž s jeho modelem. Princip algoritmu je obsažen v jeho názvu, Mean shift algoritmus hledá na každém snímku střed neboli těžiště modelu, které reprezentuje daný objekt. Mean shift algoritmus je popisován v [3], [4] a [5], odkud vychází tato kapitola.

4.1 Princip algoritmu

Mean shift algoritmus patří mezi algoritmy, které vyžadují inicializaci objektu. Před zahájením sledování je tedy nutné vybrat na snímku objekt, který má být algoritmem sledován. Tento objekt se umístí do sledovací oblasti. V této práci je použita sledovací oblast ve tvaru obdélníku, častým tvarem sledovací oblasti bývá také elipsa. Vybraný objekt je označen jako cíl sledování. Ze sledovací oblasti, kterou je vybraný cíl obklopen, je vypočítán histogram způsobem uvedeným v 3.4. Tento histogram plní funkci modelu sledovaného objektu a označuje se jako histogram cíle. Mean shift algoritmus pracuje se staticky daným cílem, což má za následek, že algoritmus bude sledovat takovou barevnou reprezentaci, která byla sestavena při inicializaci cíle. Z tohoto důvodu je cílový histogram uchováván a po celou dobu sledování se nemění. Kromě histogramu cíle je nutné také uchovat polohu vybraného sledovaného objektu. Poloha se uchovává v podobě souřadnic středu sledovací oblasti, která je vztažena k rozměrům daného snímku. Tento celý krok je krokem inicializačním, v této chvíli samostatné sledování neprobíhá. Vlastní Mean shift algoritmus začíná krokem následujícím.

V druhém kroku se předpokládá, že je zpracováván další snímek sekvence, v této fázi je již pravděpodobné, že se vybraný cíl může nacházet na poloze posunuté oproti poloze zachycené v předchozím snímku. V současném snímku je sledovací oblast umístěna na stejnou polohu, v jaké se nacházela v předchozím kroku. Ze sledovací oblasti, která pravděpodobně nyní neobklopuje celý sledovaný objekt, se vypočítá histogram kandidáta na cíl stejným způsobem, jakým byl vypočítán histogram cíle. Histogram cíle je rovněž potřeba uchovat. V tuto chvíli pracuje algoritmus s dvěma histogramy, statickým histogramem cíle a histogramem kandidáta.

V třetím kroku se použije histogram cíle a histogram kandidáta pro výpočet souřadnic těžiště dle vztahů (7) a (8). Pro výpočet těžiště je použito nultého (3) a prvního statického momentu (4), (5). Kde $I(x, y)$ je intenzita jednotlivého pixelu na souřadnicích x a y. Výpočet intenzity se zde provádí pomocí vztahu, pro výpočet váhy vzorku (6) [3], kde q je označení histogramu cíle a p zastupuje histogram kandidáta, u představuje index sloupce histogramu, ve kterém je barva analyzovaného pixelu zastoupena. Dosazením (6) do vztahů (3), (4) a (5) za hodnotu I pro každý pixel ve sledovací oblasti se získá hodnota nultého a prvního statického momentu. Následným dosazením do vztahů (7) a (8), se získají souřadnice těžiště sledovaného objektu. Výpočet souřadnic těžiště může být také zapsán ve tvaru (9) a (10), kde čitatel představuje sumu prvních momentů a jmenovatelem je součet nultých statických momentů, kde je jednotlivá intenzita I zastoupena váhovým vzorkem (6). Uvedené vztahy jsou blíže popsány ve [2] a [3].

Ve čtvrtém kroku se použijí souřadnice těžiště z kroku předchozího, k vystředění sledovací oblasti na novou polohu. Vzdálenost mezi původní polohou středu sledovací oblasti a nově vypočítanou polohou těžiště se nazývá Mean shift vektor [3]. Po posunutí sledovací oblasti o tento vektor, se opět provede výpočet histogramu kandidáta a získá se nové těžiště. Celá procedura posouvání na pozici nově získaného těžiště se provádí, dokud je vektor různý od nuly, nebo dokud je větší než předepsaná minimální vzdálenost. Tato podmínka je podmínkou konvergence a cyklus provádí posouvání a znovu vystřeďování na pozici těžiště, dokud podmínka konvergence není splněna nebo dokud není dosaženo maximálního počtu iterací algoritmu. [3] Hranice maximálního počtu iterací je v tomto projektu nastavena konstantně na hodnotu 20. Tato hranice maximálního počtu iterací zde pomáhá zabránit zacyklení v případě, že je cíl ztracen.

Při ukončení cyklu vystřeďování se předpokládá, že cíl byl nalezen, jeho poloha je zaznamenána a algoritmus očekává další snímek ke zpracování. Dále pokračuje krokem 2 a probíhá dokud není přerušen vnějším zásahem nebo dokud není snímková sekvence u konce.

$$M_{00} = \sum_x \sum_y I(x, y) \quad (3)$$

$$M_{10} = \sum_x \sum_y xI(x, y) \quad (4)$$

$$M_{01} = \sum_x \sum_y yI(x, y) \quad (5)$$

$$w_i = \sqrt{\frac{q_u}{p_u}} \quad (6)$$

$$x_c = \frac{M_{10}}{M_{00}} \quad (7)$$

$$y_c = \frac{M_{01}}{M_{00}} \quad (8)$$

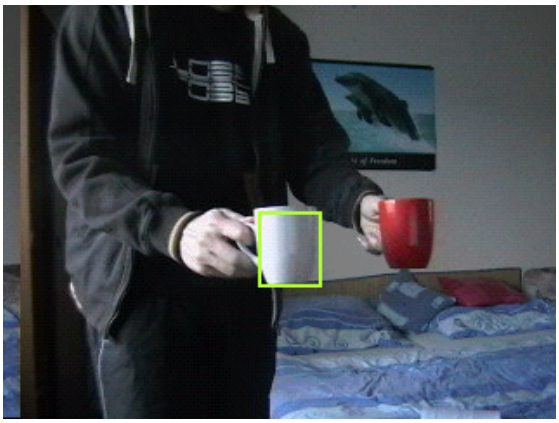
$$x = \frac{\sum_i w_i x_i}{\sum_i w_i} \quad (9)$$

$$y = \frac{\sum_i w_i y_i}{\sum_i w_i} \quad (10)$$

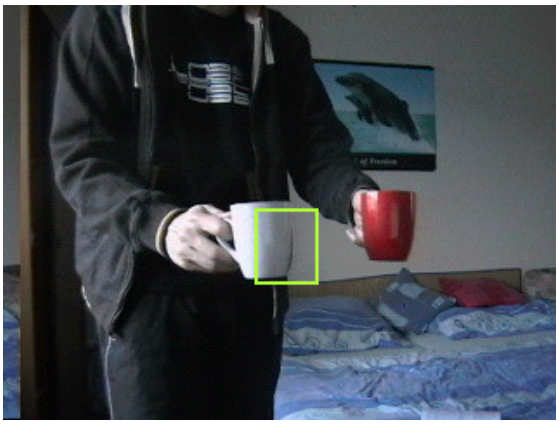
4.2 Mean shift algoritmus

Algoritmus může tedy být stručně shrnut do následujících šesti kroků:

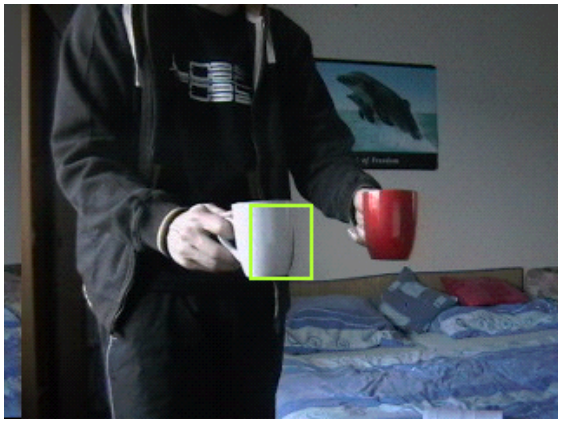
1) Výběr polohy sledovací oblasti, výpočet a zálohování histogramu cíle, zálohování souřadnic středu sledovací oblasti a přechod na nový snímek.
2) Pro následující snímek umístit střed sledovací oblasti na souřadnice z předchozího snímku a výpočet histogramu kandidáta.
3) Z histogramu cíle a histogramu kandidáta, provést výpočet souřadnic těžiště a výslednou polohu zálohovat.
4) Vystředit sledovací oblast na souřadnice těžiště nalezeného v kroku 3 a vypočítat nový histogram kandidáta.
5) Opakovat kroky 3 až 4, dokud nedojde ke konvergenci nebo dokud není dosažen maximální počet povolených iterací.
6) Pro následující snímek pokračovat krokem 2, dokud nenastane konec snímkové sekvence nebo nedojde k vnějšímu zásahu.



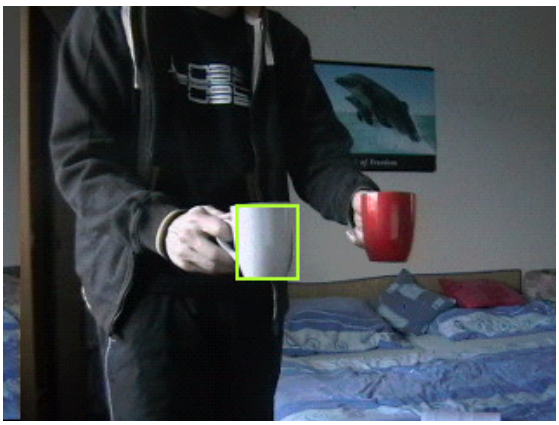
snímek 32



snímek 33, iterace 1



snímek 33, iterace 5



snímek 33, iterace 7

Obr. 4 Mean shift iterace při přechodu mezi dvěma snímky.

Ukázka Mean shift iterací je zobrazena na obrázku 4. Je zde zobrazen přechod ze snímku 32 na snímek 33, kdy snímek 32 by mohl představovat snímek inicializační. Dále je na následujícím snímku 33, možné pozorovat posouvání sledovací oblasti při jednotlivých iteracích, kdy sedmá iterace je závěrečná. V této chvíli algoritmus nalezl těžiště sledovaného objektu a očekává další snímek sekvence ke zpracování.

4.3 Eliminace pozadí

Při sledování objektu v dynamickém obraze je nutné, vypořádat se s nežádoucími elementy, které se na snímcích mohou vyskytovat v podobě změn osvětlení, změn tvarů, změn barvy objektu v podobě clonících objektů nebo ruchu, které představuje pozadí sledovaného objektu.

Pozadí objektu patří mezi jednu z nejhůře překonatelných překážek. Pozadí může infikovat model cíle. V tomto případě tedy cílový histogram, což může později vést ke ztrátě sledovaného objektu. Horším případem je, pokud se sledovaný objekt nachází v prostředí, které má identickou barvu jako daný objekt.

U Mean shift algoritmu se jedná o minimalizaci pozadí na úrovni inicializace cíle. Cílový objekt se tedy označí tak, aby sledovací oblast obsahovala pouze barvy patřící cíli. Označení cílového objektu takovým způsobem, aby sledovací oblast obsahovala pouze pixely patřící cíli, může být v některých situacích velice obtížné. Taková situace může nastat například pokud je objekt sledování částečně cloněn cizím předmětem, nebo pokud je objekt asymetrického tvaru. Přesné označení cílového objektu může být velice obtížné v případě sledování v reálném čase, kdy se objekt sledování neustále pohybuje. V takovém případě je nutné najít vhodný způsob, kterým by se případné zachycené pozadí aspoň z větší části eliminovalo. Řešení tohoto problému, které je navrhováno v [3] a [4], je aplikování filtrovací jádrové funkce při sestavování modelu objektu, tedy při výpočtu

histogramu. Mezi tyto funkce patří například Gausova jádrová funkce (Gaussian kernel), Epanechnikova funkce (Epanechnikov kernel) nebo funkce trojúhelníku.

4.3.1 Aplikace funkce jádra

Důvodem užití jádrových funkcí, je snaha o eliminaci pozadí vybraného objektu. Toho je možné dosáhnout sestavením histogramu cíle takovým způsobem, aby reprezentoval maximum barevných složek, které patří cíli a naopak minimum barevných složek, které patří pozadí [3]. Používaná technika na eliminaci pozadí spočívá v aplikaci jádrových funkcí. Jádrové funkce zde zastupují funkci filtru, který maximalizuje hodnoty kolem středu sledovací oblasti a naopak minimalizuje hodnoty v okrajích sledovací oblasti. Vychází se z předpokladu, že sledovaný objekt se nachází ve středu sledovací oblasti, kdežto po stranách je možný výskyt pozadí a proto je tato část eliminována. Při testování bylo dosaženo nejlepších výsledků užitím funkce Epanechnikova jádra, která byla užita dle [3]. Nicméně je možné aplikovat i ostatní funkce v závislosti na požadavcích sledování.

Aplikace jednotlivých funkcí se provádí v okamžiku výpočtu histogramu. V 3.5 bylo popisováno sestrojení histogramu takovým způsobem, kdy se inkrementovaly jednotlivé hodnoty sloupců histogramu o +1, v závislosti na barevných složkách právě analyzovaného pixelu. Princip jádrových funkcí spočívá v dynamickém přepočítávání inkrementované hodnoty. Přepočítaná hodnota je závislá na tom, v jaké vzdálenosti od středu sledovací oblasti se právě zkoumaný pixel nachází. Tento přepočet je dán právě některou z jádrových funkcí. Například v případě užití Epanechnikova jádra nebo jádra trojúhelníku, jsou hodnoty blíže ke středu vyšší. Kdežto hodnoty od středu se vzdalující, jsou nižší, hodnoty jenž se nenacházejí v intervalu <-1, 1> jsou nulové. Vyšší hodnoty indikují vyšší pravděpodobnost, že barva daného pixelu patří cíli. Naopak nižší nebo nulové hodnoty jsou přiřazovány pozadí objektu, kde je naopak snaha o eliminaci. [3]

Aby bylo možné jednotlivé funkce aplikovat, je nejprve potřeba vypočítat normalizovanou polohu pro každý pixel podle vztahů (11) a (12). Kde proměnné x a y představují polohu aktuálně analyzovaného pixelu, proměnné X_c a Y_c představují polohu středu sledovací oblasti. Následně je nutný výpočet vzdálenosti pixelu od středu sledovací oblasti podle vztahu (13), jehož výsledná hodnota je užita jako parametr dané jádrové funkce. Uvedené vztahy vycházejí ze zdroje [3].

$$X_n = \frac{2(x - X_c)}{\text{šířka oblasti}} \qquad (11)$$

$$Y_n = \frac{2(y - Y_c)}{\text{výška oblasti}} \qquad (12)$$

$$x = \sqrt{X_n^2 + Y_n^2} \qquad (13)$$

$$k = 2(1 - x)/\Pi \qquad (14)$$

Vztah (14) představuje funkci pro výpočet Epanechnikova jádra pro hodnotu $x^2 < 1$, v ostatních případech je hodnota k rovna nule. Z pixelů které splňují podmínku, se obdrží přepočítané hodnoty k, odpovídající průběhu jádrové funkce, kdežto ostatním hodnotám bude přiřazena nula. To znamená, že barevné hodnoty pixelu na vzdálené pozici budou pro další výpočty eliminovány, čímž je docíleno, že sestavený histogram bude více odpovídat barvám, které pravděpodobně patří cíli.

Pokud je tedy při výpočtu histogramu použita jádrová funkce, obecný vztah pro výpočet histogramu je následující (15) [10]. Kde k zastupuje jádrovou funkci a x představuje vzdálenost aktuálního pixelu od středu sledovací oblasti.

$$q_u = \sum_{i=1}^{n} k(\|x_i\|^2)\,\delta[c(x_i) - u] \qquad (15)$$

4.4 Adaptace velikosti sledovací oblasti

Jak bylo popsáno v části 4.1, Mean shift je algoritmus, který pracuje se staticky zadaným cílem a také s pevně danou velikostí sledovací oblasti. Velikost oblasti se po inicializaci již nezmění, což má negativní dopad na efektivitu sledování. Pokud se cíl vzdaluje, sledovací oblast obsahuje příliš mnoho pozadí nebo větší množství podobných objektů, což může vést k selhání. Naopak pokud je cíl příliš blízko, sledovací oblast zabírá pouze malou část sledovaného objektu, což při rychlém pohybu objektu může rovněž vést ke ztrátě cíle.

Jednou z možností jak tento problém řešit, je adaptace velikosti sledovací oblasti [5] na základě vztahu (16), pomocí kterého je možné vypočítat vzdálenost dvou rozdílných barevných histogramů. Kde $\rho[\hat{p}(y),\hat{q}]$ je Bhattacharyya koeficient, který je vypočítán z histogramu cíle a histogramu kandidáta. Na základě výstupů vztahu (16), je možné rozpoznat zda se sledovaný objekt vzdaluje nebo přibližuje a tudíž na přechody mezi vzdálenostmi reagovat zmenšováním, respektive zvětšováním sledovací oblasti. V tomto projektu byla změna velikosti sledovací oblasti prováděna na základě procentuálního poměru velikosti aktuální sledovací oblastí. Nejlepších výsledků bylo dosaženo při změnách velikosti o 5% až 10% při každém přechodu. Výpočet velikosti sledovací oblasti na základě podobnosti histogramu cíle a histogramu kandidáta je blíže popsán v [5].

$$d(y)=\sqrt{1-\rho[\hat{p}(y),\hat{q}]} \qquad (16)$$

Na základě experimentů se tento způsob adaptace neprojevil jako optimální. Z důvodu, že Mean shift pracuje se staticky daným cílem a objekt, který se vzdaluje nebo přibližuje, prochází barevnými změnami (osvětlení, odrazy, clonění, apod.) dochází k znehodnocování kandidáta a funkce (16) tak může uvést vzdálenosti, které mají značnou odchylku. V tomto případě může dojít k nepřiměřeným úpravám velikosti sledovací oblasti, která může následně vést ke ztrátě sledovaného cíle.

Z tohoto důvodu bylo nutné aplikovat efektivnější způsob adaptace velikosti sledovací oblasti. Technika která umožňuje tuto problematiku řešit efektivněji, je popsaná algoritmem Continuously Adaptive Mean Shift (CAMShift) a právě tento algoritmus je dalším předmětem této práce.

5 CONTINUOUSLY ADAPTIVE MEAN SHIFT (CAMSHIFT)

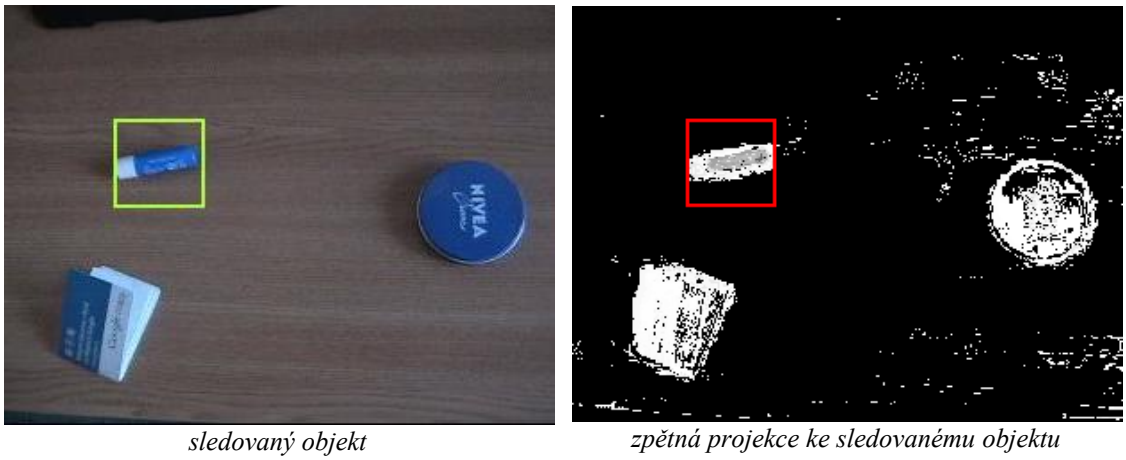
CAMShift je algoritmus, který rovněž spadá do kategorie jádrového sledování a jeho nejpříbuznějším algoritmem je právě algoritmus Mean shift. Nevýhodou Mean shift algoritmu je, že umožňuje sledování pouze statických cílů a neumožňuje přizpůsobování velikosti sledovací oblasti v závislosti na tom, v jaké situaci se sledovaný objekt právě nachází. CAMShift algoritmus byl původně vyvinut jako herní rozhraní pro „hands-free“ počítačové hry, kde byl uplatněn pro sledování tváře. Důvodem bylo, že CAMShift algoritmus je výpočetně nenáročný a poskytuje omezené informace o natočení sledovaného objektu v prostoru. [7, 12]

Na rozdíl od algoritmu Mean shift umožňuje CAMShift dynamické změny velikosti sledovací oblasti [7,12]. Není zde zapotřebí žádného externího zásahu, jako bylo popisováno ve 4.4 v rámci Mean shift algoritmu. CAMshift algoritmus je schopen přizpůsobovat velikost sledovací oblasti implicitně, pomocí funkce nultého statického momentu, jenž je nalezen při výpočtu těžiště [7]. Dynamické změny velikosti sledovací oblasti jsou užitečné zejména v případech, kdy sledovaný objekt mění v průběhu sledování tvar, nebo prochází změnami osvětlení a tudíž se mění odstín barev, které sledovaný objekt obsahuje. Rovněž pokud se objekt vzdálí od objektivu kamerového zařízení, sledovací oblast se zmenší tak, aby obklopovala sledovaný objekt a dojde tak k minimalizaci zachycení pozadí, nebo případných cizích objektů, které obsahují podobné barvy jako cíl sledování.

U Mean shift algoritmu bylo pro výpočet těžiště užito statických momentů. Kde intenzita pixelů byla nahrazena váhovým poměrem mezi histogramem cíle a histogramem kandidáta. To udávalo míru pravděpodobnosti, zda barevné hodnoty právě analyzovaného pixelu patří cíli. U CAMShift algoritmu je implementován odlišný způsob rozložení pravděpodobnosti. Z výpočtů statických momentů pomocí váhového vzorku, který byl užit u Mean shift algoritmu, nebylo možné získat odpovídající hodnoty nultého momentu, které byly nutné pro výpočet nové velikosti sledovací oblasti. Z toho důvodu bylo pro odlišení pixelů, které patří cíli použito techniky zvané zpětná projekce (Back projection).

5.1 Zpětná projekce

Zpětná projekce je technika, která umožňuje vytvoření pravděpodobnostního obrazu z objektu, který je středem zájmu [2]. Zjednodušeně řečeno, udají se barvy na obrazu, které nás zajímají a zpětná projekce je vyobrazí. Příklad zpětné projekce vybraného objektu je znázorněna na obrázku 5.



sledovaný objekt *zpětná projekce ke sledovanému objektu*

Obr. 5 Ukázka vypočítané zpětné projekce ke sledovanému objektu.

5.1.1 Výpočet zpětné projekce

Výpočet zpětné projekce se provádí rovněž na základě barevných histogramů. Pokud je označen nějaký objekt na snímku, provede se sestavení jeho modelu – výpočet barevného histogramu.

Histogram reprezentuje barvy, které je potřeba sledovat. Aby bylo možné vypočítat obraz zpětné projekce, je nutné sestavit také model celého snímku, na kterém se sledovaný objekt nachází. Provede se tedy výpočet histogramu sledovaného objektu a výpočet histogramu celého snímku.

V další fázi se vytvoří nový obraz o stejné velikosti, jako je právě analyzovaný snímek. Pro každý pixel na aktuálním snímku se z histogramu celého snímku a histogramu objektu získá jeho četnost výskytů barevných složek. Podílem četnosti výskytů barevných složek z histogramu cíle a histogramu celého obrazu se získá nová barevná hodnota, která se zapíše do všech tří barevných složek pixelu na novém obrazu. Tímto postupem se projdou všechny pixely na analyzovaném snímku a přepočtem se zapíší na odpovídající pozici pixelů v nově vytvořeném snímku zpětné projekce. Informace o výpočtu a implementaci obrazu zpětné projekce vycházejí ze zdroje [2], kde je technika popisována podrobněji.

5.2 Adaptace sledovací oblasti

V případě Mean shift algoritmu, se sledovací oblast pohybovala přímo po neupravovaném předaném snímku sekvence. Rozlišování zda prohledávané pixely patří cíli nebo nikoliv, bylo zajištěno právě pomocí vztahu (6). Jak již bylo řečeno, tímto způsobem nebylo možné získat takové hodnoty nultého momentu, pomoci kterých by se dala vypočítat velikost sledovaného objektu.

V případě použití zpětné projekce, pro rozlišení pixelů patřící cíli je vypočítán zcela nový snímek. Na snímku jsou světlou barvou vyznačeny pixely, které pravděpodobně patří cíli a naopak tmavou barvou jsou označeny pixely takových pixelů, které se na objektu nevyskytují.

Sledovací oblast se již nepohybuje po původním snímku sekvence, ale pohybuje se právě po vypočítaném obrazu zpětné projekce. Pro výpočet statických momentů je opět použito vztahů jako v případě Mean shift algoritmu (3, 4, 5) s tím rozdílem, že intenzita není nahrazena váhovým poměrem mezi histogramy cíle a kandidáta, ale je použita intenzita jedné z barevných složek, právě analyzovaného pixelu. Například 0 v případě kdy se barva pixelu na sledovaném objektu nevyskytuje a naopak 255 v přídě, kdy barva objektu náleží. Postačující je intenzita právě jedné barevné složky analyzovaného pixelu z důvodu, že všechny tři barevné složky (R, G, B) jsou nastaveny na stejnou hodnotu. Poloha bodu těžiště je rovněž vypočítána dle vztahů (7, 8).

Výpočet velikosti sledovací oblasti je prováděn pomocí vztahu (17), který je uveden a vysvětlen v [7]. Velikost sledovací oblasti se přizpůsobuje na základě aktuální velikosti sledovaného objektu, který je vyobrazen zpětnou projekcí. Vztah (17) vypočítá velikost tak, aby odpovídala celé délce objektu stejné barevné intenzity, ale jen v případě, že mezi pixely stejné intenzity je lineární přechod. V opačném případě vypočítá velikost sledovací oblasti pouze pro část, kde je objekt nepřerušen. Například pokud je objekt z části cloněn, velikost sledovací oblasti se přizpůsobí tak, aby obklopovala necloněnou část objektu, pokud to je možné. [7]

$$s = a * \sqrt{\frac{M_{00}}{256}} \qquad (17)$$

Vztah (17) je použit pro výpočet šířky, kterou zaujímá rozložení pravděpodobnosti sledovaného objektu na obrazu. Kde M_{00} je hodnota nultého momentu a hodnota 256 zde představuje maximální počet hodnot na jednu 8-bitovou barevnou složku. Proměnná *a* zde představuje konstantu, jejíž hodnota byla získána experimentálně a nastavuje se na hodnotu $a = 2$. [7] Bradski [7] použil tento vztah pro výpočet šířky sledovací oblasti v rámci sledování tváře. Autoři [7] vycházejí z faktu, že tvář je eliptického tvaru. Z tohoto důvodu nastavili výšku sledovací oblasti hodnotou 1,2s. Tato technika je dostačující v situacích, kde známe předem tvar sledovaného objektu a je zřejmé, že se tento tvar nijak výrazně nebude měnit. Z tohoto důvodu, není možné aplikovat stejnou techniku u obecné sledování objektu. Důvodem je, že během sledování může objekt jakkoliv měnit tvar nebo odstín. Řešení přináší výpočet velikosti sledovací oblasti uvedený v [8].

Výpočet vychází ze vztahu (17) a je doplněn o výpočty majoritní a minoritní osy a úhlu natočení sledovaného objektu. Za předpokladu, že proměnná *b* vyjadřuje šířku, proměnná *h* výšku a proměnná *s* je velikost. Potom můžeme zapsat závislost, kterou uvádí vztah (18). [8]

$$b*h=s^2 \qquad (18)$$

Sledovací oblast by měla být úměrná k osám, vypočítá se tedy majoritní délková osa *l* a minoritní osa, která udává šířku *w* z těžiště rozložení pravděpodobnosti. [8] Závislost je vyjádřena vztahem (19).

$$b/h=w/l \qquad (19)$$

Z tohoto vztahu je možné vyjádřit vztahy (20) a (21). [8]

$$b=\sqrt{(w/l)}*s \qquad (20)$$

$$h=\sqrt{(l/w)}*s \qquad (21)$$

Zahrne-li se do výpočtu úhel natočení θ, získá se nová šířka objektu *bn* (22) a *výška* objektu *hn* (23). [8]

$$bn=(b*\cos\theta+h*\sin\theta) \qquad (22)$$

$$hn=(b*\sin\theta+h*\cos\theta) \qquad (23)$$

Před výpočty velikostí majoritní a minoritní osy je nutné provést několik dílčích výpočtů, které jsou uvedeny v [7, 10]. V předchozích výpočtech byly použity statické momenty nultého a prvního řádu. Nyní je potřeba výpočet doplnit o statické momenty druhého řádu pro osy *x* a *y,* které jsou uvedeny ve vztazích (24 , 25) a vztahu (26).

$$M_{20}=\sum_x\sum_y x^2 I(x,y) \qquad (24)$$

$$M_{02}=\sum_x\sum_y y^2 I(x,y) \qquad (25)$$

$$M_{11}=\sum_x\sum_y xyI(x,y) \qquad (26)$$

$$a=\frac{M_{20}}{M_{00}}-(x_c)^2 \qquad (27)$$

$$b=(\frac{M_{11}}{M_{00}})-x_c\,y_c \qquad (28)$$

$$c=(\frac{M_{02}}{M_{00}})-y_c^2 \qquad (29)$$

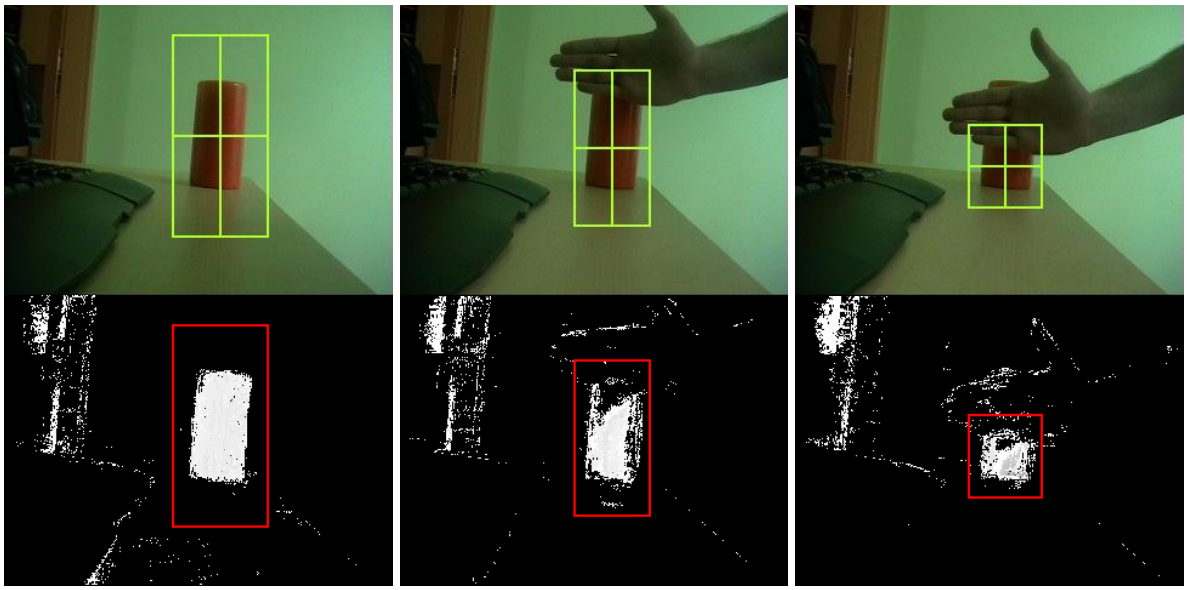
Po dosazení dílčích výpočtu (27), (28) a (29) do vztahu (30), obdržíme úhel natočení rozložení pravděpodobnosti v radiánech, na snímku zpětné projekce. [7, 10]

$$\theta=\frac{1}{2}\tan^{-1}(\frac{b}{a}-c) \qquad (30)$$

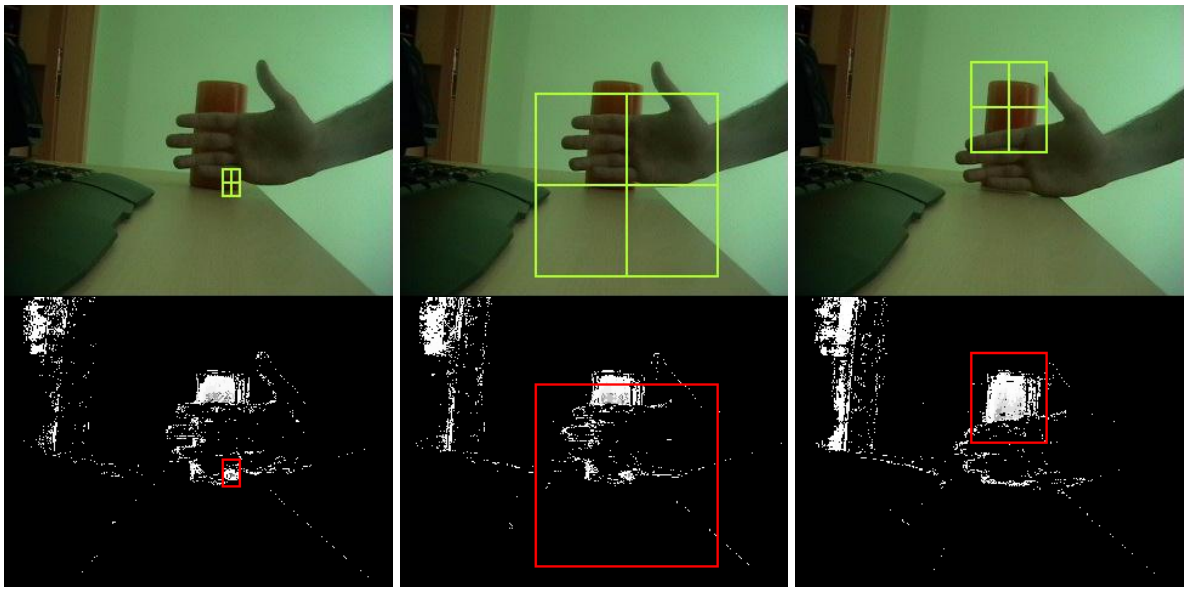
Následují vztahy pro výpočet obou os, kdy vztah (31) určuje délku majoritní osy a vztah (32) délku osy minoritní. [8]

$$l_1 = \frac{\sqrt{(a+c)+\sqrt{(b^2+(a-c)^2)}}}{2} \qquad (31)$$

$$l_2 = \frac{\sqrt{(a+c)-\sqrt{(b^2+(a-c)^2)}}}{2} \qquad (32)$$



snímek 1 *snímek 140* *snímek 175*



snímek 386 *snímek 387* *snímek 492*

Obr. 6 Ukázka adaptace sledovací oblasti při clonění sledovaného objektu. Cíl je identifikován algoritmem CAMShift.

S využitím funkce pro adaptivní změnu velikosti sledovací oblasti v závislosti na aktuálním stavu objektu, je možné částečně detekovat situace, kdy je sledovací algoritmus ztracen. Pokud se velikost sledovací oblasti zmenší na předepsanou hranici, je možné predikovat ztrátu cíle. V takovém případě je možné aplikovat detekční algoritmy k znovu zachycení sledovaného objektu, příkladem mohou být algoritmy určené k detekování tváře. V této práci není zakomponován žádný detekční

algoritmus, nýbrž se ke znovu zachycení objektu používá vlastnost algoritmu Mean shift. V případech, kdy se sledovací oblast zmenší na předepsanou minimální hranici, implementovaný algoritmus zareaguje okamžitým roztažením sledovací oblasti ve snaze zachytit část sledovaného objektu, jehož barvy korespondují s histogramem cíle v místě, kde byl cíl ztracen. V případě, že se ve sledovací oblasti takový objekt nebo jeho část vyskytne, algoritmus provede nezbytný počet iterací k dosažení středu cílového objektu a sledovací oblast opět přizpůsobí aktuální velikosti znovu zachyceného cíle. Tímto způsobem lze relativně efektivně reagovat na ztrátu sledovaného objektu. Selhání může nastat v případě, kdy se na snímku nachází více objektů podobného barevného odstínu. V takovém případě algoritmus zachytí nejbližší objekt, jehož barvy korespondují s barvami histogramu cíle.

Na obrázku s číslem 6 je zobrazeno, jak algoritmus CAMShift manipuluje s velikosti sledovací oblasti v situacích, kdy je sledovaný objekt cloněn cizím objektem jiné barevné reprezentace. Za povšimnutí stojí situace na snímcích s čísly 386 a 387. Na snímku 386 se velikost sledovací oblasti blíží k minimální hranici. To je vyhodnoceno jako ztráta sledovaného objektu a dochází k rychlému roztažení sledovací oblasti v místě, kde algoritmus selhal ve snaze zachytit část objektu, který má stejnou barevnou reprezentaci jako cíl. V tomto případě bylo znovu nalezení objektu úspěšné. Jak je vidět na snímku 387, roztažená sledovací oblast zachytila část barevné reprezentace v horní části. Na to navazuje několik mean shift iterací k nalezení středu viditelné části objektu.

5.3 CAMShift algoritmus s dynamickým modelem cíle (CAMShift DM)

Continuously adaptive mean shift je charakteristický tím, že v průběhu sledování dynamicky mění rozměry sledovací oblasti. To je hlavním rozdílem oproti algoritmu Mean shift. [12] Jak bylo popsáno výše, algoritmy Mean shift a CAMShift pracují s histogramem cíle, který se sestaví v inicializační fázi, kdy je cíl manuálně vybrán. Po celou dobu sledování se již histogram cíle nezmění a slouží k výpočtu pravděpodobnostního obrazu [12] (back projection) v případě CAMShift algoritmu nebo k porovnání kandidáta v případě Mean shift algoritmu. Výhoda statického histogramu je zřejmá, algoritmus má přesnou informaci o vzhledu cíle. Problém nastává v případě, kdy sledovaný objekt změní vzhled. Ke změně vzhledu objektu může snadno dojít například změnou osvětlení. Pokud byl histogram cíle vytvořen v dobře osvětleném prostředí a v průběhu sledování intenzita tohoto osvětlení poklesne, odstín sledovaného objektu se může změnit natolik, že pro sledovací algoritmus bude objekt téměř nebo zcela nerozpoznatelný. Ukázka sledování při změně osvětlení je zachycena na obrázcích s čísly 45, 46 a 47.

Jedním ze způsobů, kterým je možné řešit problém se změnou osvětlení nebo případně změnou tvaru, je průběžné aktualizování histogramu cíle a tím možnost zaznamenávat případné změny modelu. Možnost aktualizace histogramu cíle nebyla u algoritmu Mean shift možná z důvodu, že algoritmus pracuje se statickou velikostí sledovací oblasti. V případě kdy by byl malý objekt ohraničen podstatně větší sledovací oblastí, došlo by při aktualizaci cílového histogramu k infikováním nežádoucím pozadím a sledovaný objekt by byl ztracen. Z důvodu, že algoritmus CAMShift disponuje dynamicky měnící se velikostí sledovací oblasti, je možné histogram cíle průběžně aktualizovat. Tato aktualizace se provádí po ukončení mean shift iterací, kdy je nalezen střed sledovaného objektu. Tento upravený algoritmus je v této práci označován jako CAMShift s dynamickým modelem cíle.

Ačkoliv je objekt sledovaný CAMShift algoritmem ohraničován přesněji, než v podání algoritmu Mean shift, je každá aktualizace histogramu cíle velice náchylná na znehodnocení pozadím objektu nebo jiným rušením. V případě, že algoritmus započítá do histogramu cíle část pozadí objektu, začne sledovat i barevnou reprezentaci, která patří pozadí a následně může dojít k selhání. Z důvodu minimalizace šance na znehodnocení cílového histogramu, byl aplikován nový typ filtrovaného histogramu, v kterém dochází k vyvážení pozadí cíle.

5.3.1 Eliminace pozadí

Stejně jako u Mean shift algoritmu, je i zde nutné eliminovat nežádoucí vlivy pozadí. Je možné opět použít profilu Epanechnikova jádra, stejně jako v předchozím algoritmu. Nicméně Epanechnikův profil neeliminuje pozadí zcela dostatečně. Z důvodu, že v tomto případě je algoritmem na každém snímku aktualizován cílový histogram, je velké nebezpečí, že při aktualizaci zachytí část

pozadí, jehož barvu započítá do histogramu cíle. V takovém případě se barva pozadí bude jevit jako součást sledovaného objektu a algoritmus začne sledovat i předměty nebo pozadí této barvy, což vede k selhání algoritmu a následné ztrátě sledovaného objektu. Možnost infikování cílového histogramu pozadím, je nejslabším místem popisovaného algoritmu a eliminace je v některých případech velmi obtížná.

Eliminace pozadí se provádí stejně jako v předchozím případě na úrovni výpočtu histogramu. Je zde použita filtrace pomocí jádrové funkce a navíc se aplikuje technika vyvážení histogramu cíle a histogramu pozadí uváděná v [3, 4, 10]. Z principu je odvozen název histogramu: histogram vyváženého pozadí (Background weighted histogram). Použití a implementace vyváženého histogramu je realizována podle [3].

5.3.1.1 Vyvážený histogram

Princip vyvážení histogramu spočívá ve výpočtu dvou odlišných histogramů. [3] Prvním je histogram cíle neboli popředí, jehož výpočet se nijak neliší od výpočtů v předchozích částech této práce (3.4 a 4.3.1).

V případě výpočtu histogramu pozadí se konvence poněkud liší. Již bylo zmíněno, že histogram cíle se vypočítá ze sledovací oblasti, která cíl obklopuje. Je zřejmé, že pokud je požadován výpočet histogramu pozadí, je nutné provést výpočet z oblasti, která obklopuje sledovaný objekt, respektive sledovací oblast, pouze obsah sledovací oblasti cíle nesmí být do výpočtu zahrnut. Pro vyřešení tohoto problému se nabízejí dvě varianty.

První variantou je provést výpočet histogramu pro celý snímek s vyjímkou sledovací oblasti cíle. Výpočet histogramu a následné analyzování pro takto rozsáhlou plochu, jakou může být například snímek s rozlišením 640 x 480 pixelů, je výpočetně časově náročné a z důvodu optimalizace pro sledování v reálném čase nevyhovující.

Druhým a efektivnějším způsobem, je umístit kolem oblasti cíle, sekundární sledovací oblast, která bude n-krát větší než oblast cíle, kterou obklopuje [3, 10]. Sledovací oblast musí být dostatečně velká na to, aby zachytila maximální množství pozadí v okolí cíle a zároveň musí být dostatečně malá, aby její analýza nebyla příliš výpočetně náročná. V tomto projektu se velikost sledovací oblasti volí 3x větší než sledovací oblast cíle a tato velikost se dynamicky mění v závislosti na aktuální velikosti oblasti cíle.

K sestavení obou histogramů, postačí analyzovat sledovací oblast a výpočet větvit dle podmínky, zda aktuální pixel spadá do oblasti sledovací nebo oblasti cíle. [3] Tímto způsobem je možné ušetřit výpočetní čas sestavení histogramu, na rozdíl od situace kdy by se pro každý histogram analyzovala odpovídající oblast zvlášť. Po sestavení histogramů se histogramy normalizují a aplikuje se vztah (33) [3, 10] na vypočítaný histogram pozadí.

$$\{\hat{w}_u = min(\frac{\hat{O}^x}{\hat{O}_u}, 1)\}_{u=1\ldots m} \qquad (33)$$

Kde $\hat{O}^x$ je nejmenší nenulový vstup, obsažený v histogramu pozadí a $\hat{O}_u$ je hodnota odpovídajícího sloupce histogramu pozadí. Konečný vztah pro vyvážený histogram je proto dán vztahem (34), který je uveden v [10].

$$q_u = \hat{w}_u \sum_{i=1}^{n} k(\|x_i\|^2)\, \delta[c(x_i) - u] \qquad (34)$$

6 IMPLEMENTACE ALGORITMŮ

Algoritmy, které jsou popisovány v této práci, byly vybrány pro svou efektivitu a rychlost. Pro otestování a porovnání jednotlivých sledovacích algoritmů, byly zmíněné algoritmy implementovány. Implementace je provedena v objektově orientovaném jazyce C#, který využívá platformu .NET framework a umožňuje tak vyšší rychlost vyvíjení. Architektura celé aplikace těží z předností objektově orientovaného programování.

Zpracování obrazu je obecně náročné z pohledu výpočetního výkonu. Jedním z požadavků zadání této práce bylo implementovat algoritmus, který bude schopen sledovat objekt na snímku o rozlišení 640x480 pixelů s minimální dobou výpočtu 5 milisekund na snímek, na běžném počítači. Z tohoto důvodu byly voleny rychlé sledovací algoritmy a při samotné implementaci byla snaha o maximální optimalizaci. Běžný počítač je relativní pojem, aplikace byla vyvíjena a testována na tři roky starém stroji osazeným procesorem Intel Core 2 Duo E8400 s frekvencí 3,00 GHz. Aplikace dosahuje výborných výsledků a je možné ji proto použít za účelem sledování v reálném čase i na méně výkonných strojích. V této kapitole bude popsána architektura aplikace a klíčové části implementace.

6.1 Analýza aplikace

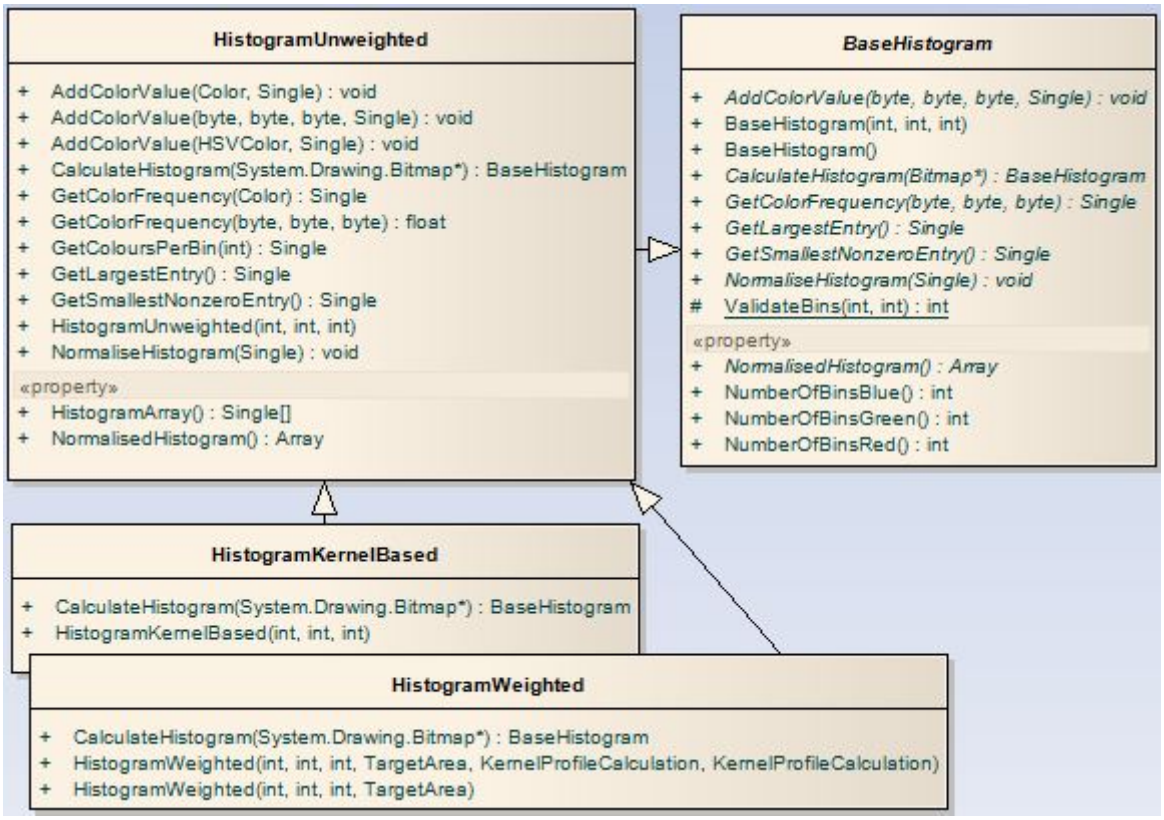
Aplikace je rozdělena podle logiky zpracování do čtyř projektů. Mezi tyto projekty patří HistogramGenerator, Tracker, TrackerGraphic a UserInterface. V celé aplikaci je kladen důraz na univerzálnost použití, je zde plnohodnotně využíváno dědičnosti a rozhraní. Architektura aplikace byla navržena tak, aby jednotlivé části s minimálními modifikacemi posloužily jako základ pro silnější algoritmy, které jsou založeny na obdobném principu sledování. Společně s aplikací pro sledování objektu, je dodávaná aplikace na zachytávání jednotlivých snímků z kamerového zařízení, která snímky ukládá v požadovaném formátu. Popis této aplikace však není zahrnut do konceptu této práce. Za zmínku stojí, že aplikace pro zachytávání snímku využívá pro získávání snímků z kamery stejné funkcionality jako v případě sledování v reálném čase. Tato funkcionalita je dosažena použitím knihoven AForge [11]. Kromě využití uvedených externích dynamických knihoven pro zachytávání snímků, je aplikace implementována pouze pomocí knihoven frameworku .NET. V následující části budou stručně popsány projekty, z nichž se aplikace skládá. Jsou zde také zobrazeny diagramy tříd. Z důvodu, že se samotná aplikace sestává z více než třiceti tříd, budou zde zobrazeny pouze třídy klíčové. V uvedených diagramech tříd, nejsou z důvodů úspory místa zobrazeny instanční proměnné (fieldy). Detailní popis funkčností metod je uveden v přiložených zdrojových kódech, kde je každá metoda okomentována dostatečně.

6.1.1 Projekt HistogramGenerator

Projekt HistogramGenerator obsahuje třídy, které jakkoliv souvisejí s výpočtem nebo manipulací histogramu. Jsou zde obsaženy třídy pro výpočet všech tří typů histogramů, které dědí ze stejného abstraktního předka. Diagram tříd histogramů je zachycen na obrázku 7. V tomto projektu jsou také obsaženy jedny z nejdůležitějších funkcí aplikace a to funkce pro výpočet zpětné projekce a polohy těžiště.

6.1.2 Projekt Tracker

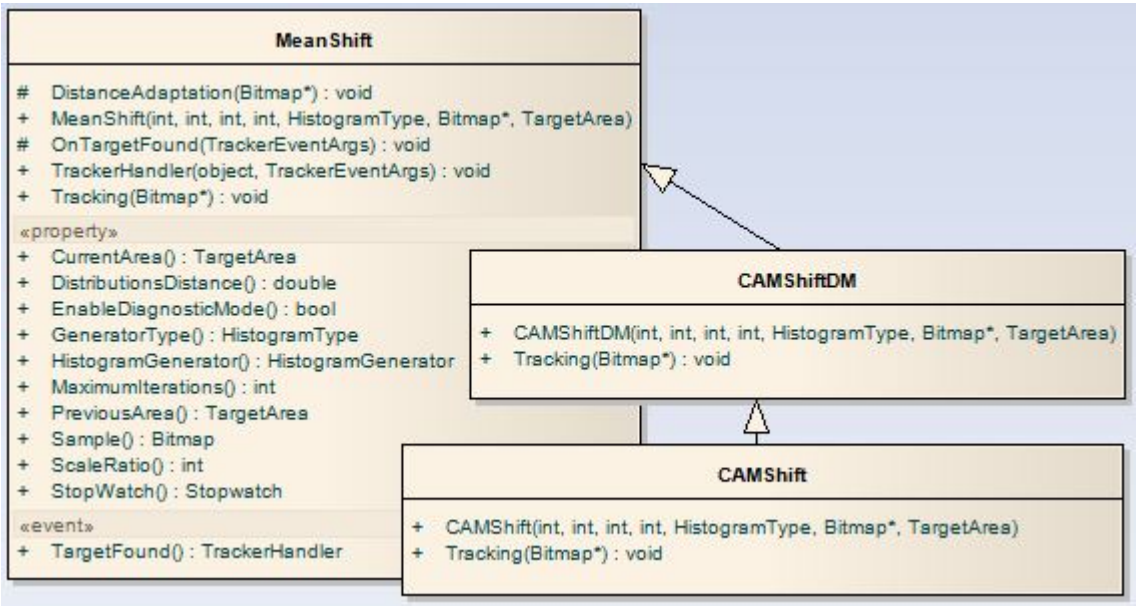
Projekt Tracker obsahuje jednotlivé třídy sledovacích algoritmů. Třída MeanShift tvoří bázovou třídu pro zbývající dva algoritmy CAMShift a CAMShift s dynamickým modelem cíle. Diagram tříd všech tří typů algoritmů je zachycen na obrázku 8. Za povšimnutí zde stojí, že v případě odvozených algoritmů stačilo překrýt pouze jedinou virtuální metodu.



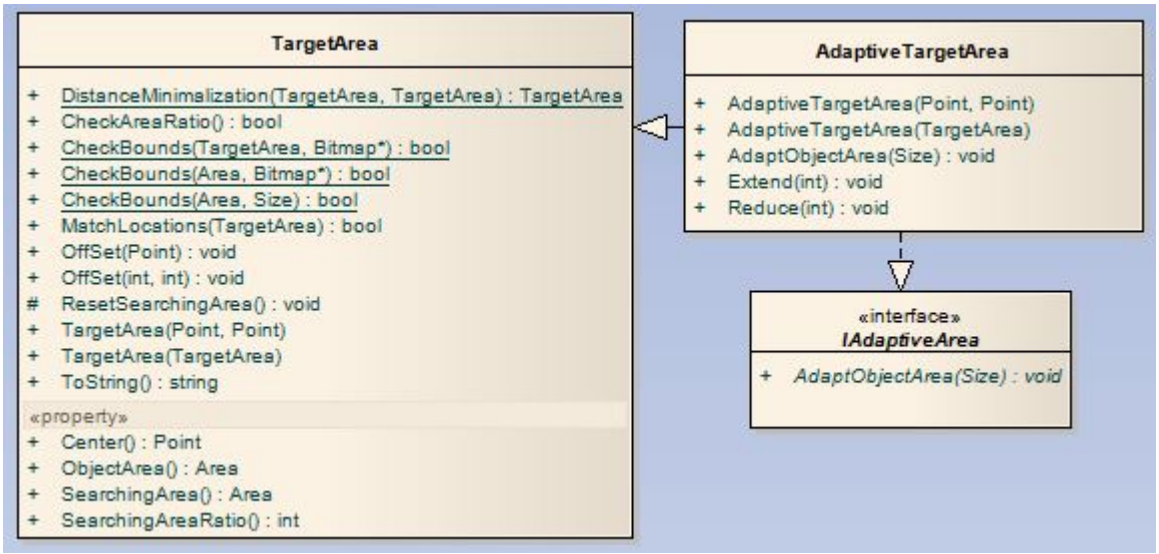
Obr. 7 Diagram tříd - hierarchie tříd histogramů.

6.1.3 Projekt TrackerGraphic

Tento projekt obsahuje třídy představující sledovací oblast cíle a funkcionality pro manipulaci s ní. Jsou zde obsaženy dva typy sledovacích oblastí, bázová TargetArea, jenž má fixní rozměry a je použita v případě MeanShift algoritmu. Potomek této třídy AdaptiveTargetArea doplněna o adaptaci rozměrů sledovací oblasti je využita ve zbývajících dvou algoritmech. Hierarchie je vyobrazena diagramem tříd na obrázku 9.



Obr. 8 Diagram tříd - hierarchie tříd sledovacích algoritmů.



Obr. 9 Diagram tříd - hierarchie tříd sledovací oblasti.

6.1.4 Projekt UserInterface

Poslední z projektů obsahuje třídy a formuláře určené pro interakci s uživatelem. Kromě aplikačních formulářů, je zde wrapper FramerCapturer pro již zmíněné knihovny AForge, sloužící k zachytávání snímků z kamerového zařízení. Je zde také obsažena třída zastávající funkci jednoduchého bufferu v případě sledování v reálném čase. Hlavním z formulářů je okno s názvem ObjectTrackingForm. V tomto formuláři je obsažena globální obsluha sledovacích algoritmů, sestavování algoritmů, načítání snímku z úložiště a jejich dávkování, je zde obsaženo označování cíle a podobně.

6.2 Optimalizace

Jak již bylo řečeno v úvodu, otázka optimalizace byla při implementaci klíčová a z tohoto důvodu bylo nutno maximálně zefektivnit zápis zdrojového kódu.

Jedním z opatření, které značně urychlí běh aplikace, je odstranění try...catch bloků. O této technice obecně platí, že je velice pomalá a pokud je tedy použita ve funkci, jenž se volá několikrát za vteřinu, může dojít k značnému úbytku výpočetní rychlosti. Méně známým problémem, který může mít v některých případech negativní dopad na rychlost výpočtu, je explicitní přetypování. Prefixové přetypování je pomalejší, než přetypování operátorem „as“. Nevýhodou je, že je potřeba ošetřit přetypovanou hodnotu na null, pomocí operátoru „is“.

V celé aplikací se neustále pracuje s obrazem, každý obraz je podroben analýze, kde je potřeba často prozkoumat všechny pixely na obrazu. Platforma .NET framework nabízí statické metody Bitmap.GetPixel a Bitmap.SetPixel. Nevýhodou této dvojice metod je, že jsou velice pomalé. Problém spočívá v tom, že v případě aplikací metod na polohu zkoumaného pixelu v obrazu, dojde k zamknutí celé bitmapy v pamětí, následuje akce volané metody a poté je bitmapa opět uvolněna. Tato režie, se opakuje v případě každého pixelu. Tento problém je možné obejít způsobem, kdy se celá bitmapa v paměti zamkne pouze jednou a v tom se projdou všechny pixely, vykonají se příslušné operace a poté se celá bitmapa uvolní. Tato technika mnohonásobně urychluje práci s obrazy, jediným problémem je, že je nutné použit ukazatele do paměti, což je důvodem proč musí být kód v nechráněném bloku „unsafe“. V následující ukázce je popsána metoda určená ke čtení jednotlivých hodnot pixelů z paměti.

```
static unsafe void GetPixels(ref System.Drawing.Bitmap bmp)
{
            //zamknutí bitmapy v paměti
            BitmapData data = bmp.LockBits(
                new Rectangle(0, 0, bmp.Width, bmp.Height),
                ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
            int* pos;
            Color clr;
            for (int y = 0; y < data.Height; y++) {
                // vypočítá pozice, kde začínají data daného řádku
                pos = (int*)((int)data.Scan0 + (y * data.Stride));
                for (int x = 0; x < data.Width; x++) {
                    //získání hodnot argb z dané pozice v paměti
                    //a inkrementování pozice
                    clr = Color.FromArgb(*pos++);
                }
            }
            bmp.UnlockBits(data);//odemknutí Bitmapy z paměti
}
```

Stejná technika se použije v případě, kdy je na pozici potřeba přiřadit definovanou hodnotu. To se používá při sestavování obrazu zpětné projekce, kdy je vytvářen zcela nový snímek s vypočítanými barevnými hodnotami.

Z hlediska optimalizace je vytváření jakékoliv datové struktury nebo typu vždy časově náročnější než práce se základními typy. V tomto případě je pro každý pixel na obrazu vytvářena vždy nová struktura Color. Jednotlivé barevné složky lze z dané pozice v paměti získat i rychlejším způsobem a to pomocí aplikace bitového posunu přímo na hodnotu v paměti. Tato operace opět ušetří několik milisekund z celkového výpočtu a její konstrukce je zobrazena níže, kde parametr pos představuje odkaz na polohu v paměti, parametry r, g, b představují jednotlivé hodnoty barevných složek červené, zelené a modré barvy.

```
static unsafe void IntArgbToByteValues(int* pos, out byte r, out byte g, out byte b)
{
            b = (byte)(*pos & 0xFF);
            g = (byte)((*pos >> 8) & 0xFF);
            r = (byte)((*pos >> 16) & 0xFF);
}
```

Pomocí bitového posunu je také možno zpětně získat celočíselnou hodnotu barvy, která bude zapsána na patřičné místo do paměti. Tato hodnota je získaná z jednotlivých hodnot barevných složek v bajtech. Metoda zastávající tuto funkcionalitu je zobrazena níže. Pro parametry platí stejná konvence jako v předchozí metodě, navíc je zde uveden parametr a – alpha, který představuje průhlednost.

```
static unsafe int ByteValuesToIntArgb(byte a,byte r, byte g, byte b)
{
            return (a << 24) + (r << 16) + (g << 8) + (b << 0);
}
```

Z pohledu optimalizace bylo potřeba vyřešit problém, který je spojen se sledováním v reálném čase. Zejména na méně výkonných strojích může nastat situace, kdy zvolený sledovací algoritmus zpracovává jednotlivé snímky pomaleji, než je rychlost zasílání snímků z kamerového zařízení. Z tohoto důvodu bylo nutno implementovat buffer, který snímky z kamery zachytává do fronty a algoritmus si jednotlivé snímky odebírá, jakmile je připraven na další výpočet.

Toto opatření řeší problém, kdy je sledovací algoritmus puštěn na krátkou dobu. Snímky se ukládají do paměti a po ukončení sledování se garbage collector postará o jejich odstranění. Problém nastává v situaci, kdy sledovací algoritmus běží výrazně déle. V takovém případě buffer se snímky, které čekají na zpracování se neustále rozrůstá a dochází k zahlcování paměti. Z tohoto důvodu bylo nutné opatření, kdy bufferu byla nastavena pevná hranice maximálního počtu vkládaných snímků. V případě, že se počet snímků obsažených v bufferu blíží k této hodnotě, ukládání do fronty se přesměruje na zápis na pevný disk. Vytvoří se dočasná složka s umístěním C:\\TrackerBufferTemporary, do této složky se začnou s pořadovým číslem zapisovat snímky z kamerového zařízení, které se již nevejdou do bufferu. Sledovací algoritmus, dále odebírá snímky z bufferu a tedy dochází k uvolňování místa. Ve chvíli, kdy je buffer ze tří čtvrtin vyprázdněn, je vyvolána událost oznamující, že je buffer připraven k doplnění snímků. V obslužné metodě této události se ihned spouští načítání snímků, načtou se soubory z pevného disku, seřadí se podle pořadí vložení a buffer se doplní podle vypočítaného aktuálního volného místa. Položky, kterými se buffer doplnil se ihned z pevného disku odstraní. Když sledování skončí nebo je aplikace vypnuta, dočasná složka je odstraněna včetně jejího obsahu. Z důvodu, aby režie ukládání, načítání a mazání snímků neovlivnila rychlost běhu sledovacího algoritmu na hlavním vlákně, bylo nutno tyto funkcionality provádět na vláknech odlišných.

Implementací bufferu s pevnou kapacitou a dodatečným uložením na pevný disk, byl vyřešen problém zahlcování paměti a aplikaci je tedy možné nasadit v situacích vyžadující dlouhodobější působení při sledování v reálném čase.

7 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

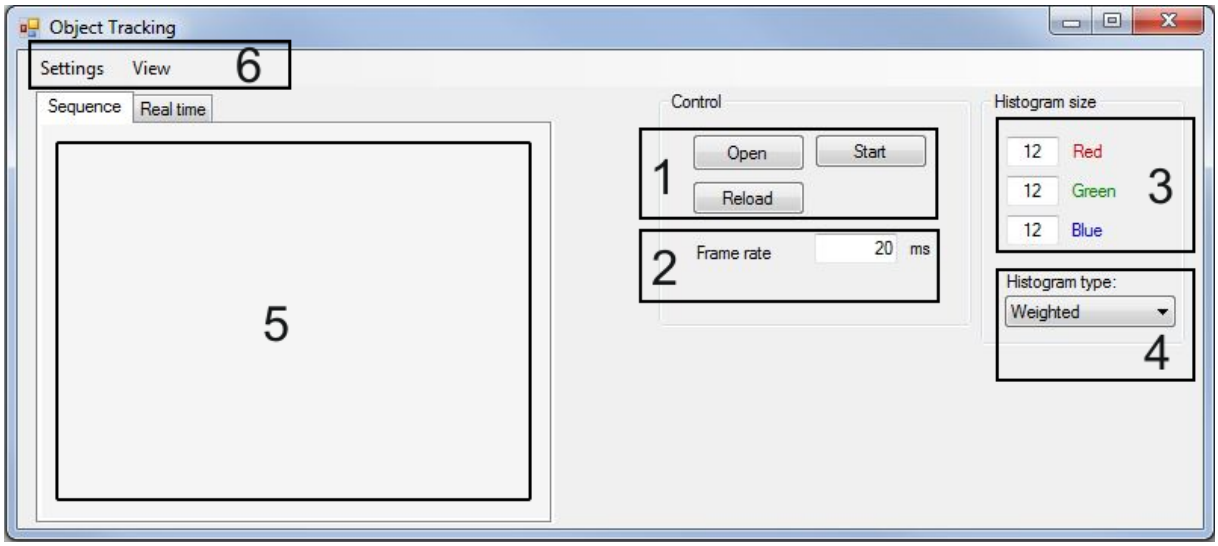
V této kapitole bude popsáno grafické uživatelské rozhraní aplikace, které implementuje výše uvedené sledovací algoritmy. Grafické uživatelské rozhraní je implementováno v prostředí WinForms.

Aplikace je rozdělena do tří formulářů, kde na hlavním formuláři (obr. 10) jsou umístěny ovládací prvky a prvky pro nastavování aplikace. Formulář výstupu algoritmu (obr. 14) zobrazuje informace o průběhu sledování a umožňuje diagnostikovat současný stav algoritmu. Třetí formulář (obr. 15) je možné zobrazit pouze v případě, kdy je prováděn výpočet zpětné projekce, tedy v případě aplikace algoritmu CAMShift nebo algoritmu CAMShift s dynamickým modelem cíle. Formulář s náhledem zpětné projekce, je rovněž velice vhodný pro diagnostiku, na jejíž základě je možné predikovat přesnost sledovacího algoritmu v závislosti na pozadí a vzdálenosti sledovaného objektu. Výhodou zobrazení zpětné projekce je také fakt, že napomáhá k bližšímu pochopení funkcionality sledovacího algoritmu.

7.1 Hlavní formulář aplikace

Ovládání formuláře zde bude detailně popsáno v šesti bodech.

1. Tato část zastává funkci ovládacího panelu. Jsou zde obsaženy tři tlačítka. Tlačítko Open, slouží k namapování umístění sekvence se snímky. Po kliknutí na tlačítko je zobrazeno dialogové okno vyzývající uživatele k výběru adresáře, který obsahuje požadovanou sekvenci. Nutno podotknout, že jednotlivé snímky sekvence v adresáři musí být číselně pojmenovány, vzestupně podle jejich pořadí v sekvenci s příponu JPG. Pokud tato podmínka nebude splněna, bude sledování



Obr. 10 Hlavní formulář aplikace.

přerušeno s příslušným chybovým hlášením. V opačném případě bude v oblasti s číslem 5 zobrazen první snímek sekvence. Nutno podotknout, že v případě sledování v reálném čase budou tlačítka jak Open tak i Reload deaktivovány. Důvodem pro deaktivaci je fakt, že tato dvě tlačítka zastávají funkci pro práci s externím úložištěm snímku, ty se ovšem v případě sledování v reálném čase zpracovávají implicitně v rámci aplikace pomoci bufferů. Aplikace si uchovává předchozí stav, za účelem maximálního usnadnění ovládání a v případě opakovaného kliknutí na tlačítko Open, bude automaticky otevřena cesta k adresáři dříve vybrané sekvence. Tlačítko Reload má funkci znovu načtení prvního snímku frekvence. Pokud je tlačítko stisknuto v průběhu sledování, je sledování vždy přerušeno. Tlačítko Start má funkci dvou polohového spínače, v případě využití úložiště k získávání snímků, je jím možné sledování spustit a také pozastavit. Dříve než je sledování spuštěno, je nejprve nutné na zobrazeném snímku označit objekt, který se má sledovat. Zobrazený snímek je reprezentován

oblastí označenou číslem 5. Po té co je tlačítko Start stisknuto, začne načítání snímků z úložiště v zadaných intervalech a popis tlačítka je změněn na text Stop. Po opětovném stisknutí, je sledování pozastaveno. V případě Real time sledování, není tlačítko Start dostupné a sledování je spuštěno implicitně, po výběru požadovaného cíle na oblasti 5. Pokud je objekt vybrán a probíhá sledování, tlačítko Start se přepne do režimu Stop a v případě jeho stisknutí, je sledování ukončeno a aktivace sledování je možné pouze opětovným vybráním objektu.

2. Ovládací prvek označen číslem 2 umožňuje redukování rychlosti zasílání snímků z úložiště, ke zpracování sledovacím algoritmem. Stejně tak jako v případě tlačítek Open a Reload, je tento ovládací prvek bezpředmětný v případě sledování v reálném čase. V tomto případě jsou snímky zpracovávány tak rychle, jak jím umožní výkon počítače a snímková frekvence aktivní kamery. Textové pole ovládacího prvku je validováno na kladné nenulové numerické hodnoty, které lze modifikovat i v průběhu sledování.

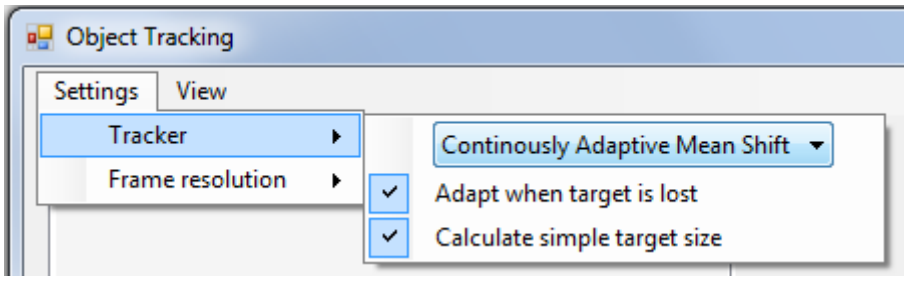
3. Množina ovládacích prvků označená číslem 3 hraje klíčovou roli v sestavování modelu sledovaného objektu. Jak bylo popsáno v kapitole 3, v této práci je použito modelování sledovaného objektu pomocí třírozměrného histogramu. Pomocí těchto ovládacích prvků je možné definovat pevné rozměry histogramu, respektive počet sloupců na jednu barevnou složku. Všechny tři textové pole jsou validovány na nezáporné numerické hodnoty. Optimální rozměry histogramů pro sledování jsou v rozmezí hodnot 6 až 22 sloupců na barevnou složku.

4. Prvek s označením 4 umožňuje výběr jednoho ze třech typů histogramů, který bude využit pro sestavování modelů. Po stisknutí tohoto tlačítka, je zobrazena kontextová nabídka, která reprezentuje typy histogramů: Unweighted (nevyváženy), KernelBased (s aplikací jádra), Weighted (vyvážení pozadí). Optimální typ histogramu pro algoritmus CAMShift, je typ histogramu KernelBased, algoritmus CAMShift s dynamickým modelem cíle dosahuje nejpřesnějších výsledků s užitím histogramem typu Weighted. Základní MeanShift algoritmus dosahuje nejpřesnějších výsledků použitím stejných typů histogramu, jako ve zmíněných algoritmech.

5. Tento ovládací prvek umožňuje zobrazovat jednotlivé snímky sekvence. V této části aplikace je možné také označit objekt, který má být sledován. V levé horní části jsou umístěny záložky s názvy Sequence a Real time. První z těchto záložek slouží k zobrazování snímků, které jsou načteny ze snímkové sekvence na externím úložišti. Záložka Real time umožňuje zobrazení snímků, které jsou získány z připojeného kamerového zařízení. V obou případech se objekt ke sledování vybere stejným způsobem. Výběr se provádí intuitivně za pomoci stisku levého tlačítka myší a tahu směrem dolů a doprava. Oblast sledování je označována zeleným obdélníkem a její ukončení se provede uvolněním stisknutého tlačítka myši. Chování v případě zvolené záložky Sequence, je intuitivní a bylo již částečně popsáno v bodu 1. V případě selektování záložky Real time, mohou nastat tři situace. V případě, že není rozpoznáno žádné kamerové zařízení, je uživatel informován patřičnou zprávou. V opačném případě je kamerový systém automaticky přepnut do režimu snímání a jeho obraz je možno sledovat v oblasti 5. Poslední situace může nastat v případě, kdy má uživatel připojeno více kamerových zařízení. V takovém případě je zobrazeno dialogové okno, vyzývající uživatele k výběru zařízení, které má být použito pro snímání. Snímání kamerového zařízení se automaticky vypne, pokud je opět vybrána první záložka.

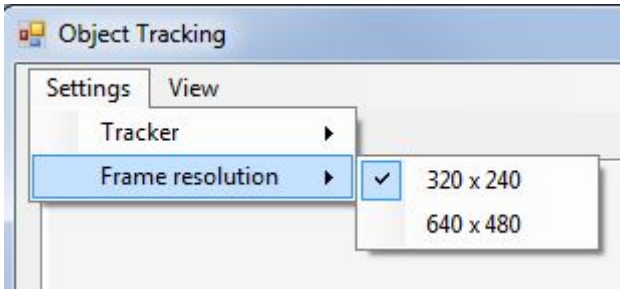
6. Poslední množinou ovládacích prvků na formuláři, je dvojice kontextových nabídek Settings a View. Obsah kontextové nabídky Settings obsahuje položky Tracker a Frame resolution. První z položek je zachycena na obrázku číslo 11. Položka Tracker umožňuje uživateli volbu sledovacího algoritmu, který bude aplikován. A dále možnost automatickou adaptaci sledovací v oblasti v případě, že je objekt pravděpodobně ztracen. Tuto možnost je vhodné použít pouze při volbě algoritmu CAMShift. U základního Mean shift algoritmu, je velikost sledovací oblasti fixní a tedy funkce nemůže být aplikována. U algoritmu CAMShift s dynamickým modelem cíle, má násilná adaptace spíše negativní dopad na průběh sledování. Poslední položka s názvem Calculate simple target size umožňuje volbu výpočtu sledovací oblasti pro algoritmy CAMShift s dynamickým modelem cíle a

CAMShift s modelem statickým. Pokud je položka aktivována, výška i šířka sledovací oblasti bude nastavena na stejnou hodnotu a to na vypočítanou šířku objektu podle vztahu (17), který je uveden v [7]. Pokud bude položka neaktivní, algoritmus bude vypočítávat šířku i výšku objektu podle vztahů (20) a (21).



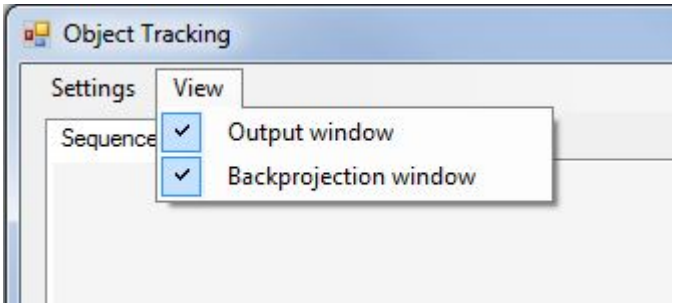
Obr. 11 Nabídka Settings - nastavení algoritmu.

Druhou položkou z nabídky Settings, která je zachycena na obrázku s číslem 12, je položka s názvem Frame resolution. Položka se týká sledování objektu v reálném čase a umožňuje potlačit rozlišení získávaného obrazu z kamery. Tato funkce je velice užitečná v případech, kdy je aplikace spuštěna na počítačích s menším výpočetním výkonem a obrazy z kamerového zařízení jsou získávány v rozlišení 640x480 nebo vyšším. V takovém případě může být analýza jednoho snímku několika násobně vyšší, obzvláště v případě užití kamerových systému s vysokou snímkovou frekvencí. Zmenšením obrazu na menší rozlišení, je možné analýzu obrazu výrazně urychlit.



Obr. 12 Nabídka Settings - volba rozlišení snímku.

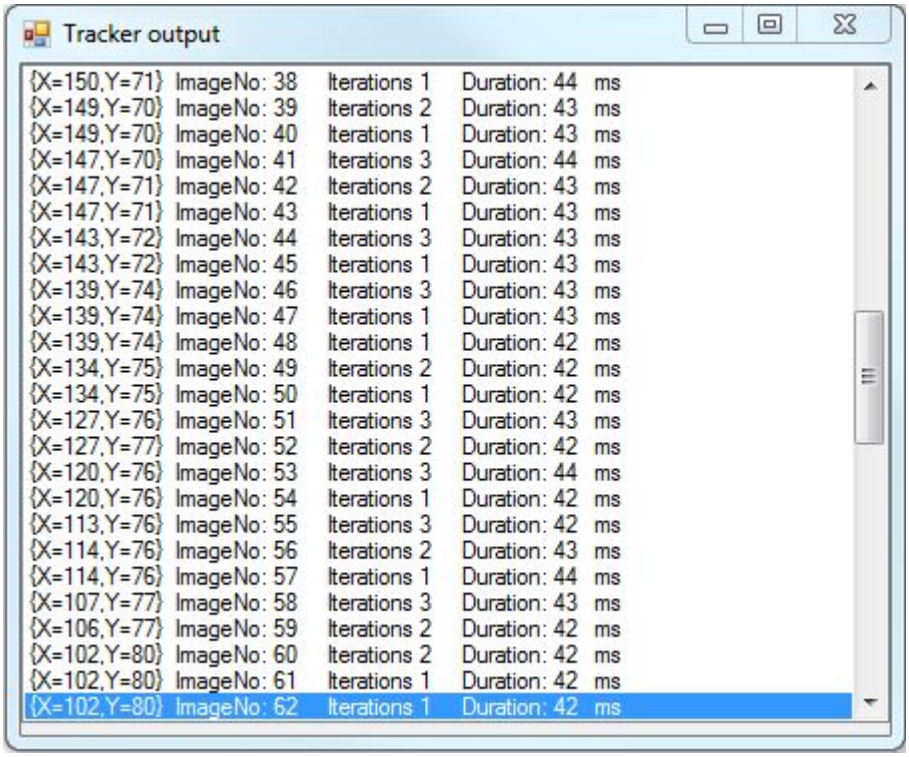
Posledním ovládacím prvkem zobrazeným v oblasti 6, je nabídka s názvem View, která je zobrazena na obrázku 13. Zde jsou obsaženy pouze dvě dvoustavové položky, umožňující zobrazování nebo skrývání dodatečných formulářů.



Obr. 13 Nabídka View - zobrazení infomativních formulářů.

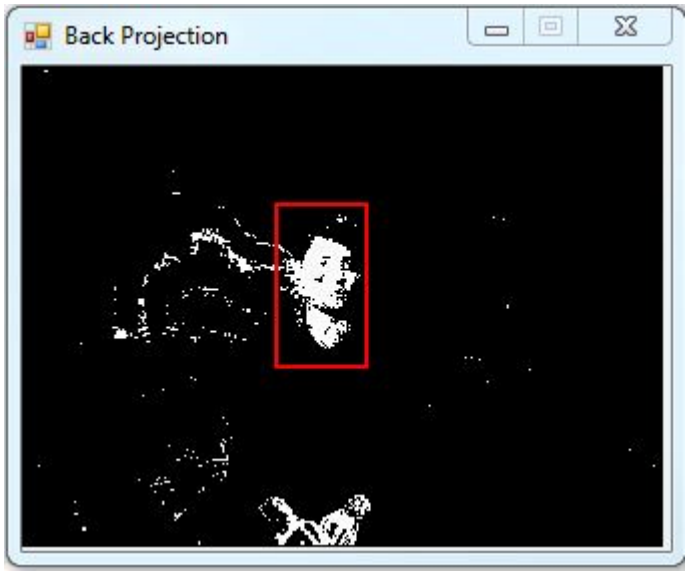
První položka s názvem Output window obsahuje seznam informací, které jsou aktualizovány

po analýze každého snímku. Okno je vyobrazeno na obrázku 14 a je možné ho kdykoliv skrýt nebo naopak zobrazit. Jednotlivé parametry jsou rozděleny do třech sloupců. První sloupec zleva, nese informaci o pozici sledovaného objektu v kartézském souřadnicovém systému. Hodnoty X a Y označují polohu středu sledovací oblasti. Druhý sloupec určuje pořadí analyzovaného snímku. Třetí sloupec nese informaci o počtu provedených iterací algoritmu, do okamžiku než byl cíl nalezen. Poslední sloupec označuje dobu trvání analýzy aktuálního snímku v milisekundách.



Obr. 14 Informace o průběhu sledování.

Poslední položkou nabídky View, je Backprojection window. Obraz zpětné projekce byl již v této práci dostatečně vysvětlen, ukázka je znázorněna na obrázku 15.



Obr. 15 Zpětná projekce.

8 TESTY A POROVNÁNÍ

V této části práce budou zobrazeny výstupy výše popsaných algoritmů a následně bude provedeno srovnání. Aby bylo možné porovnat jednotlivé algoritmy a rozhodnout, který z nich dosahuje lepších výsledků, bylo nutné vytvořit systém porovnávání. V této části budou uvedeny testy všech tří algoritmů, na třech snímkových sekvencích. Konkrétně se jedná o sledování tváře, sledování předmětu, jehož barva splývá s pozadím a v poslední sekvenci je sledován předmět, jehož barva je odlišná od pozadí, ale v průběhu sekvence dochází k částečnému nebo celkovému clonění cizím objektem.

Všechny implementované algoritmy implicitně vracejí následující údaje: polohu těžiště sledovaného objektu vztaženou k levému hornímu rohu snímku, údaj o počtu provedených iterací algoritmu, údaj o celkové časové náročnosti nalezení těžiště na jednotlivých snímcích, dále údaje o výšce a šířce sledovací oblasti a přesnou polohu všech krajních bodů sledovací oblasti. Tyto údaje odpovídají poloze těžiště sledovaného objektu v podání algoritmu, avšak aby bylo možné porovnat míru přesnosti algoritmu, je nutné zjistit přesnou polohu objektu na každém snímku sekvence. V tomto projektu bylo zjištění přesné polohy objektu na snímku dosaženo pomocí vnějšího pozorovatele. Na každém snímku sekvence, byl objekt manuálně označen. Manuálním označení není možné dosáhnout korektní přesnosti středu objektu, proto je nutné brát označení polohy cíle s určitou tolerancí.

Aby bylo možné adekvátně porovnat všechny tři algoritmy, bylo také nutné zajistit, aby byl objekt při každém testování vybrán stejně přesně. Tohoto požadavku bylo dosaženo výběrem cíle podle předaných parametrů, jenž se sestávají ze souřadnic levého horního rohu a pravého spodního rohu sledovací oblasti.

8.1 Diagnostický mód aplikace

Aby bylo možné získávat výše uvedené údaje, bylo nutné do aplikace zakomponovat mód diagnostiky, který je zcela nezávislý na aplikaci sledování objektu. Mód je integrován do aplikace a je mu vymezen samostatný projekt s názvem Diagnostic. V projektu jsou zahrnuty třídy, jenž plní funkci shromažďování údajů a jejich zápis na pevný disk v textové formě. Diagnostický mód je možné sepnout dvoustavovým stejnojmenným tlačítkem. V případě aktivace se zobrazí textové pole pro vkládání přesných souřadnic sledovací oblasti v následujícím formátu: x-ová, y-ová souřadnice horního levého rohu, následuje oddělení středníkem a zapíše se x-ová a y-ová souřadnice spodního pravého rohu. Zápis může vypadat následovně: „120,100;150,135“. Po zmáčknutí tlačítka s názvem Mark Target, je cíl sledování na snímku přesně označen. Pokud je po dokončení sledování zmáčknuto tlačítko Reload a zároveň je diagnostický mód stále aktivní, jsou všechny výstupy sledování uloženy do odpovídající složky na pevném disku. Vytvořený textový soubor obsahuje hlavičku, v které je vygenerováno datum a čas sledování, název sekvence, typ použitého algoritmu, typ a rozměry použitého histogramu a údaje o počáteční pozici sledovací oblasti. Dále již následují data vztažená ke každému snímku sekvence - vždy jeden na řádek. Typy jednotlivých uložených údajů jsou odděleny patřičnými nadpisy.

8.2 Výsledky provedených testů

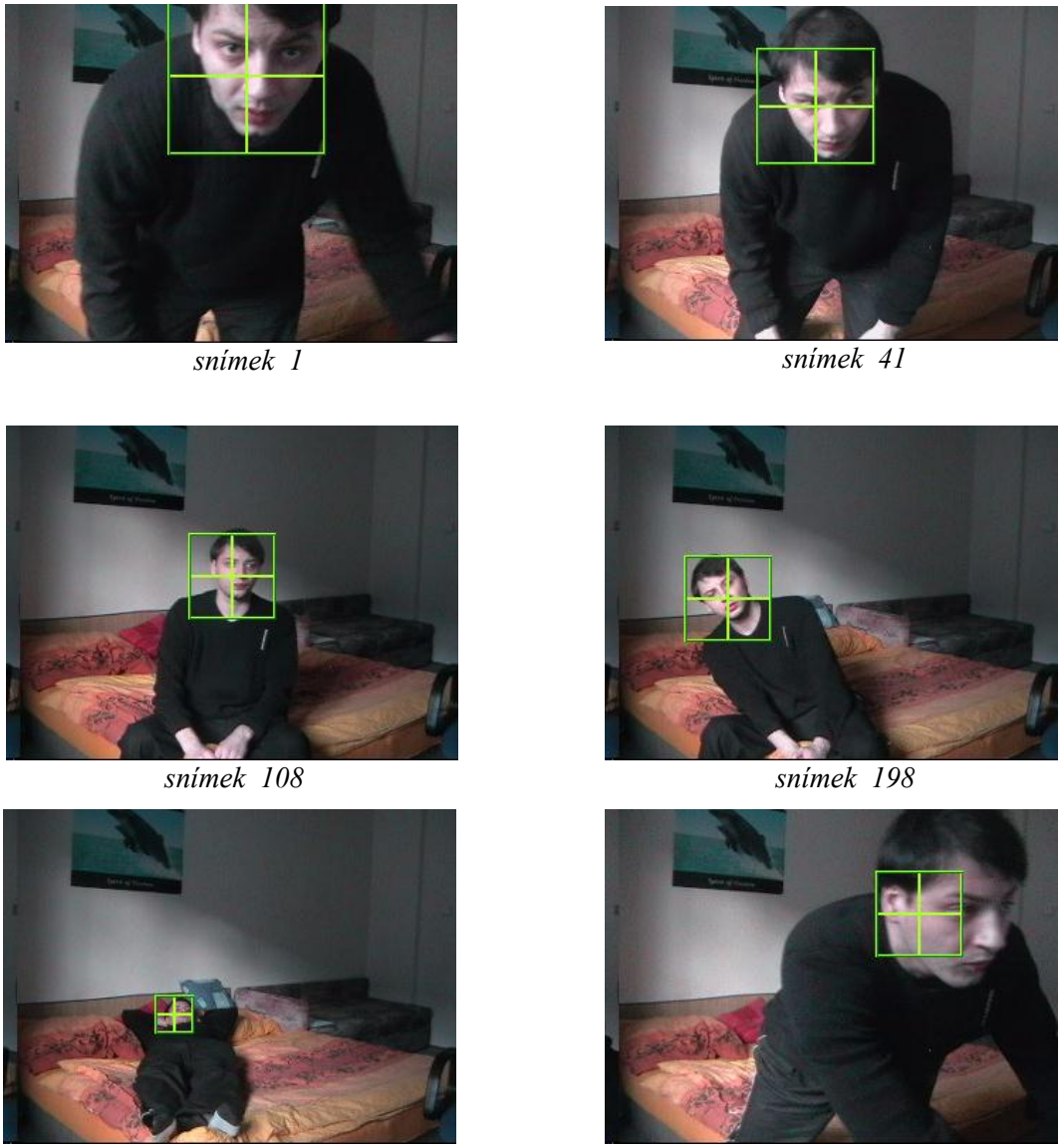
Nyní budou následovat údaje z provedených testů na výše zmíněných sekvencích. Údaje jsou reprezentovány formou grafů, na nichž je patrná úspěšnost implementovaných algoritmů, je zde také zobrazena závislost počtu provedených iterací na době trvání výpočtu.

Ve všech testovaných případech byly pro sledování použity histogramy typu KernelBased, s vy jímkou algoritmu CAMShift s dynamickým modelem cíle, který je schopen adekvátního sledování pouze v případě použití vyváženého histogramu. Po celou dobu sledování byl zvolen histogram s rozměry 12x12x12. Tyto rozměry byly získány experimentálně a představují kompromis, v podobě relativně přesného a z pohledu sestavení časově nenáročného, modelu sledovaného objektu. Všechny sekvence na nichž bylo testování prováděno, jsou dodávány společně s aplikaci, včetně externích

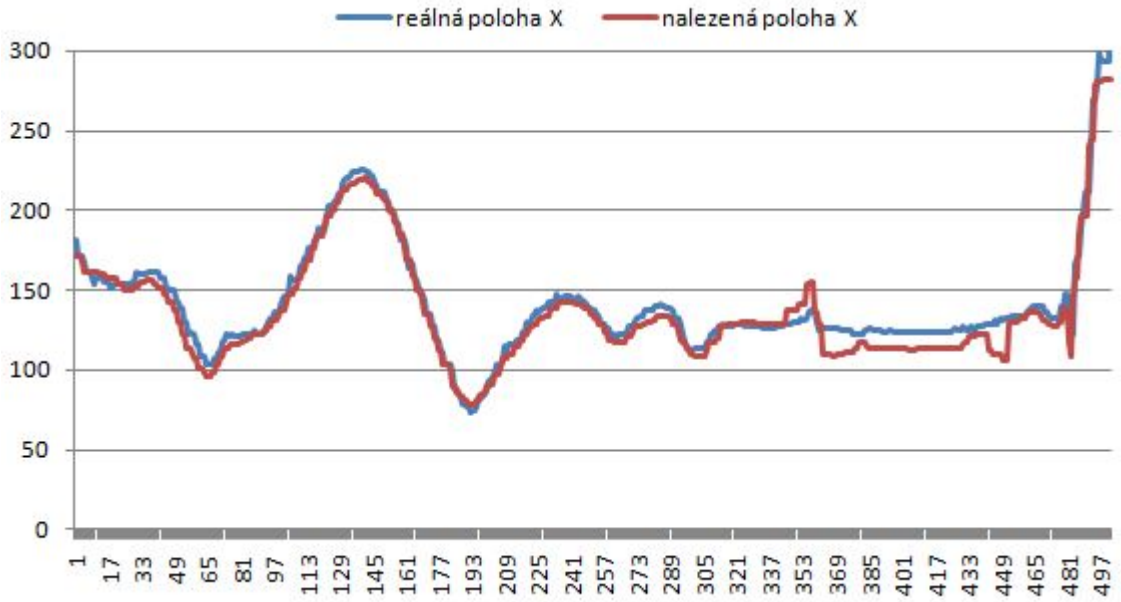
souborů, představující výsledky sledování a testování. V následující části budou vždy uvedeny výsledky všech tři algoritmů, jenž jsou vztaženy k jednotlivým souřadnicím X a Y. Nejprve bude zobrazen průběh pro hodnoty souřadnice X, následně pro hodnotu souřadnice Y. Uvedené testy byly provedeny na snímcích s rozlišením 320 x 240 pixelů. Průběhy jednotlivých testů je možné zopakovat zadáním vstupních souřadnic polohy sledovacích oblastí v diagnostickém módu aplikace. Tyto informace jsou společně s testovacími sekvencemi umístěny v přílohách tohoto projektu, kde jsou ke každé sekvenci přiloženy soubory obsahující výsledky jednotlivých algoritmů, inicializační údaje sledovacích oblastí a nastavení parametrů aplikace pro každou testovací sekvenci.

8.2.1 Testovací sekvence Face2

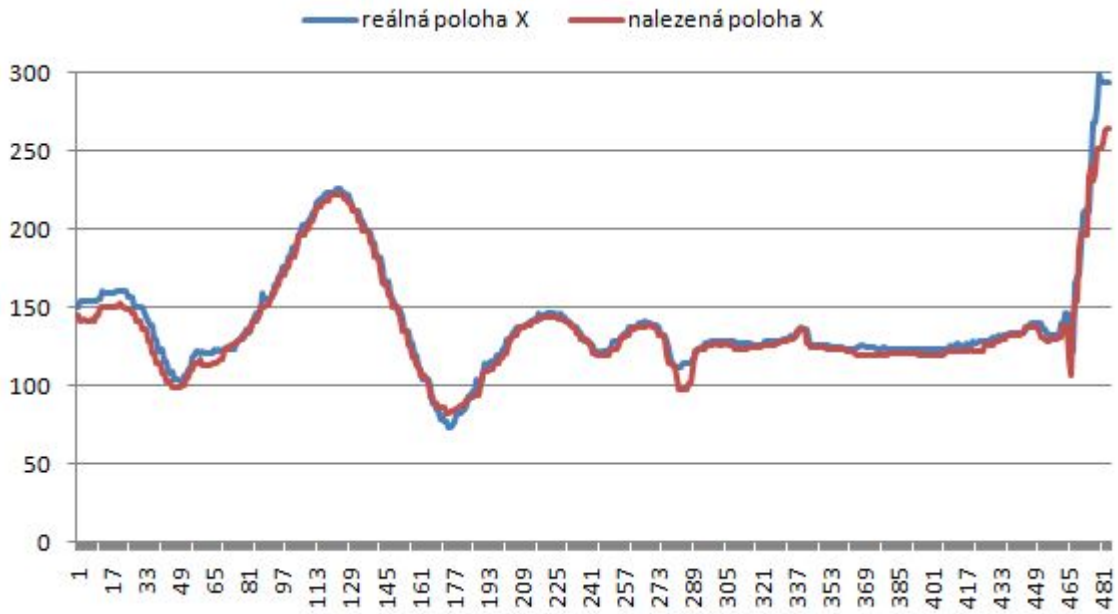
Předmětem sledování v této sekvenci byla tvář. Subjekt se pohybuje před kamerovým systémem ve vzdálenosti od třiceti centimetrů až po vzdálenost čtyř metrů. Cílem sledování bylo otestovat implementované algoritmy v případech, kdy se sledovaný objekt značně zmenšuje v závislosti na vzdálenosti od objektivu. Ukázka snímkové sekvence je znázorněna na obrázku 16.



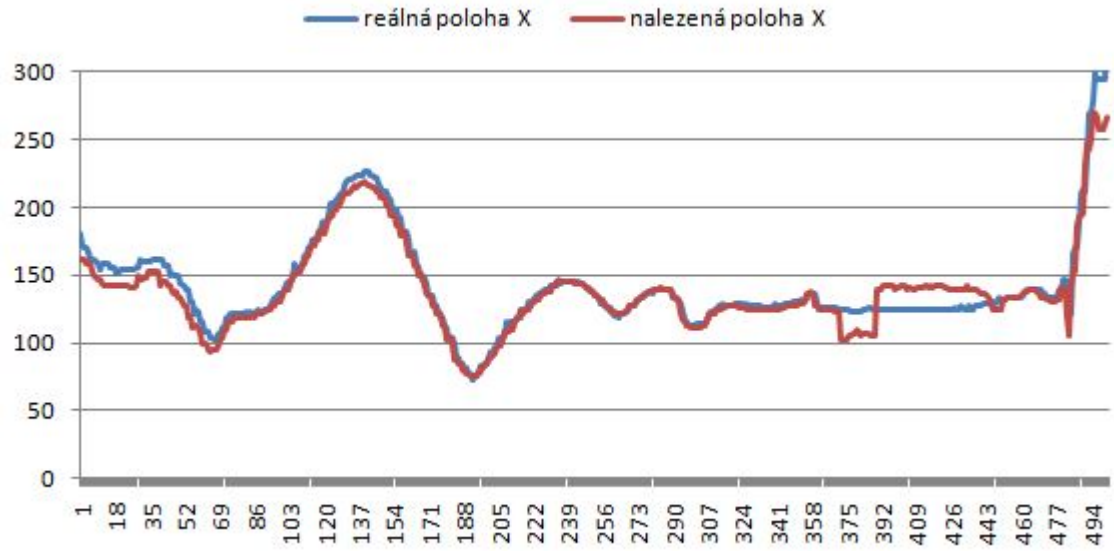
Obr. 16 Ukázka testovací sekvence Face2. Cíl je identifikován algoritmem CAMShift.



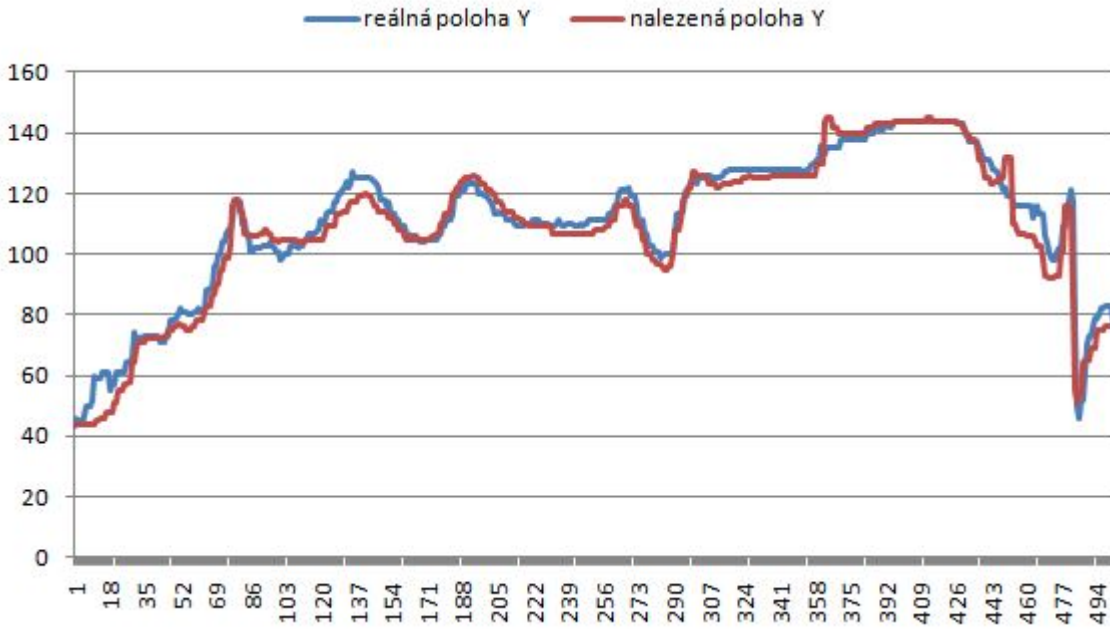
Obr. 17 Mean shift algoritmus - přesnost nalezené souřadnice X.



Obr. 18 CAMShift algoritmus - přesnost nalezené souřadnice X.



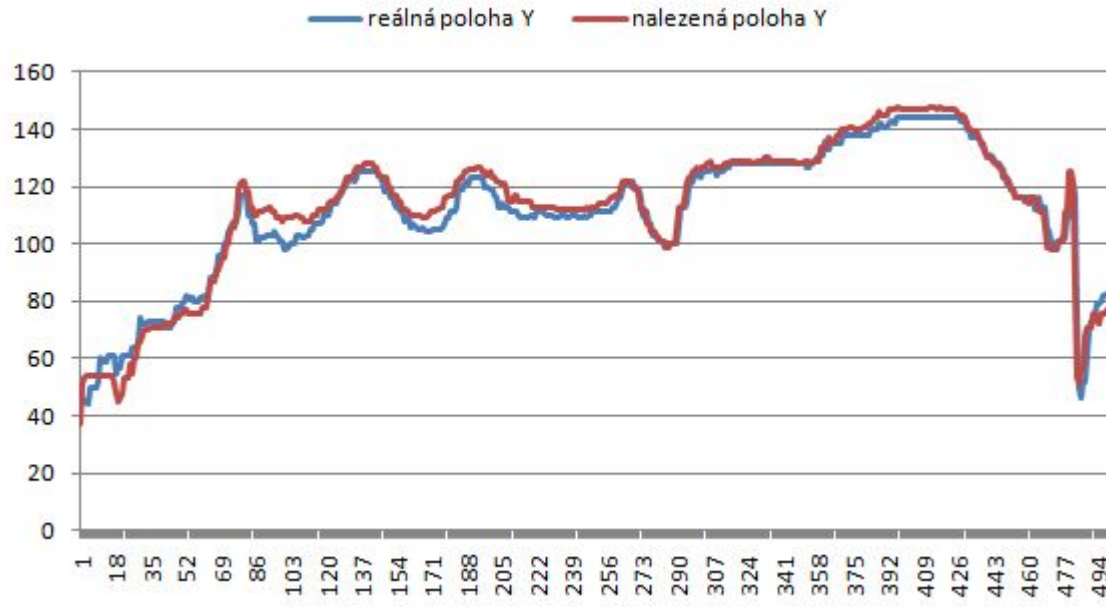
Obr. 19 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice X.



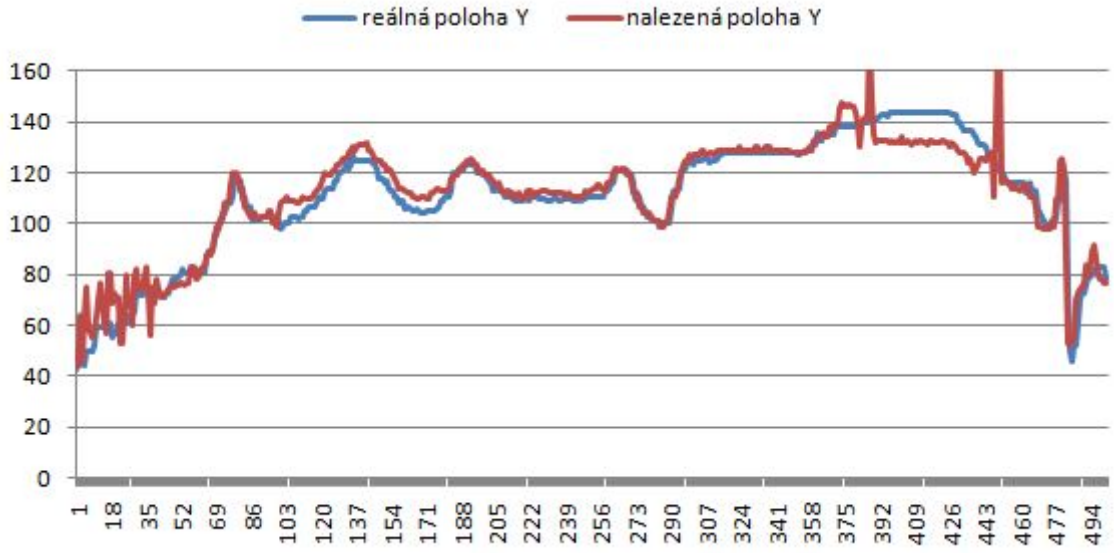
Obr. 20 Mean shift algoritmus - přesnost nalezené souřadnice Y.

Na grafech uvedených na obrázcích 17 až 19, je zachycen průběh skutečné pozice souřadnice X a souřadnice nalezená sledovacím algoritmem. Jak je patrné z průběhu grafů, všechny tři algoritmy opisují trajektorii skutečné souřadnice s vysokou přesností. Podle průběhu se kritickým místem sekvence jeví oblast počínaje snímkem 358 a končící snímkem 460. Ačkoli všechny testované algoritmy nejeví přílišné odchylky, v této oblasti nejlépe obstál algoritmus CAMShift, jehož průběh je zachycen na obrázku 18.Výsledky algoritmu Mean shift a CAMShift s dynamickým modelem cíle, by se daly považovat za stejně významné, jak je možné vypozorovat z průběhů grafů.

Grafy uvedené na obrázcích 20 až 22, představují průběh hodnoty souřadnice Y. V případě této souřadnice je možné pozorovat, že všechny testované algoritmy mají patrnou odchylku v první polovině grafu, přičemž průběh CAMShift algoritmu s dynamickým modelem cíle byl na hranici stability. Výsledky naznačují, že algoritmus se úspěšně adaptoval a jeho průběh se ve střední části grafů nijak výrazně neodlišuje od ostatních dvou algoritmů. Závěr průběhu grafu na obrázku 22 naznačuje, že algoritmus patrně v této fázi selhal, ale i přes značnou odchylku byl cíl znovu zachycen a úspěšně sledován do konce sekvence. Algoritmy Mean Shift a CAMShift vykazují obdobné

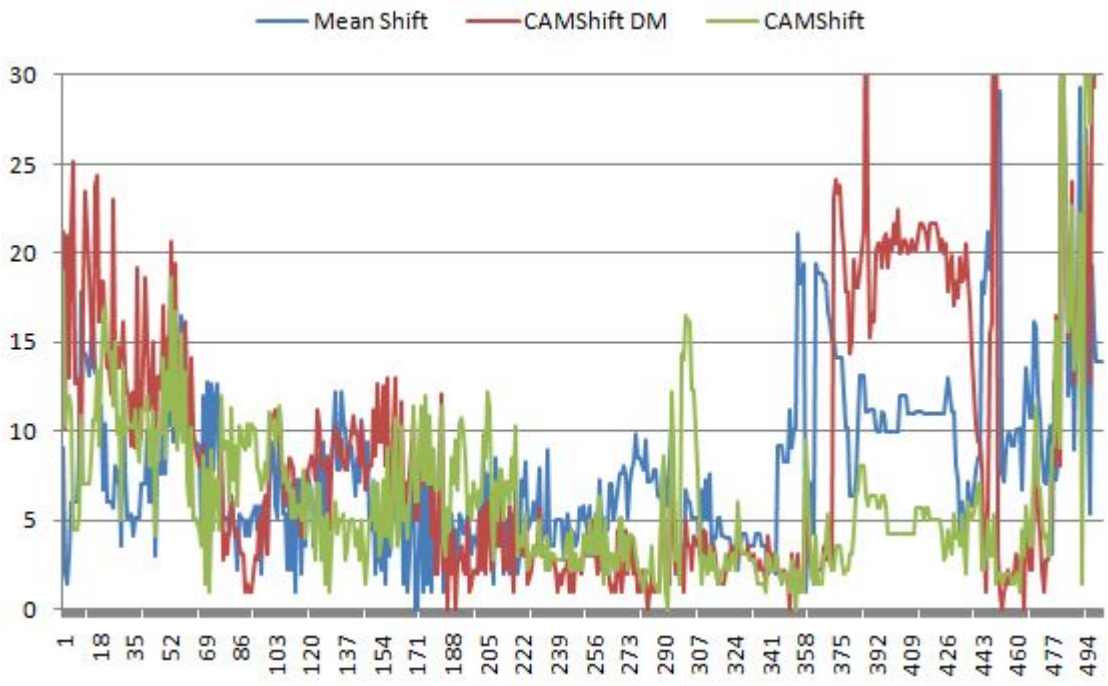


Obr. 21 CAMShift algoritmus - přesnost nalezené souřadnice Y.



Obr. 22 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice Y.

výsledky, přičemž dle průběhu grafu, algoritmus CAMShift vykazuje přesnější hodnoty, především v úvodu a závěru sekvence.

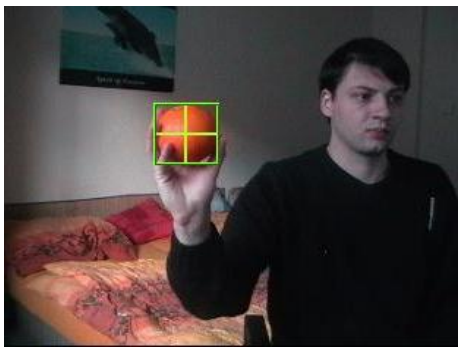


Obr. 23 Sekvence Face 2 - vzdálenost nalezené polohy od skutečné polohy objektu.

Na obrázku s číslem 23, je znázorněna vzdálenost nalezené polohy od skutečné polohy objektu v podání všech testovaných algoritmů. Y-ová osa zde představuje vzdálenost od skutečné polohy objektu v pixelech. Na průběhu grafu je pozorovatelné, že algoritmy CAMShift a Mean shift označovaly sledovaný objekt s nejmenší odchylkou. Všechny algoritmy jeví značnou odchylku v rozmezí mezi snímky 350 až 450, v této fázi sekvence byl objekt vzdálen nejdále od objektivu, jak je možné odvodit z ukázky testovací sekvence na obrázku 16. V situaci kdy byl objekt vzdálen od objektivu, uspěl s nejvyššími výsledky algoritmus CAMShift.

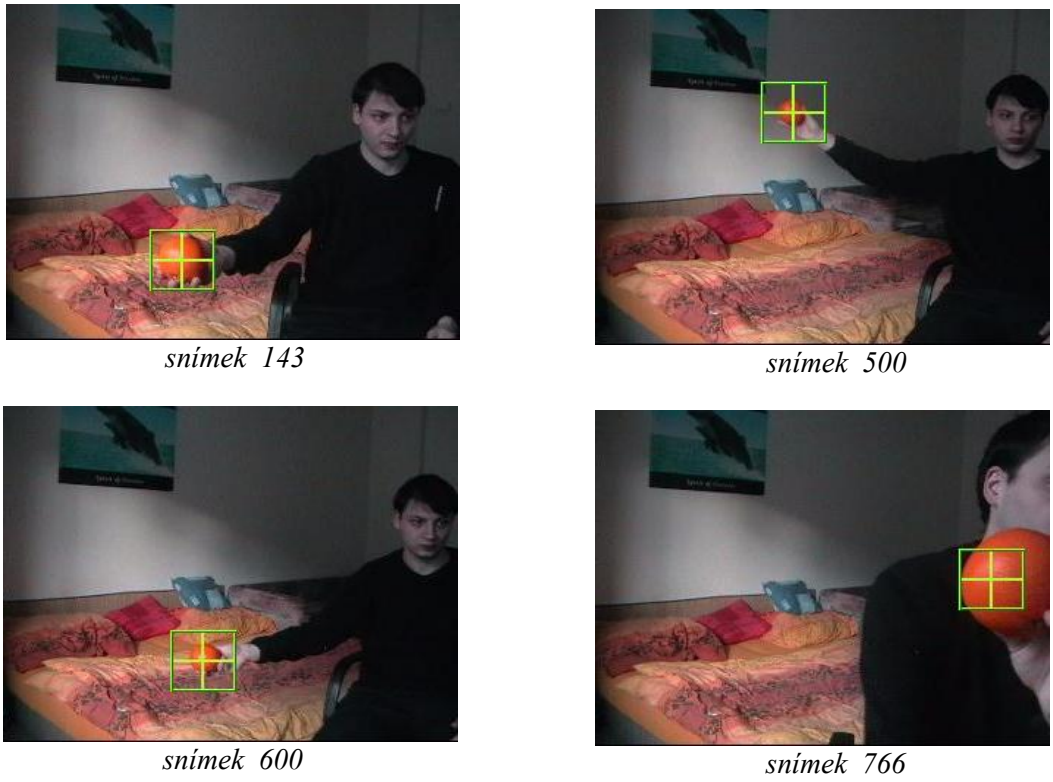
8.2.2 Testovací sekvence Orange

Předmětem této sekvence bylo otestovat chování algoritmů, při sledování objektu malých rozměrů, který při pohybu přechází mezi dvěma různorodými pozadími, přičemž jedno z nich je podobného barevného odstínu jako sledovaný objekt. Ukázka snímkové sekvence Orange je znázorněna na obrázku 24.

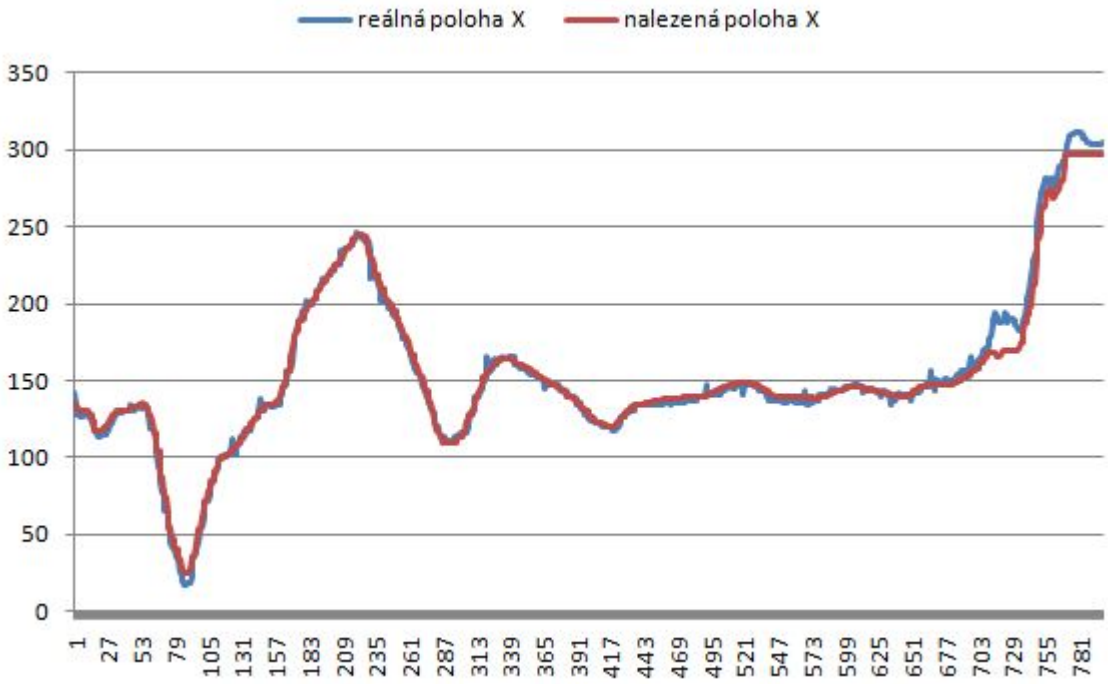


snímek 1

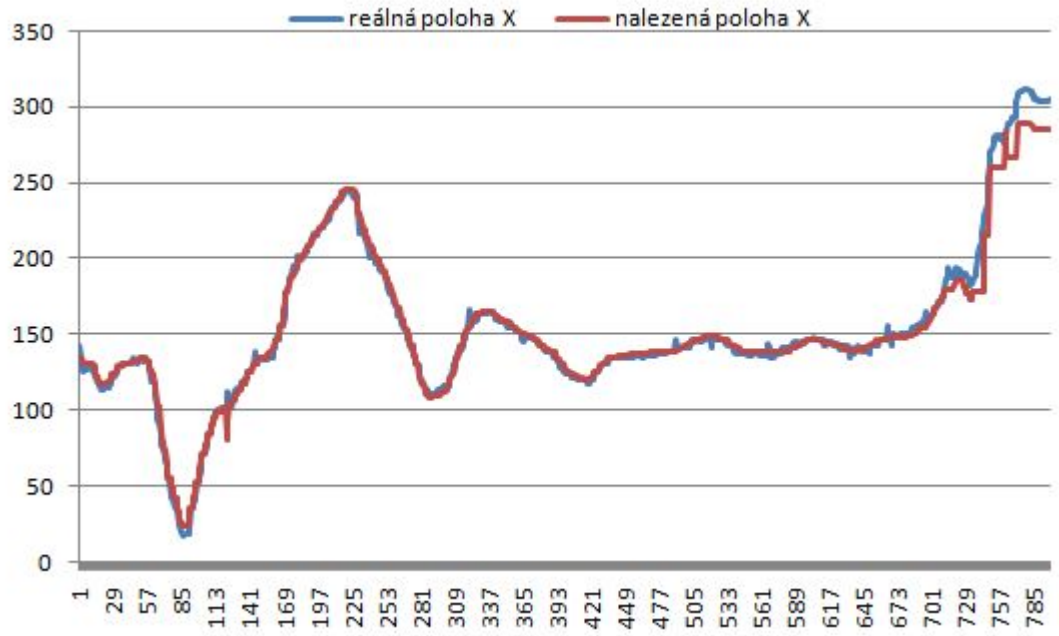
snímek 83



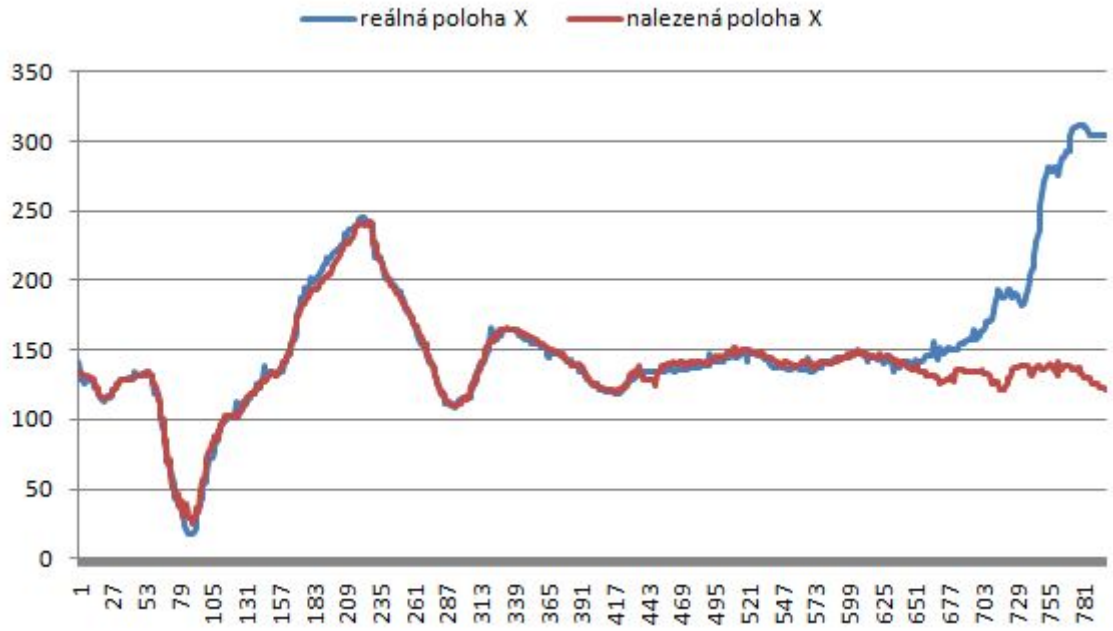
Obr. 24 Ukázka testovací sekvence Orange. Cíl je identifikován algoritmem Mean shift.



Obr. 25 Mean shift algoritmus - přesnost nalezené souřadnice X.

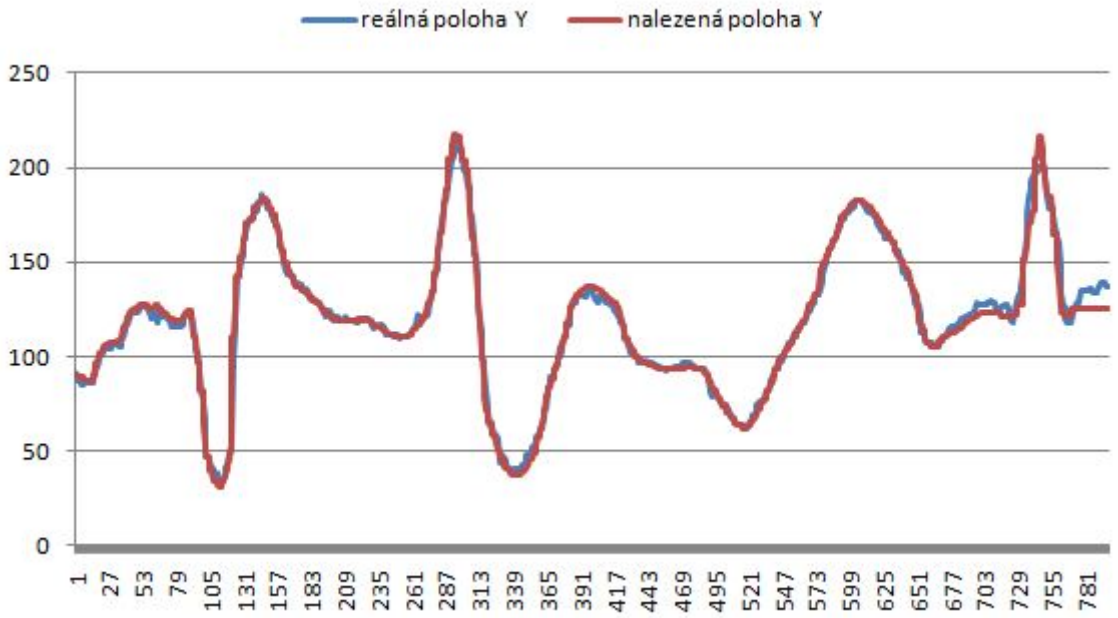


Obr. 26 CAMShift algoritmus - přesnost nalezené souřadnice X.

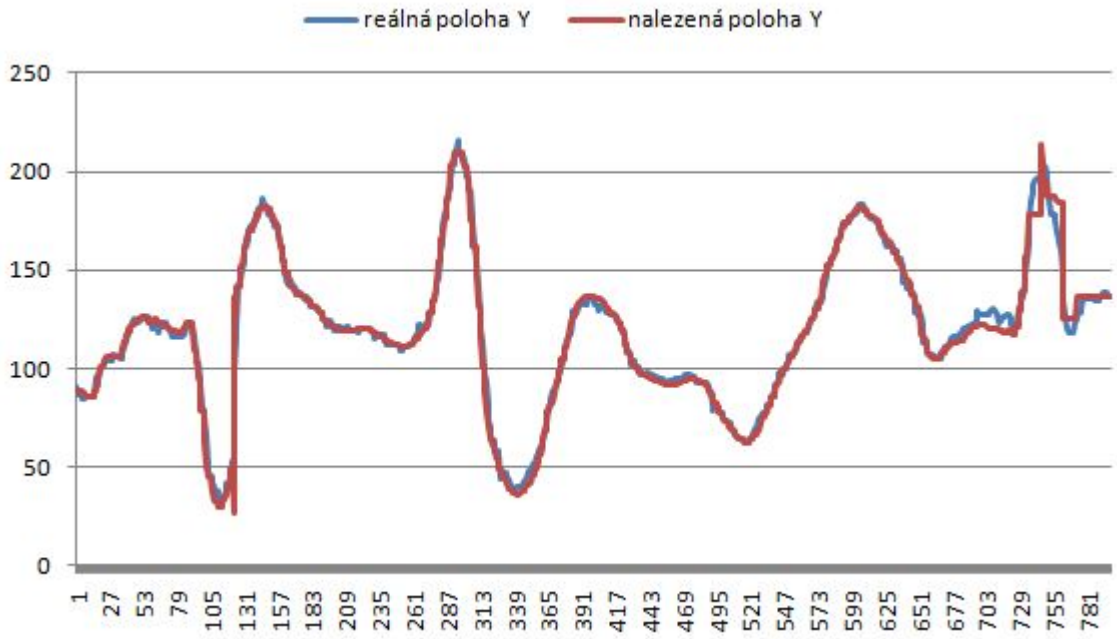


Obr. 27 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice X.

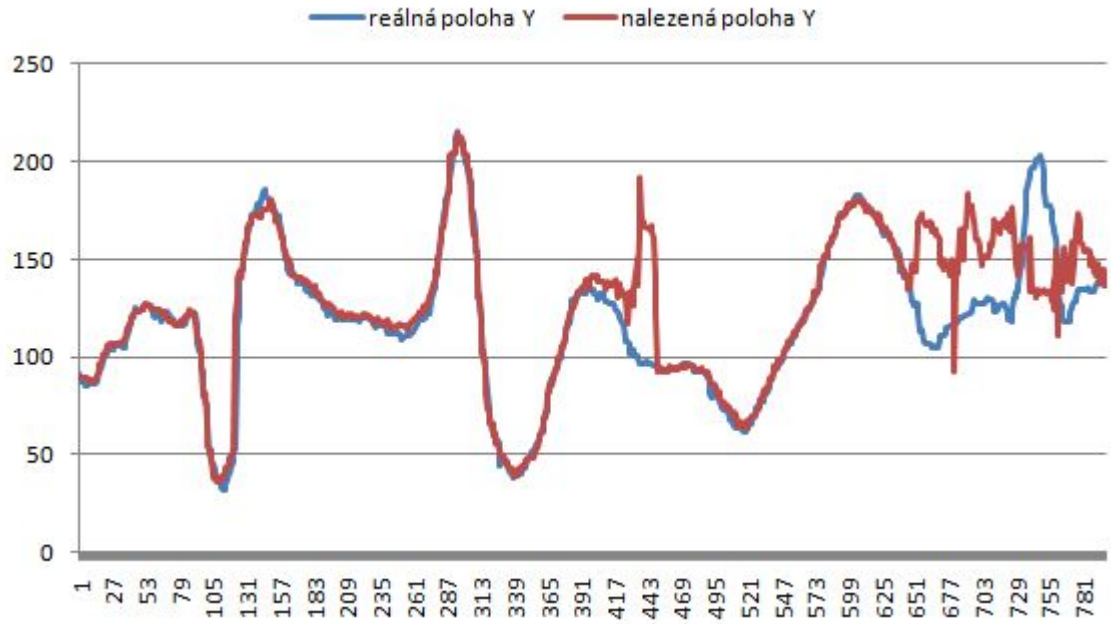
Grafy zobrazené na obrázcích 25 až 27, představují stejně jako v předchozím případě průběh hodnot souřadnice X. Průběh hodnot Mean shift algoritmu na obrázku číslo 25 a CAMShift algoritmu na obrázku 26, jsou téměř identické. Mean shift algortimus lépe zvládl situaci v závěru sekvence, CAMShift byl úspěšnější v situací v rozmezí snímků 700 až 729. Na obrázku 27 je zachycen průběh CAMShift algoritmu s dynamickým modelem cíle. Je patrné, že algoritmus dosahoval do tří čtvrtin sekvence obdobných výsledků jako předchozí dva zmíněné algoritmy. V poslední čtvrtině sekvence začíná algoritmus selhávat.



Obr. 28 Mean shift algoritmus - přesnost nalezené souřadnice Y.

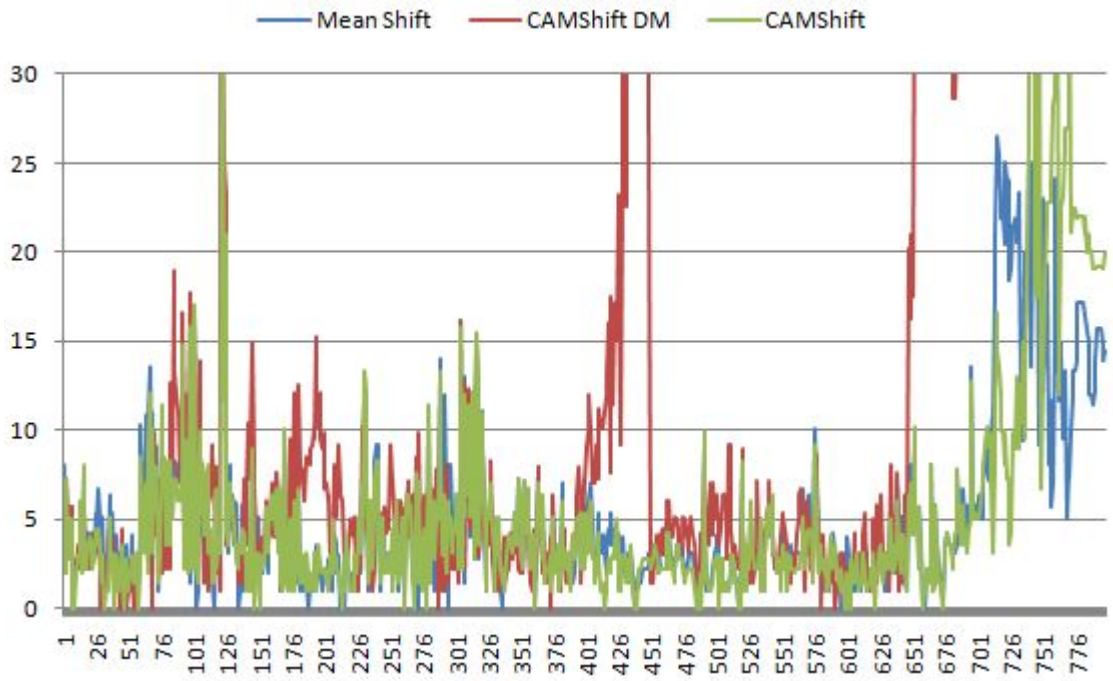


Obr. 29 CAMShift algoritmus - přesnost nalezené soudnice Y.



Obr. 30 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice Y.

Obrázky 28 až 30 znázorňují průběh hodnoty souřadnice Y. První dva algoritmy se průběhem opět téměř shodují. Za povšimnutí stojí, že obtížnější situaci v okolí snímků 703 až 755 v lokalizaci souřadnice Y, zvládá lépe Mean shift algoritmus oproti předchozímu případu, kde vynikal CAMShift.

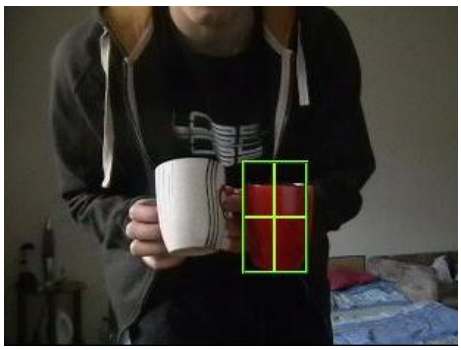


Obr. 31 Sekvence Orange - vzdálenost nalezené polohy od skutečné polohy objektu.

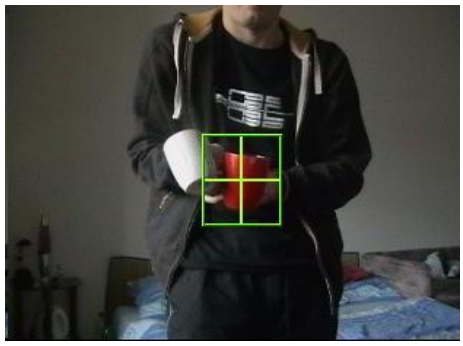
Poslední z grafů zastupuje CAMShift algoritmus s dynamickým modelem cíle a jeho průběh reflektuje výsledky dosažené v lokalizaci X-ové souřadnice v předchozím srovnání. Mimo jiné je zde zachyceno selhání algoritmu v okolí snímku 417 až 433, přičemž z pohledu X-ového průběhu byl tento úsek vyhodnocen správně. V závěru sledování je opět patrné selhání algoritmu, ačkoli odchylka souřadnic Y je tentokrát podstatně nižší. Vzdálenosti nalezené polohy od skutečné polohy objektu pro testované algoritmy jsou zachyceny na obrázku 31.

8.2.3 Testovací sekvence Cup

Předmětem poslední testovací sekvence, bylo porovnání sledovacích algoritmů v situacích, kdy je sledovaný objekt částečně nebo zcela cloněn cizím pohybujícím se předmětem v jeho blízkosti. Tato sekvence je reprezentována téměř tisíci snímky a patří k nejdéle trvajícím testovaným sekvencím.



snímek 1

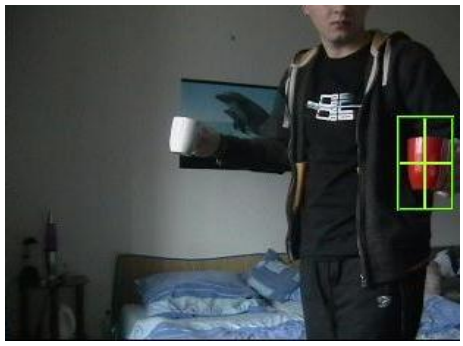


snímek 65

snímek 463

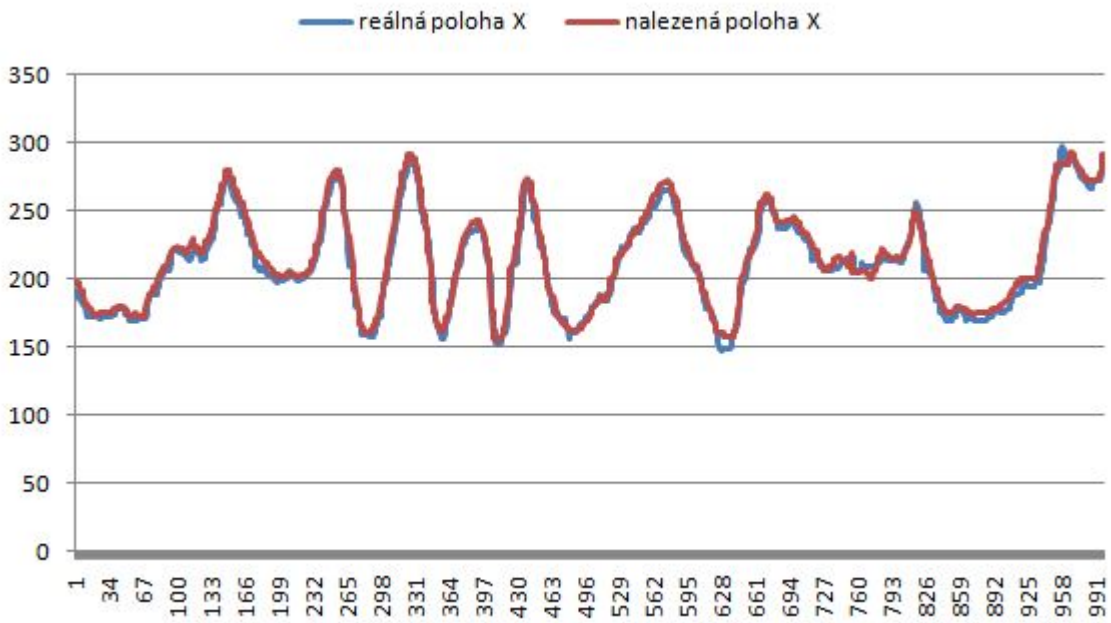
snímek 754

snímek 900

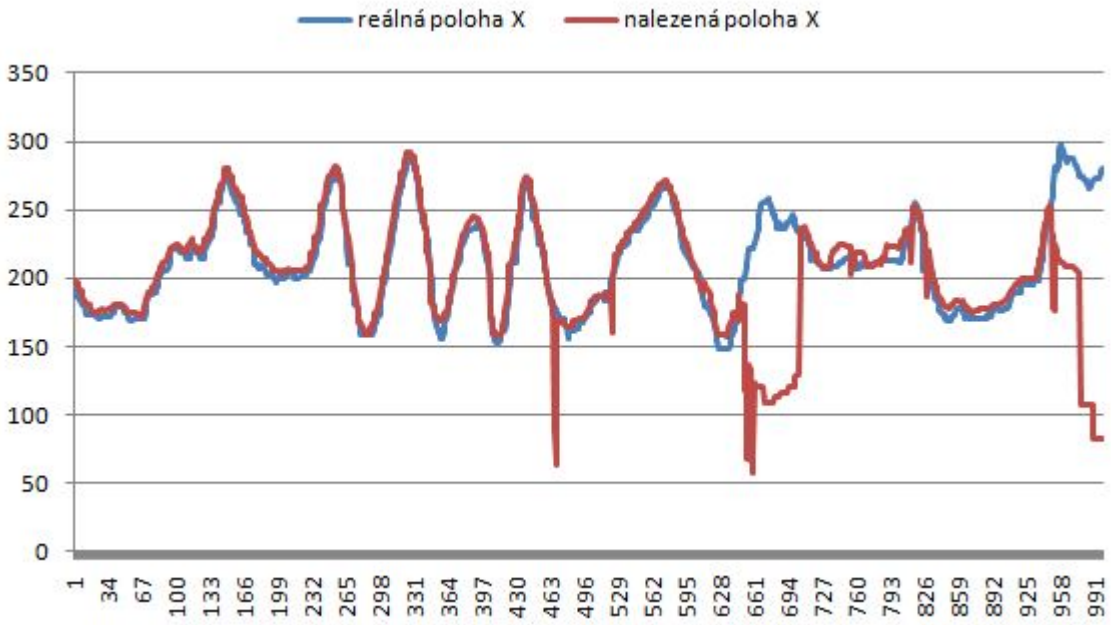


snímek 955

Obr. 32 Ukázka testovací sekvence Cup. Cíl je identifikován algoritmem CAMshift s dynamickým modelem cíle.

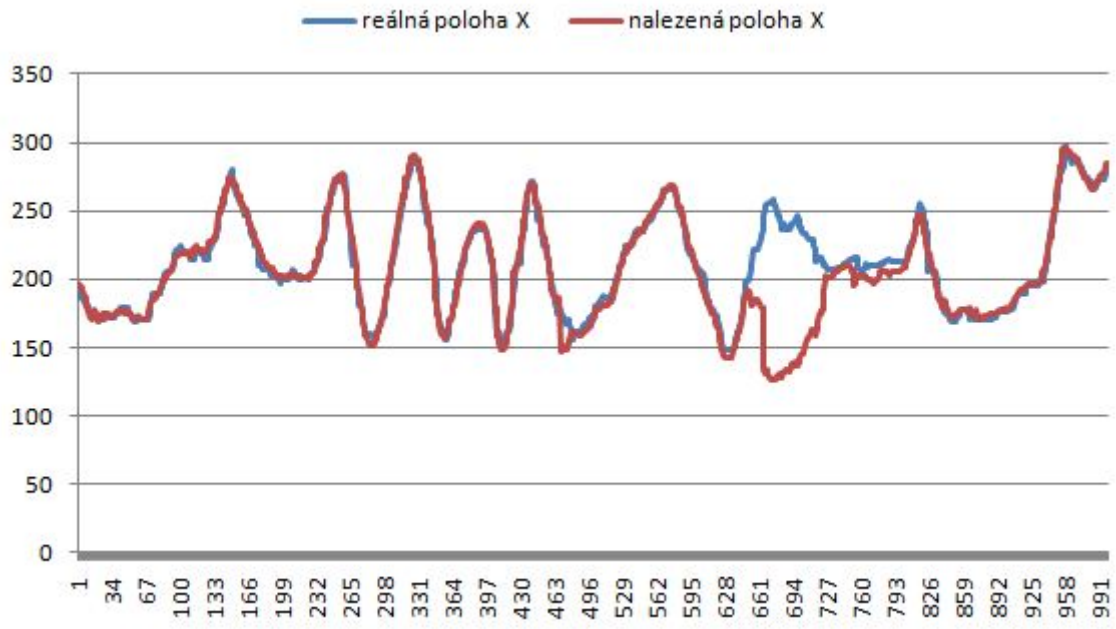


Obr. 33 Mean shift algoritmus - přesnost nalezené souřadnice X.

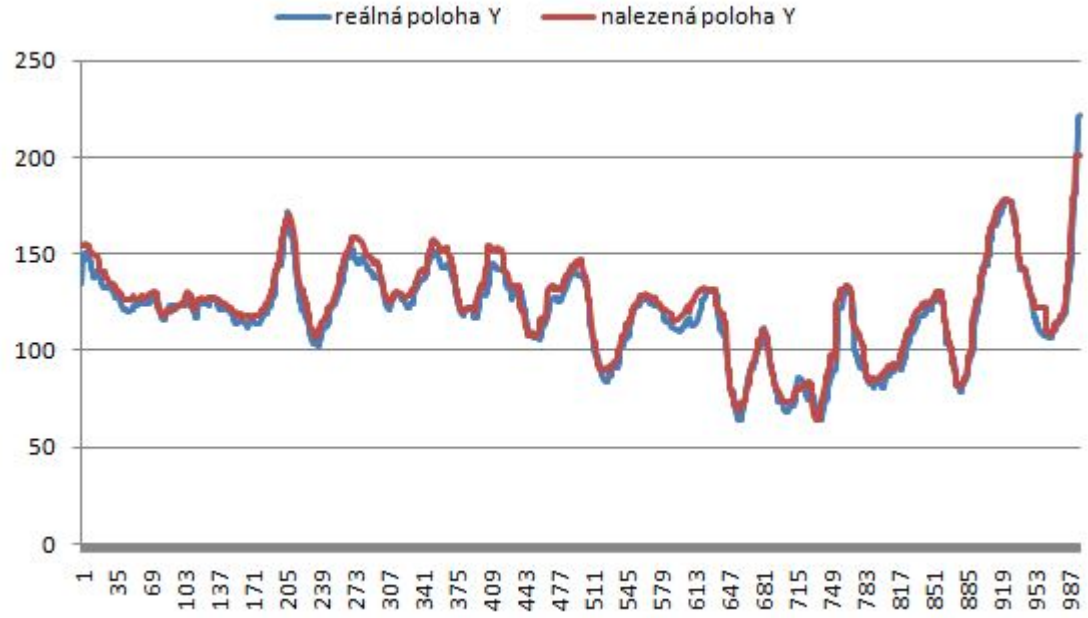


Obr. 34 CAMShift algoritmus - přesnost nalezené soudnice X.

Na obrázcích 33 až 35 jsou uvedeny průběhy hodnot souřadnice X. Algoritmus Mean shift v této sekvenci dosahuje nejpřesnějších výsledků. Algoritmus CAMShift selhával od druhé poloviny sekvence, jak je patrné z obrázku s pořadím 34.

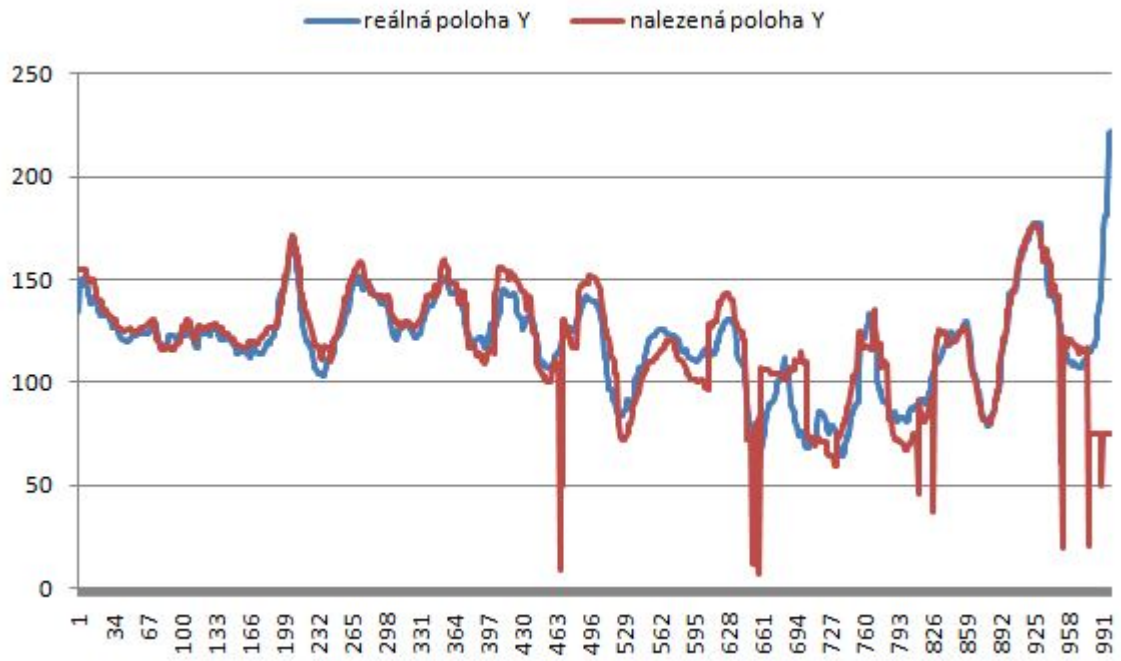


Obr. 35 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice X.

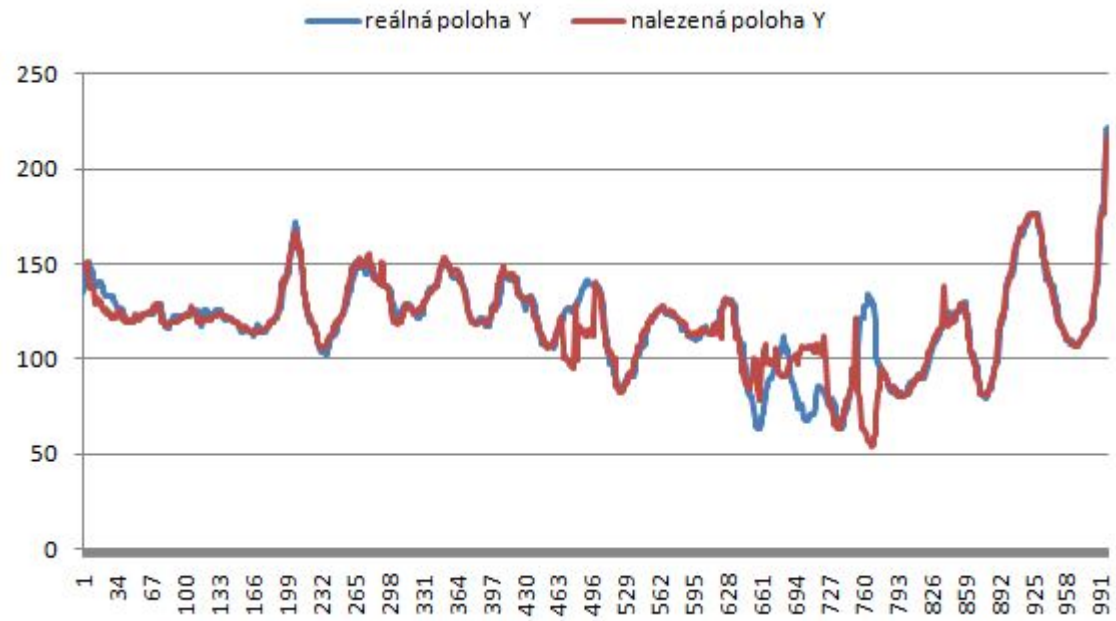


Obr. 36 Mean shift algoritmus - přesnost nalezené souřadnice Y.

Algoritmus CAMShift s dynamickým modelem cíle, byl při nasazení na tuto sekvenci relativně přesný, s vyjímkou oblasti v intervalu snímků s pořadím 628 až 727. Zde průběh identifikuje dočasnou ztrátu sledovaného objektu. V případě porovnání hodnot Y-ových souřadnic je patrná přesnost sledování algoritmu Mean shift, na obrázku 36. Z průběhu který patří algoritmu CAMShift, je opět patrný neúspěch algoritmu v oblasti snímků s pořadím 463, 661 a 958. Je zde pozorovatelná značná odchylka Y-ové souřadnice, kdy se téměř blíží k hodnotě 0, která reprezentuje horní hranici obrazu. Odchylka je značná, a to až o více než 70 pixelů. V tomto případě je zřejmé, že algoritmus sledovaný objekt ztratil.

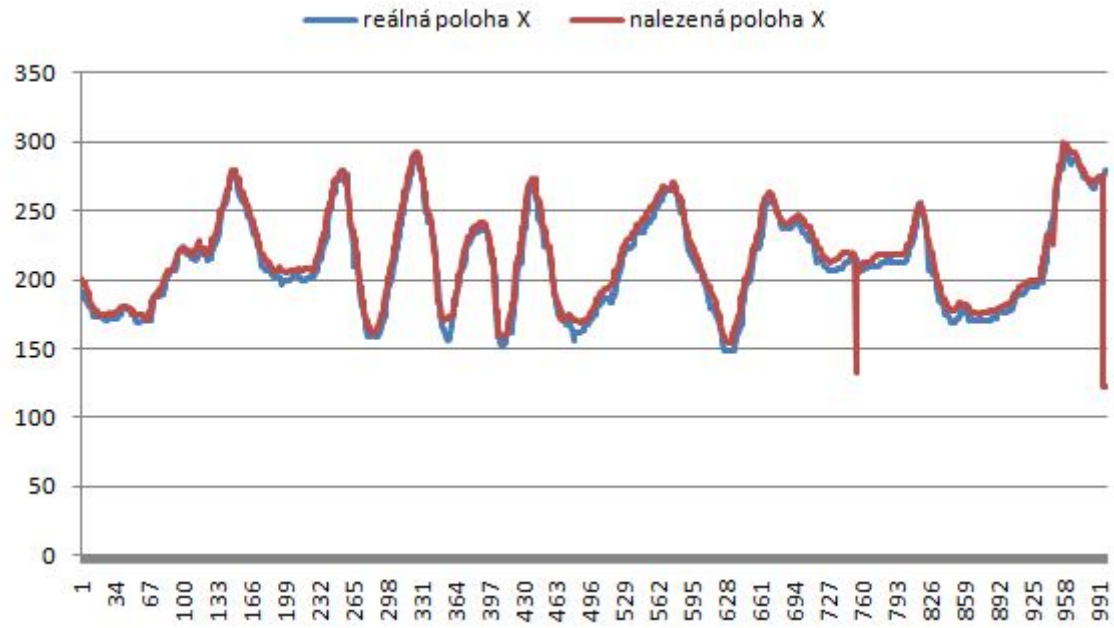


Obr. 37 CAMShift algoritmus - přesnost nalezené soudnice Y.

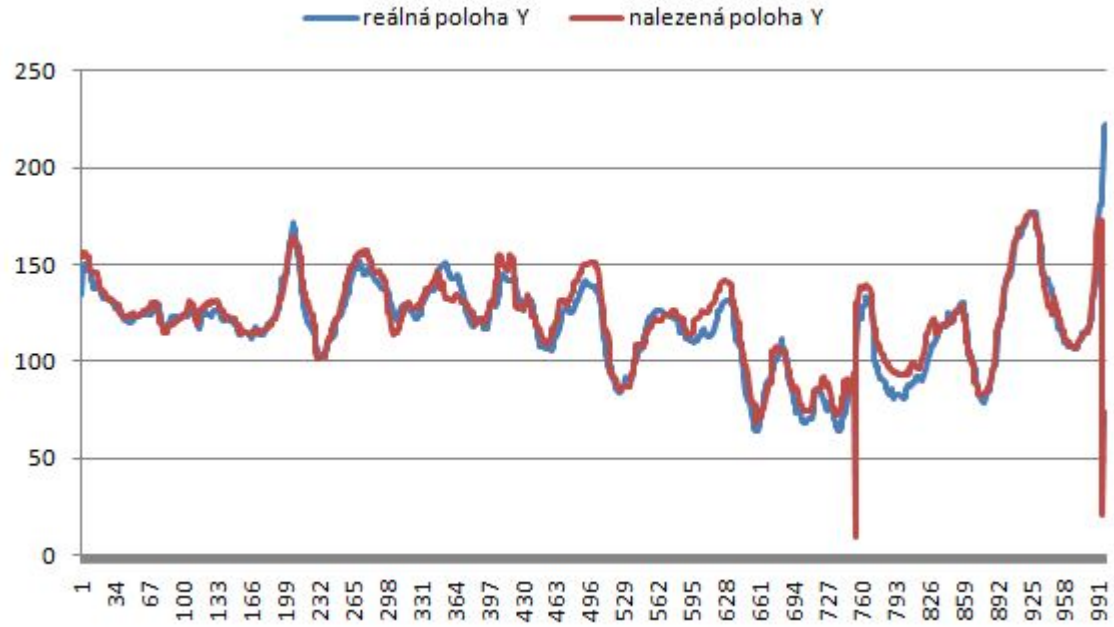


Obr. 38 CAMShift algoritmus s dynamickým modelem cíle - přesnost nalezené souřadnice. Y.

V případě algoritmu CAMShift s dynamickým modelem cíle, výsledky reflektují průběh hodnot souřadnice X. Je zde patrná odchylka v intervalu snímků, jako v případě průběhu hodnot X-ových souřadnic. Algoritmus byl však schopen sledovaný objekt opětovně zachytit a úspěšně sledovat do konce sekvence.



Obr. 39 CAMShift algoritmus - přesnost nalezené soudnice X, při použití nevyváženého histogramu.

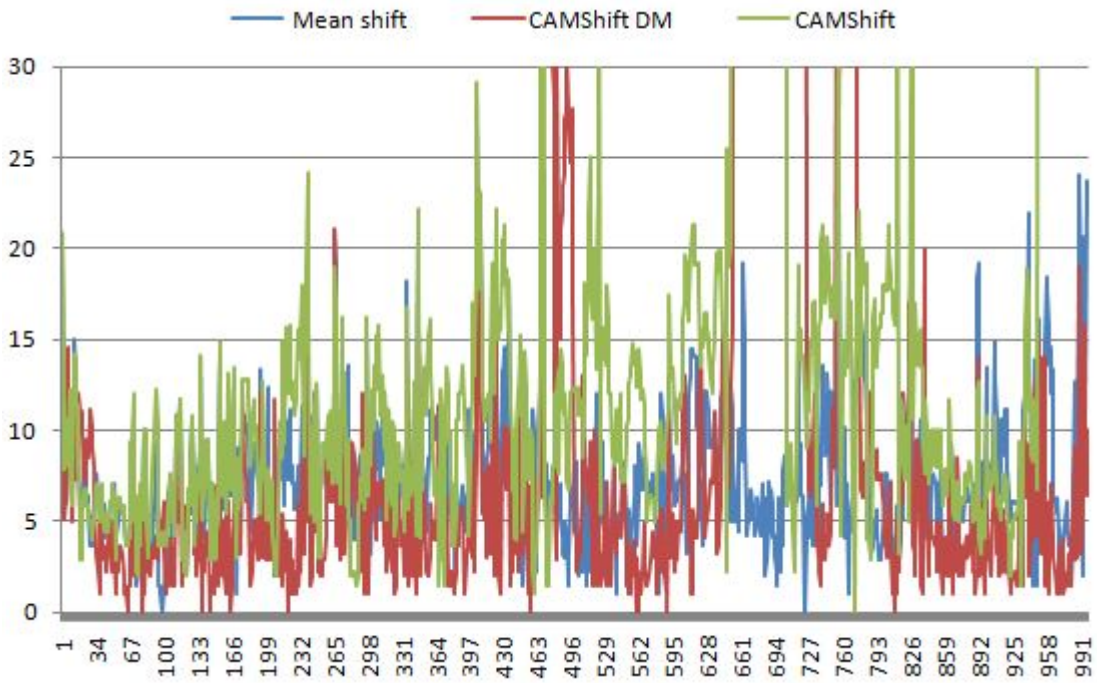


Obr. 40 CAMShift algoritmus - přesnost nalezené soudnice Y, při použití nevyváženého histogramu.

V této testovací sekvenci nastal poprvé případ, kdy algoritmus CAMShift selhal. Toto selhání bylo způsobeno výběrem cíle. Jak již bylo zmíněno v úvodu, všechny testy s vyjímkou algoritmu CAMShift s dynamickým modelem cíle, byly provedeny s histogramem typu KernelBased o rozměrech 12x12x12. Jak popisuje kapitola 4, výhoda aplikace jádra při sestavování histogramu, spočívá ve vyfiltrování nežádoucího vlivu pozadí objektu, které jej většinou při výběru obklopuje. Problém v tomto případě paradoxně způsobil relativně přesný výběr cíle. Při aplikaci jádra tedy došlo k eliminaci barevných hodnot po stranách sledovaného objektu, které v tomto případě nepatřily

pozadí, nýbrž sledovanému objektu a histogram cíle byl značně nekompletní. Adaptivní velikost sledovací oblasti způsobila, že nebyl označen celý objekt, nýbrž pouze část objektu, která nebyla jádrem znehodnocena. V takovém případě, se sledovací oblast nemohla adaptovat na celý objekt z důvodu, že všechny barvy cíle „neznala“. Důvodem proč Mean shift algoritmus ve stejné situaci obstál, je paradoxně jeho neměnící se velikost sledovací oblasti. Z tohoto důvodu byl algoritmus CAMShift otestován opakovaně s nevyváženým typem histogramu a zcela přesným výběrem cíle, kdy žádná část sledovací oblasti nezachycovala pozadí objektu. V tomto případě nevyvážený histogram způsobí, že model objektu bude obsahovat veškeré barevné hodnoty pixelů, jenž daný objekt obsahuje. Značný nárůst přesnosti při přesném výběru objektu a nefiltrovaném histogramu, je zřejmý z průběhů grafů na obrázcích 39 a 40. V průběhu obou souřadnic, je znatelná odchylka v okolí snímku 760, avšak algoritmus sledovaný objekt neztratil.

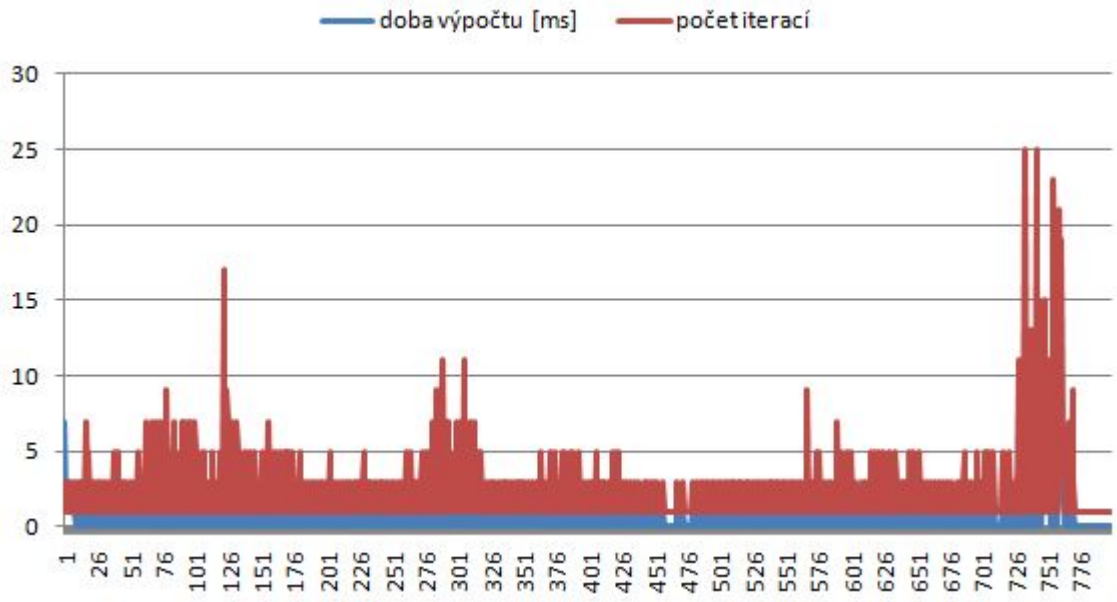
Jednotlivé odchylky vzdálenosti nalezené polohy od skutečné polohy objektu jsou znázorněny na obrázku s číslem 41. Ve vygenerovaném grafu je zachyceno původní sledování algoritmu CAMShift, bez výše popsané korekce výběru objektu. Z grafu jsou tedy patrné značné odchylky od skutečné polohy objektu v případě použití algoritmu CAMShift.



Obr. 41 Sekvence Cup - vzdálenost nalezené polohy od skutečné polohy objektu.

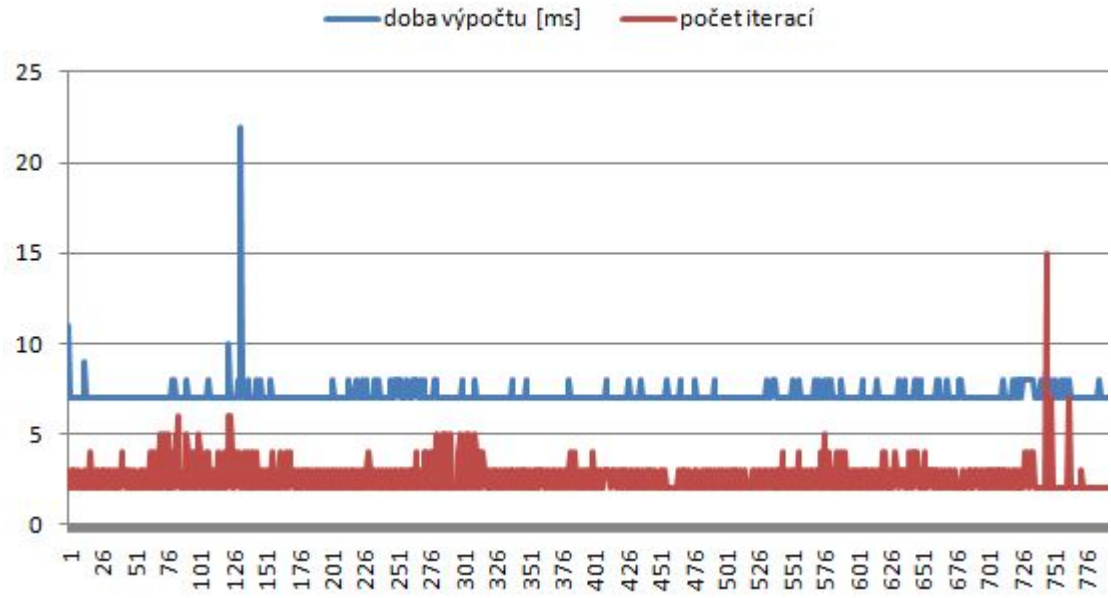
8.2.4 Porovnání výpočetní rychlosti

Předchozí část kapitoly byla věnována porovnání všech tří algoritmu z pohledu přesnosti sledování. V této části budou uvedeny výsledky testů z hlediska časové náročnosti a počtu provedených iterací, nutných k nalezení středu cíle na každém cíle. Z důvodu, že se algoritmy od sebe obecně v délce výpočtu značně liší nezávisle na testované sekvenci, postačí když bude uvedeno porovnání pro testovací sekvenci Orange. Porovnání vztažená k ostatním sekvencím, jsou uvedena v přílohách.

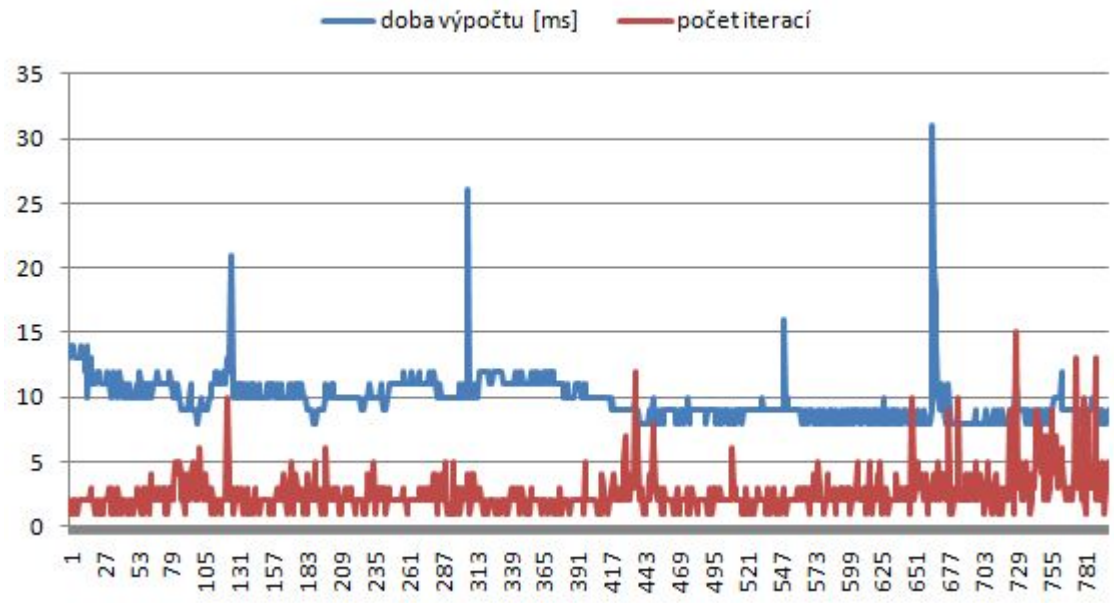


Obr. 42 Mean shift algoritmus závislost - doby výpočtu na provedených iteracích.

Testovací sekvence Orange byla pro tento typ porovnání vybrána záměrně z důvodu, že všechny uvedené algoritmy měly v závěru sekvence problémy se sledováním cíle. Doba výpočtu je uvedena v milisekundách a maximální počet iterací je aplikací nastaven na fixní hranici 25 iterací na snímek. Při srovnání rychlostí, které jsou zobrazeny na obrázcích 42 až 44, je zřejmé, že nejvyšších výsledků dosahuje algoritmus MeanShift, jehož doba výpočtu se po většinu sledování udržuje pod hranicí pěti milisekund na snímek.



Obr. 43 CAMShift algoritmus - závislost doby výpočtu na provedených iteracích.



Obr. 44 CAMShift algoritmus s dynamickým modelem cíle - závislost doby výpočtu na provedených iteracích.

Algoritmus CAMShift udržuje dobu výpočtu v průběhu sekvence pod hranicí deseti milisekund. CAMShift algoritmus s dynamickým modelem cíle podle provedených testů zpracovává snímky nejdéle. V porovnání s algoritmem Mean shift, zachyceným na obrázku 42, se jeví zbývající algoritmy pomalé, ale i jejich doba zpracování výpočtu pod patnáct milisekund, je dostačující pro sledování v reálném čase. Za povšimnutí zvláště stojí algoritmus CAMShift, který je zachycený na obrázku 43. V závěru sekvence v intervalu snímků 726 až 766, se potýkaly všechny testované algoritmy s problémy nalezení cíle. V tomto případě i algoritmus Mean shift provedl výrazně více iterací a zpracovával výpočty o poznání déle, než tomu bylo po celou dobu sledování. Algoritmus CAMShift v této situaci obstál mnohem lépe, ačkoliv odchylka v počtu iterací je znatelná, cíl byl znovu nalezen o více než polovinu iterací méně, než v případě Mean shift algoritmu.

Důvodem nižší doby výpočtu v podání Mean shift algoritmu je fakt, že algoritmus neprovádí výpočet zpětné projekce. To s sebou nese další výhody. Není zde potřeba provádět výpočet histogramu pro celý snímek a rovněž není potřeba provádět výpočet pro nové rozměry sledovací oblasti. Oba zbývající algoritmy již výpočet zpětné projekce provádějí, nicméně CAMShift algoritmus s dynamickým modelem cíle je z trojice testovaných algoritmů nejpomalejší. Důvodem je, že algoritmus jako jediný provádí sestavení časově náročnějšího histogramu s vyváženým pozadím. Ten je nutný sestavovat opakovaně na každém snímku z důvodu pravidelného aktualizování cílového histogramu.

Dobu výpočtu by bylo možné u obou CAMShift algoritmů značně urychlit v případě, kdy by se neprováděl výpočet zpětné projekce pro celý snímek, ale pouze pro oblast, která by byla větší než aktuální velikost sledovací oblasti. V tomto případě by se rychlost výpočtu mnohonásobně urychlila. Toto rozšíření však nebylo v této práci implementováno z důvodu použití detekčního mechanismu. Tento je popsán v 5.2 v podání prudkého zvětšení sledovací oblasti v případě ztráty sledovaného objektu.

8.3 Vyhodnocení testů

Jak je patrné z výše uvedených srovnávacích testů, nejlepších výsledků bylo dosaženo použitím algoritmu Mean shift, který neselhal v žádném výše zmíněném testování. Také algoritmus CAMShift v testování obstál, selhání bylo pozorováno pouze v jedné z testovacích sekvencí a důvod toho selhání byl objasněn. Algoritmus CAMShift s dynamickým modelem cíle, ačkoli disponuje technikou mnohem pokročilejší, než je u předchozích dvou algoritmů, dosáhl v testování nejnižších výsledků. V následující části budou detailněji popsána chování jednotlivých algoritmů.

8.3.1 Vyhodnocení algoritmu Mean shift

Tento algoritmus ačkoli jako jediný nedisponuje adaptivní velikostí sledovací oblasti, obstál v testování nejlépe jak z pohledu přesnosti, tak i z pohledu doby výpočtu. Přesnost algoritmu je diskutabilní s porovnáním algoritmu CAMShift, který v jistých situacích dosahoval větší přesnosti, zvláště v kritických oblastech. Nicméně selhání tento algoritmus řadí na druhou pozici.

Z pohledu výpočetní náročnosti dosahuje algoritmus Mean shift nejlepších výsledků z důvodu, že zde není použita technika zpětné projekce. V tomto případě se neanalyzuje celý snímek, nýbrž pouze jen jeho určitá část, vztažená k aktuální pozici sledovací oblasti. Nárůst rychlosti doby výpočtu je v takovém případě značný. Jelikož algoritmus nevyužívá zpětné projekce, nebylo možné provést výpočty za účelem adaptace sledovací oblasti, a tak přesně ohraničovat sledovaný objekt v závislosti na vzdálenosti od objektivu nebo při clonění jiným objektem. I přes tuto nevýhodu algoritmus dosahoval nejvyšších výsledků, avšak z pohledu pozorovatele nemusí být zcela jasné, jaký objekt je sledován. Například v testovací sekvenci s názvem Face2, ve chvíli kdy je objekt značně vzdálen od objektivu, velká sledovací oblast ohraničuje malý, v prostoru vzdálený objekt. V případě, kdy by do této velké sledovací oblasti vstoupil cizí objekt podobné barevné reprezentace, jako má model cíle, pravděpodobně by došlo k selhání algoritmu. V případě, že by měl být algoritmus použit za účelem měření, například rozměrů objektu v závislosti na vzdálenosti nebo na jeho deformaci, byl by algoritmus pro tento účel nepoužitelný. Další problém by mohl nastat právě ve zmíněné situaci, kdy by do velké sledovací oblasti obklopující malý cíl vstoupil další objekt stejné barvy. V této situaci by algoritmus pravděpodobně selhal, zejména pokud by byl cizí objekt větších rozměrů než původní cíl.

Důvodem, proč algoritmus dosahoval nejpřesnějších výsledků, je paradoxně jeho nevýhoda a to právě nemožnost adaptace sledovací oblasti. Obecně u algoritmu založených na Mean shift iteracích platí, čím je větší sledovací oblast, tím je větší šance na zachycení cíle. Algoritmu stačí zachytit pouze část sledovaného objektu a pomocí iterací je schopen vycentrovat střed sledovací oblasti i navzdory větší vzdálenosti středu sledovací oblasti od sledovaného objektu.

8.3.2 Vyhodnocení algoritmu CAMShift

Algoritmus CAMShift se statickým modelem cíle dosahoval rovněž vysokých výsledků. Došlo zde pouze k jednomu selhání a to v důsledku nevhodně označeného cíle. V případě korektnějšího označení algoritmus dosahoval velice přesných výsledků. Jeho velkou výhodou je použití adaptivní sledovací oblasti, jenž ohraničuje sledovaný objekt dostatečně přesně. Tato výhoda s sebou nese úskalí v podobě nárůstu doby výpočtu, z důvodu výpočtu obrazu zpětné projekce a velikosti sledovací oblasti. V poslední testovací sekvenci bylo předvedeno, jak důležitá je korektní volba sledovací oblasti a jak její výběr může vést k selhání algoritmu. Byl zde také popsán důvod tohoto selhání.

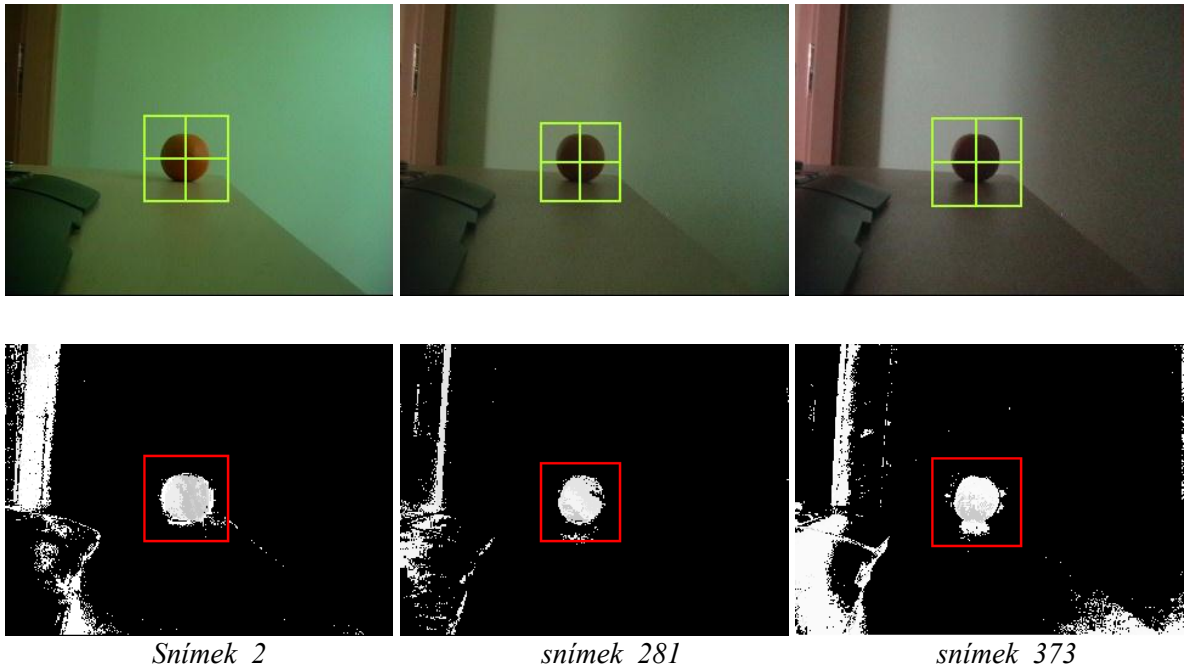
U algoritmu CAMShift se sledovací oblast na každém snímku přizpůsobí části sledovaného objektu, která je zastoupena barvami, které obsahuje histogram cíle. Sledovací oblast ohraničí pouze tu část objektu, která v obrazu zpětné projekce není nijak přerušena, tedy označuje pouze celistvé, spojité, barevné části. Tato vlastnost je uplatněna zejména v případech, kdy je předmět cloněn cizím objektem jiné barvy. Tato vlastnost také umožňuje indikovat, kdy je cíl ztracen. Sledovací oblast se v takovém případě neustále zmenšuje až na minimální předepsanou hranici velikosti, kdy je rozhodnuto, že cíl byl ztracen. V takovém případě lze reagovat zvětšením sledovací oblasti, ve snaze zachytit část sledovaného objektu. V této situaci je cíl znovu nalezen pomoci iterací algoritmu, stejně jako bylo popsáno v 8.3.1 u algoritmu Mean shift. S využíváním adaptivní sledovací oblasti, je možné algoritmus použít například pro účely měření. Algoritmus CAMShift se používá pro sledování tváře v knihovnách Intel Open Source Computer Vision [13], kde je objekt reprezentován modelem jenž je sestaven pomocí barevného prostoru HSV (hue, saturation, value). V podání Intel Open Source Computer Vision, je sledování zaměřeno na barvu lidského těla, pomocí jednorozměrného histogramu, jak je uváděno v [10]. V tomto případě je sledování založeno na předpokladu, že barva těla je stejného odstínu (stejná hodnota hue). Problém může nastat v případě, kdy má být algoritmem sledován objekt, který se skládá z barev odlišných odstínů.

8.3.3 Vyhodnocení algoritmu Continuously adaptive mean shift s dynamickým modelem cíle

Algoritmus, který disponuje pokročilou technikou průběžného aktualizování cíle na každém snímku, dosáhl ve srovnávání nejnižších výsledků. Důvodem selhávání a značných odchylek algoritmu od skutečné polohy, je právě technika průběžných aktualizací histogramu cíle. Jak bylo zmíněno v podkapitole 5.3, pravidelné aktualizování histogramu cíle je velice náchylné na infikování pozadím objektu. Ačkoli byl za účelem eliminace nežádoucích vlivů pozadí použit histogram vyváženého pozadí, nebyly všechny nežádoucí elementy odstraněny vždy úspěšně. Odstranění pozadí je velice obtížné a přes vynaloženou námahu, nebyl nalezen způsob jak nežádoucí vlivy eliminovat dostatečně. Algoritmus CAMShift s dynamickým modelem cíle využívá adaptivní sledovací oblasti, jenž má stejné chování jako v případě původního CAMShift algoritmu. S funkcí aktualizací histogramu cíle, je možné vypočítat histogram celého souvislého objektu. Zejména v případě testovací sekvence s názvem Face2, je vidět plynulé roztažení sledovací oblasti po celé tváří a sledovací oblast v obrazu zpětné projekce obsahuje minimum černých pixelů, jenž reprezentují barvy, které cíli nepatří.

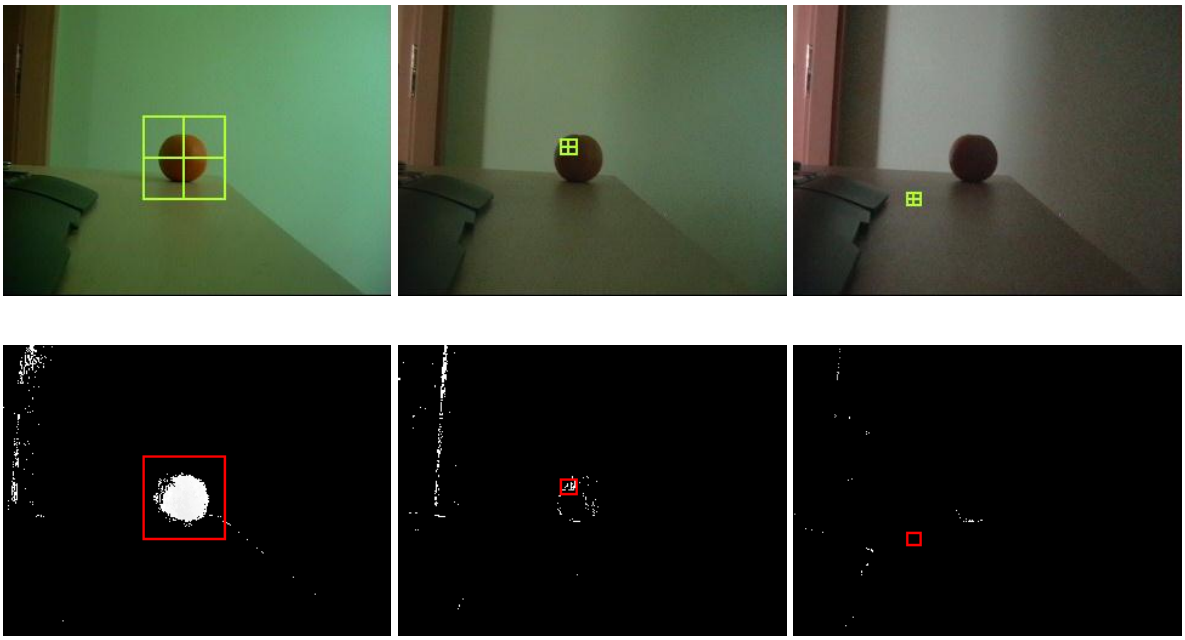
V porovnání s CAMShift algoritmem je tento rozdíl znatelný. Navíc je zde pozorovatelná snaha o adaptaci sledovací oblasti a tedy i histogramu cíle, na maximální plochu, která obsahuje pixely stejné barvy jako má objekt, ovšem pouze v případě, kdy je mezi jednotlivými barevnými částmi lineární přechod. V případě sekvence Face2 se sledovací oblast rozšíří tak, aby zachytila tvář i krk objektu, které obsahují barvy stejného odstínu. CAMShift algoritmus s dynamicky se měnícím modelem cíle, je schopen úspěšného sledování v případě, kdy sledovaný objekt prochází barevnými změnami, jako mohou být například změny osvětlení.

Ukázka sledování jednotlivých algoritmů v situacích, kdy dochází ke změně osvětlení, je zachyceno na obrázcích 45, 46 a 47. Na obrázku číslo 45 je zachycen průběh sledování algoritmem CAMShift s dynamickým modelem cíle. Jak je vidět na obrazech zpětné projekce, algoritmus jako jediný přesně identifikuje sledovaný objekt i v případě silných změn osvětlení. Oproti tomu algoritmus CAMShift se statickým modelem cíle na obrázku 46, sledovaný objekt ztratil v důsledku změn osvětlení. Z obrazu zpětné projekce na snímku 373 pro CAMShift algoritmus je zřejmé proč ke ztrátě cíle došlo. Statický histogram cíle neobsahuje takovou barevnou reprezentaci, která by byla na snímku 373 k nalezení. Na obrázku s číslem 47 je ukázka sledování Mean shift algoritmem. U tohoto algoritmu není obraz zpětné projekce k dispozici a také díky statické velikosti sledovací oblasti, není možné přesné určení situace, v které se algoritmus nachází.



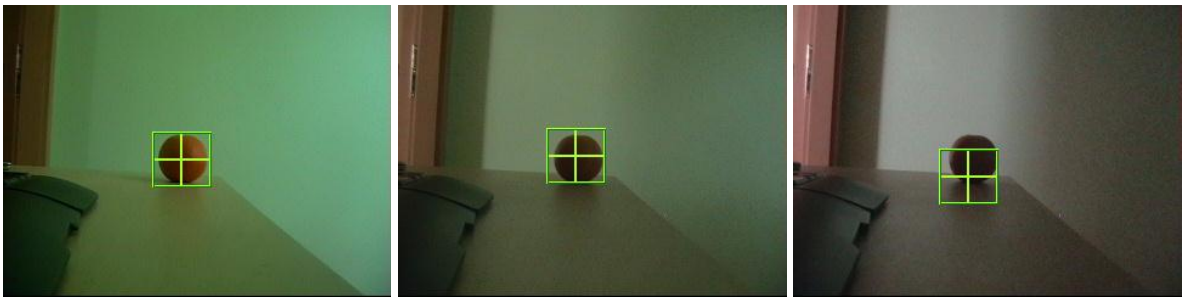
Snímek 2 *snímek 281* *snímek 373*

Obr. 45 Sledování algoritmem CAMShift s dynamickým modelem cíle.



snímek 2 *snímek 281* *snímek 373*

Obr. 46 Sledování algoritmem CAMShift se statickým modelem cíle.



snímek 2 *snímek 281* *snímek 373*

Obr. 47 Sledování algoritmem Mean shift.

Na snímku s číslem 373 je možné pozorovat, že sledovací oblast je vystředěna na spodní část sledovaného objektu. Po nahlédnutí na snímek zpětné projekce s číslem 373 na obrázku 46, je pozorovatelné že se v těchto místech objektu nachází tenká světlá křivka, reprezentující barvy histogramu cíle. Je tedy pravděpodobné, že algoritmus Mean shift na obrázku s číslem 47 uvízl právě na této malé části barevné reprezentace a efekt statické velikosti sledovací oblasti působí tak, jako by algoritmus sledoval cíl s vyšší přesností.

8.3.4 Nežádoucí vlivy sledování

Nežádoucí dopad na průběh sledování může mít často použité technické vybavení nebo prostředí v němž se sledování provádí. Uvedené testy byly prováděny na sekvencích, které byly pořízeny v dostatečně osvětleném prostředí. Špatně osvětlené prostředí má negativní dopad na průběh celého sledování.

Dalším aspektem, který má značný vliv na výsledky sledování, je použité technické vybavení. Běžně dostupné webové kamery jsou často vybaveny nedostatečným automatickým zaostřováním a mohou tak způsobovat rozmazávání sledovaného pohybujícího se objektu. Rovněž sledování závisí na frekvenci pořizovaných snímků za vteřinu. Příliš pomalá frekvence snímání může způsobit velké rozestupy mezi sledovaným objektem na jednotlivých snímcích, což může vést k selhání algoritmu. Stejná situace může nastat v případě, kdy se objekt pohybuje vyšší rychlostí rychlostí.

9 ZÁVĚR

Cílem této diplomové práce bylo nalézt vhodný algoritmus pro obecné sledování objektu v dynamickém obraze. Implementovaný algoritmus měl splňovat požadavek v podobě frekvence zpracování pěti snímků za vteřinu, při rozlišení snímku 640 na 480 pixelů. Z tohoto důvodu bylo nejprve nutné zjistit možnosti známých sledovacích algoritmů. K tomuto účelu byla vypracována stručná rešerše nalezených algoritmů.

Na základě prostudování nalezených zdrojů byl k implementaci vybrán algoritmus Mean shift, a to pro svou vysokou frekvenci zpracovávání jednotlivých snímků, možnost obecného použití, relativní jednoduchost a velice dobrou adaptovatelnost na robustnější algoritmy. Tento algoritmus umožňuje sledování v dynamickém obraze na základě barevné struktury vybraného cíle. K vytvoření modelu sledovaného objektu bylo použito techniky trojrozměrného barevného histogramu, jenž se sestává z hodnot jednotlivých složek barevného prostoru RGB. Podrobnému popisu sestavení trojrozměrného histogramu byla věnována třetí kapitola.

V této práci byl implementován základní Mean shift algoritmus a jeho adaptace, které využívají odlišné techniky k vyhodnocení pravděpodobné polohy cíle, a to užitím techniky zpětné projekce. Tyto dvě adaptace byly implementovány za účelem přizpůsobování velikosti sledovací oblasti, jenž ohraničuje sledovaný objekt. Důvodem byla skutečnost, že obecný Mean shift algoritmus touto funkcionalitou implicitně nedisponuje. První adaptací je algoritmus označován jako Continuously adpative mean shift (CAMShift), který umožňuje implicitní adaptaci zmíněné sledovací oblasti. Druhou adaptací je předchozí algoritmus CAMShift doplněn o průběžné aktualizování modelu cíle na každém snímku sekvence.

Samotná implementace je provedena v objektově orientovaném programovacím jazyce C#. Při implementaci byl především kladen důraz na optimalizaci prováděných výpočtu, čímž bylo dosaženo, že aplikaci je možné nasadit i na méně výkonné počítače. Rovněž byl při implementaci kladen důraz na univerzálnost použití. Architektura aplikace byla navržena tak, aby byla schopná posloužit jako framework pro sledovací algoritmy, které pracují na obdobném principu sledování. Toho bylo dosaženo užitím objektově orientovaného návrhu. Rovněž byl kladen důraz na minimalizaci zásahů v případě modifikace současně implementovaných algoritmů.

Všechny výše zmíněné algoritmy byly pečlivě porovnány v kapitole nazvané Testy a porovnání. V této kapitole jsou provedena porovnání z hlediska přesnosti a doby výpočtu jednotlivých algoritmů. Jsou zde uvedeny výsledky třech testovacích sekvencí. První sekvence je zaměřena na sledování obličeje v závislosti na vzdálenosti od objektivu kamerového zařízení. Sledování tváře bylo pro porovnání algoritmů vybráno také z důvodu, že tvář je daleko komplexnější objekt ke sledování, než v případě jednobarevného předmětu. Účelem druhé testovací sekvence bylo porovnat chování algoritmů v případě, kdy sledovaný objekt je obklopen pozadím podobné barvy. Předmětem poslední sekvence bylo porovnání algoritmů v případech, kdy je sledovaný objekt v průběhu sekvence částečně nebo zcela cloněn cizím objektem. Byla zde také vysvětlena a demonstrována relevantnost přesného výběru objektu, jenž má být sledován.

Z provedených testů vyplynulo, že základní Mean shift algoritmus díky své statické velikosti sledovací oblasti a uchováváním statického modelu cíle dosáhl nejvyšších výsledků a v žádném z uvedených testů neselhal. Rovněž je nejrychlejším z testovaných algoritmů z důvodu, že se zde neanalyzuje celý snímek jako v případě CAMShift algoritmu. Algoritmus CAMShift vykazoval rovněž vysokou přesnost navzdory tomu, že v poslední testovací sekvenci selhal v důsledku nepřesné inicializace cíle. Tento algoritmus ovšem pečlivě plnil svou úlohu a relativně přesně označoval velikost sledovaného objektu v závislosti na vzdálenosti od objektivu kamerového zařízení, nebo v závislosti na případném clonění cizím objektem. Nejnižších výsledků dosáhl algoritmus CAMShift s dynamickým modelem cíle. Navzdory tomu, že tento algoritmus disponuje nejpokročilejší technikou dynamické adaptace modelu cíle podle aktuální velikosti sledovací oblasti, bylo zaznamenáno několik značných odchylek a v jednom případě i selhání algoritmu. Důvodem nepřesnosti tohoto algoritmu v některých situacích je paradoxně jeho výhoda. V průběhu sledování často nastane situace, že ve chvíli kdy algoritmus aktualizuje svůj model cíle, obsahuje adaptovaná sledovací oblast část pozadí, které se

nepodařilo eliminovat. V takovém případě je nově vypočítaný model cíle infikován nežádoucím pozadím objektu. Algoritmus vyhodnotí pozadí objektu jako součást cíle a sledovací oblast se přizpůsobí tak, že začne označovat i nežádoucí pozadí objektu, což vede ke ztrátě cíle a následnému selhání algoritmu. Výhodou algoritmu CAMShift s dynamickým modelem cíle je fakt, že si nejlépe vedl v testování při změně osvětlení, a tedy změně barvy sledovaného objektu. V tomto testu jako jediný nevykazoval výchylky a úspěšně sledoval objekt do konce sekvence, včetně označení objektu.

Podle provedených testů a srovnání je možné odvodit doporučení sledovacích algoritmů dle požadavků, které jsou na sledování kladeny. Algoritmus Mean shift je doporučeno použít v případech, kdy se jedná o sledování v reálném čase, které je prováděno na počítači s menším výkonem, nebo pokud je sledování prováděno na kamerovém zařízení, které disponuje vysokou snímkovou frekvencí. Algoritmus je možné doporučit pro případy, kdy se vzdálenost objektu od objektivu kamerového zařízení nemění, rovněž také v případech kdy objekt nemění rozměry, nebo na velikosti sledovací oblasti nezáleží.

V závislosti na provedených testech je rovněž možné odvodit doporučení pro algoritmus CAMShift. Tento algoritmus je vhodné použít v případech, kdy rozměry sledovaného objektu a přesnost označení objektu v prostoru jsou pro účely sledování prioritní, například za účelem měření. Také je vhodné použití pro sledování tváře nebo v případech, kdy bude potřeba sledovaný objekt v průběhu sledování detekovat. Algoritmus CAMShift neprokázal dobré výsledky při sledování v prostředí, kdy se mění světelné podmínky a tedy i barva objektu. Pro algoritmus CAMShift s dynamickým modelem cíle je možné odvodit doporučení pro případy, kdy jsou rovněž rozměry a označení objektu prioritní. Také v případě kdy může nastat situace, ve které se sledovaný objekt bude průběžně měnit, například změna barvy nebo osvětlení. Důvodem pro toto doporučení je fakt, že funkce dynamického aktualizování modelu cíle se projevila jako efektivní technika při sledování objektu v nestálých světelných podmínkách. Z důvodu neuspokojivých výsledků obdržených z testování, je vhodné tento algoritmus použít pouze v případech, kdy je sledovaný objekt odlišen od pozadí a nepohybuje se vyšší rychlosti.

SEZNAM POUŽITÉ LITERATURY

[1] KHAN, Arif. The Code Project, Computer Vision Applications with C# - Part I, [online]. duben 2009, [citace 05.02.2010]. Dostupné z: <http://www.codeproject.com/KB/GDI-plus/CV_Intro.aspx>.

[2] KHAN, Arif. The Code Project, Computer Vision Applications with C# - Part II, [online]. duben 2009, [citace 7.02.2010]. Dostupné z: <http://www.codeproject.com/KB/GDI-plus/MomentsTracking.aspx>.

[3] KHAN, Arif. The Code Project, Computer Vision Applications with C# - Part III, [online]. květen 2009, [citace 10.02.2010]. Dostupné z: <http://www.codeproject.com/KB/GDI-plus/MeanshiftTracking.aspx>

[4] COMANICIU, RAMESH, MEER. Kernel-Based Object Tracking. [online]. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 25, NO. 5, MAY 2003, [citace 16.03.2010]. Dostupné z:<http://www.ieee.org/index.html>.

[5] COMANICIU, RAMESH, MEER. Real-Time Tracking of Non-rigid Objects using Mean Shift, [online]. 2000, [citace 18.03.2010]. Dostupné z: <http://comaniciu.net/Papers/MsTracking.pdf>.

[6] YILMAZ, JAVED, SHAH. Object Tracking: A survey. [online]. ACM Computing Surveys, Vol. 38, No. 4, Article 13, prosinec 2006, [citace 25.03.2010]. Dostupné z: <http://www.cs.ucf.edu/vision/public_html/papers/Object%20Tracking.pdf>.

[7] GARY, BRADSKI. Computer Vision Face Tracking For Use in a Perceptual User Interface, [online]. 1998, [citace 05.04.2010]. Dostupné z: <ftp://download.intel.com/technology/itj/q21998/pdf/camshift.pdf>.

[8] ZHANG, QIAO. An improved CamShift algorithm for target tracking in vide surveillance, [online]. 2009, [citace 22.09.2010]. Dostupné z: <http://arrow.dit.ie/cgi/viewcontent.cgi?article=1001&context=ittpapnin>.

[9] Wikipedie – RGB [online]. [citace 5.12.2010]. Dostupné z: <http://cs.wikipedia.org/wiki/RGB>.

[10] ALLEN, XU, JIN. Object tracking using CamShift Algorithm and Multiple Quantized Feature Spaces, [online]. 2006, [citace 02.01.2011]. Dostupné z: <http://crpit.com/confpapers/CRPITV36Allen.pdf>.

[11] Aforge.NET Framework [online]. verze 2.0.1.0. Dostupné z <http://www.aforgenet.com/framework>.

[12] HEWITT, Robin. How OpenCV's Face Tracker Works [online]. březen 2007, [citace 16.04.2011] Dostupné z: <http://www.cognotics.com/opencv/servo_2007_series/part_3/sidebar.html>.

[13] HEWITT, Robin. Seeing With OpenCV, Part 3: Follow that Face [onine]. březen 2007, [citace 17.04.2011] Dostupné z: <http://www.cognotics.com/opencv/servo_2007_series/part_3/index.html>.

SEZNAM PŘÍLOH

Vlastní text diplomové práce
Aplikace Sledování objektu
Aplikace pro vytváření snímkových sekvencí
Testovací snímkové sekvence
Výsledky jednotlivých testování a výstupy algoritmů