OBSAH

SEZNAM OBRÁZKŮ

1. ÚVOD

Základem této práce je seznámení se se zpracováním digitálního obrazu a popisem scény, zvláště pak rozdílů ve snímcích a způsobu pohybu těles v nich. Dále pak vytvoření algoritmu vhodného pro nalezení a predikci stejných objektů na sérii snímků. Také jsou stanoveny podmínky, za kterých mají takovéto predikce smysl.

Obecně můžeme scény kategorizovat do několika typů (různé druhy klasických filmů, scéna místnosti, ulice nebo průmyslová resp. syntetická scéna). Každou z těchto typů scén lze charakterizovat podle barvy, velikosti a pohybu objektů v nich.

Tato práce se zabývá výhradně syntetickými scénami, které mají blízko k průmyslu a ke sportu. Lze je charakterizovat statickým umístěním kamery a pohybujícími se objekty scény. Obsažené objekty jsou jednoduché a pohybují se v konstantních vzdálenostech od kamery.

Tuto cestu jsem si vybral proto, že lze takovéto scény chápat jako zobecnění světa. Pokud se úspěšně podaří vyřešit trasování a predikce na syntetických scénách, je možné dalším rozšířením algoritmů postihnout i komplikovanější úlohy.

Celou práci lze rozdělit do dvou tematických částí:

První část je teoretická. Zabývá se shrnutím základních pojmů počítačového vidění a zpracování obrazu a základními nástroji pro práci s digitálními obrazy jako jsou segmentace, filtrace a detekce hran.

V druhé části práce jsou postupně popsány jednotlivé části navrženého programu, jako je výběr vhodného operátoru pro nalezení hran objektů, způsob získání jednotlivých scén, popis samotného řešení nalezení trasy a predikce stejných objektů na sérii snímků. Nakonec jsou shrnuty dosažené výsledky nalezení, trasování a predikce objektů na syntetických scénách.

2. ZÁKLADNÍ POJMY

2.1 POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je vědní obor, který se technickými prostředky snaží napodobit některé schopnosti lidského vidění. Při vyhodnocení obrazové informace hraje podstatnou roli člověk a jeho předchozí zkušenosti. Také je v širším smyslu považováno za součást kybernetiky popř. umělé inteligence.

Interpretace obrazových dat je základem porozumění obrazu v počítačovém vidění. Snahou je napodobit proces vnímání u člověka a jemu podobný způsob rozhodování na základě informace obsažené v obrazech. Příčiny obtíží v interpretaci obrazových dat v úlohách počítačového vidění lze shrnout do následujících bodů [5]:

- Ztráta informace při perspektivním zobrazení původně trojrozměrné scény do dvojrozměrné roviny čidla kamery. Všechny body na polopřímce dané bodem *(x,y)* v obrazové rovině ve směru od středu promítání se zobrazí právě do bodu *(x,y)*. V opačném případě, kdy se odvozují trojrozměrné vlastnosti objektů z obrazu jediné kamery, má nekonečně mnoho řešení. Ve skutečnosti ji lze řešit jen tehdy, pokud je možné využít, dalších dodatečných znalostí.
- Komplikovaný a nejednoznačný vztah mezi jasem, který měří kamera a tvarem povrchu 3D objektů ve scéně. Jas bodu závisí na vlivech jako: odrazivosti povrchu pozorovaného předmětu, poloze a vlastnostech zdrojů světla, orientaci povrchu vzhledem k pozorovateli.
- Velké množství obrazových dat.
- Šum, který je v obraze reálné scény vždy přítomen, je důvodem, proč je při zpracování velmi často nutné použít pravděpodobnostní techniky. Často však není k dispozici tolik obrazů, aby bylo možné korektně odhadnout statistické vlastnosti obrazových signálů.

2.2 REPREZENTACE OBRAZU

Mezi vstupním obrazem a modelem se definuje několik úrovní reprezentace obrazové informace. Počítačové vidění se potom skládá z návrhu těchto přechodových reprezentací a algoritmů pro jejich vytváření a zavedení vztahu mezi nimi.

Reprezentace obrazu lze podle organizace dat rozdělit do čtyř úrovní. Hranice mezi nimi ale nejsou zcela jednoznačné a v aplikacích se často používá jemnější odstupňování.

- Nejnižší úrovní reprezentace jsou digitální obrazy, které stále mají podobu obrazových dat, tzn. celočíselné matice s údaji o jasu v příslušných bodech. Tyto obrazy jsou i výstupem operací předzpracování, které slouží pro vylepšení obrazu před dalším zpracováním.
- Druhou úrovní reprezentace jsou příznaky. Části obrazu jsou zde spojeny do skupin, které pravděpodobně patří k jednotlivým objektům.
- Třetí úrovní reprezentace jsou objekty, které jsou výsledkem segmentace, tj. úplné interpretace obrazových dat. Ve složitějších úlohách se podaří jenom částečná segmentace, tzn. že interpretaci mají jen části obrazů, ale jejich interpretace jako objektů je možná až při dalším, spíše kvalitativním posuzování.
- Nejvyšší úrovní reprezentace obrazových dat jsou relační modely, které postihují kvantitativní i kvalitativní vlastnosti objektů v obraze. Počítačové vidění zde využívá technik rozpoznávání a umělé inteligence.

2.3 ZPRACOVÁNÍ OBRAZU

Zpracování obrazu lze chápat jako součást zpracování signálů, protože obraz je v podstatě vícerozměrný signál. V této práci budeme za obraz považovat např. fotografii, obraz sejmutý digitální kamerou.

Obraz můžeme reprezentovat jako spojité rozložení jasu nebo intenzity v rovině. Pro matematický popis použijeme matematický model, kterým je obrazová funkce [3]:

$$z = f(x, y) \quad \text{- pro statický obraz} \tag{2.1}$$

$$z = f(x, y, t) \quad \text{- pro dynamický obraz} \tag{2.2}$$

V počítačové grafice většinou nemáme k dispozici spojitý definiční obor funkce. Zpravidla pracujeme v rastru, který je tvořen obrazovými elementy – pixely.

V počítači pracujeme s digitalizovanými obrazy, kde je obrazová funkce představovaná maticí. Prvky této matice jsou pixely. Jejich hodnota je úměrná množství světelné energie. Z pohledu digitalizovaného obrazu je obrazový element nejmenší dále nedělitelná jednotka (pro jednoduchost se někdy používá termín bod, i když se jedná o pixel konečných rozměrů).

Pro další zpracování je obraz diskretizován v čase a lze ho chápat jako nespojitou sekvenci obrazových funkcí *f(x,y)* [4].

Definiční obor *D* hodnot obrazové funkce:

$$x \in (x_{\min}, x_{\max}); y \in (y_{\min}, y_{\max})$$

Obor hodnot *H* obrazové funkce:

$$0 \leq f(x, y) \leq f$$

Takto definovaná funkce ovšem odpovídá pouze šedotónovému nebo také monochromatickému obrazu s proměnnou intenzitou jedné barvy. Většinou ale pracujeme s barevnými obrazy. Barevný obraz pak lze definovat jako soubor několika obrazových funkcí [3]:

$$f_r(x, y); f_g(x, y); f_b(x, y) \tag{2.3}$$

Zde jednotlivé funkce odpovídají intenzitě jasu pro danou barevnou složku.

3. DIGITALIZACE OBRAZU

Digitální kamera, která je vstupem obrazové funkce je zdrojem spojitého signálu. Aby bylo možné v počítači obrazovou funkci zpracovat, je nutné ji digitalizovat. Obraz je dvojrozměrný.

Získáním počítačového obrazu docílíme přechodem spojité obrazové funkce *(2.1)* k diskrétní funkci jak v definičním oboru *D*, tak v jejím oboru hodnot *H*. Proces přechodu od spojitého obrazu k diskrétnímu se nazývá digitalizace.

Digitalizace v podstatě spočívá ve vzorkování obrazu v matici *M x N* bodů a v kvantování spojité jasové úrovně každého vzorku do K intervalů. Z tohoto důvodu nabývá v digitalizovaných obrazech jasová funkce celočíselných hodnot. Čím jemnější je vzorkování (tedy čím větší rozměr má matice *M x N*) a kvantování, tím lépe je aproximován obrazový signál.

3.1 VZORKOVÁNÍ

Vzorkováním spojité funkce máme na mysli zaznamenávání hodnot (vzorků) v předem daných intervalech. V souvislosti se vzorkováním spojité obrazové funkce *(2.1)* je třeba stanovit interval vzorkování a vzorkovací mřížku.

Interval vzorkování je vzdálenost mezi nejbližšími vzorkovacími body v obraze. Problém vzdálenosti vzorků řeší Shannonova věta o vzorkování [5]. Interval vzorkování se musí volit tak, aby byl menší nebo rovný polovině rozměru nejmenších detailů v obraze. Při zpracování obrazů je vhodné vzorkovat alespoň 5krát jemněji než je teoretická mez, která je daná vzorkovací větou.

Při výběru vzorkovací mřížky, tedy plošného uspořádání bodů při vzorkování se používá zpravidla pravidelná mřížka. Existují jen tři pravidelné mnohoúhelníky, jejichž síť úplně pokrývá rovinu. Těmi jsou: rovnostranné trojúhelníky, čtverce a pravidelné šestiúhelníky. V praxi se nejvíce používá čtvercová mřížka, i když je příčinou problémů se spojitostí oblastí

Jednomu vzorkovacímu bodu odpovídá v digitalizovaném obraze jeden pixel. Po uspořádání do vzorkovací mřížky pokrývají pixely celý digitalizovaný obraz.

3.2 KVANTOVÁNÍ

Amplituda ve vzorkovaném obraze musí být pro zpracování počítačem vyjádřena jako digitální údaj. Kvantování probíhá v oboru hodnot *H* obrazové funkce, který se dělí na intervaly. Těm je následně přidělena jediná zástupná hodnota. Nejčastěji se používá průměr celého intervalu, někdy se používá váženého průměru, mediánu, atp.

Počet kvantovacích úrovní musí být dostatečně velký, aby byly přesně vyjádřeny jemné detaily obrazu a nevznikaly tak falešné obrysy. Podle způsobu rozdělení kvantované veličiny dělíme kvantování na uniformní a neuniformní [1].

Většinou se používá uniformní kvantování, které používá konstantní délku intervalu. Důvodem je především jednodušší realizovatelnost. Pokud je pro reprezentaci informace o obrazovém elementu použito *b* bitů, pak je počet úrovní jasu roven $k=2^b$. Největším problémem při kvantování tímto způsobem je možný vznik falešných obrysů a obrazů kvantovaných do nedostatečného počtu jasových úrovní. Tento problém lze do jisté míry řešit použitím nelineárního, tedy neuniformního kvantování.

Neuniformní kvantování používá proměnnou délku intervalu. Umožňuje tedy zohlednit nerovnoměrné rozložení hodnot měřené veličiny. Také zvětšuje rozsah těch intervalů jasu, jejichž zastoupení není v obraze pravděpodobné. V praxi se však tento způsob kvantování používá zřídka.

4. FILTRACE A DETEKCE HRAN

4.1 LOKÁLNÍ FILRACE

Metody filtrace (předzpracování) obrazu, využívají k výpočtu nové hodnoty pixelu malé okolí σ reprezentativního pixelu. Celé princip je založen na představě, že se celý obraz systematicky prochází. Kolem reprezentativního bodu je zkoumáno malé okolí σ. Výsledek této analýzy je zapsán do výstupního obrazu jako hodnota reprezentativního pixelu. Podle účelu se metody lokálního předzpracování rozdělují do dvou skupin [5].

- Vyhlazování, které usiluje o potlačení šumu.
- Detekce hran (gradientní operátory), které z hodnot v okolí reprezentativního pixelu odhadují derivaci obrazové funkce.

Vyhlazování a detekce hran jsou ve své lineární podobě v protikladu. Proto jsou navrhovány nelineární metody, které např. vyhlazují a přitom jsou šetrné k hranám a detailům v obraze.

Předzpracování obvykle nepoužívá specifické znalosti o obraze. Je těžké takovou znalost při zpracování odvodit, protože známé okolí σ právě zpracovávaného bodu je malé. Je-li znalost o obrazu k dispozici, potom toho algoritmus zpracování může využít. Někdy takovou znalostí bývají známé statistické parametry šumu.

4.2 LOKÁLNÍ VYHLAZOVÁNÍ OBRAZU

Nejsnadnější je vyhlazování náhodného šumu v případě, kdy máme k dispozici několik obrazů téže předlohy, které se liší právě šumem. Potom je účinné průměrovat hodnotu pixelu o stejných souřadnicích přes více obrázků.

Pokud je k dispozici jediný obraz téže předlohy, využívá se obvykle značné nadbytečnosti údajů v obraze. Sousední pixely mají převážně tutéž nebo blízkou hodnotu jasu. Hodnotu obrazových elementů zkreslených šumem lze opravit na základě analýzy hodnot jasu v jeho vybraném okolí. Hodnota jasu reprezentativního pixelu je nahrazena hodnotou typického reprezentanta mezi hodnotami v okolí nebo kombinací několika hodnot.

Lineární metody vyhlazování

Lineární metody vyhlazování počítávají novou hodnotu aktuálního pixelu jako lineární kombinaci hodnot ze zvoleného okolí.

Základní metodou vyhlazování obrazu je obyčejné průměrování, kde každému bodu přiřadíme nový jas, který je aritmetickým průměrem původních jasů ve zvoleném okolí. Odpovídající konvoluční maska *h* pro okolí 3 x 3 je:

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Někdy se zvětšuje váha středového bodu masky nebo jeho 4-sousedů. Následující vztahy ukazují tyto masky pro okolí 3x3. Větší masky se vytvářejí analogicky:

$$h = \frac{1}{10}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}; h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Základní nevýhodou praktického použití obyčejného průměrování je rozmazávání hran v obraze. S ohledem na tuto skutečnost se obyčejné průměrování většinou používá jen jako pomocná metoda pro výpočet střední hodnoty jasu. Tento mezivýsledek je potom použit v důmyslnějších filtračních metod. Společným principem takových metod je, že průměrují jen tu část okolí, ke které bod pravděpodobně patří.

Nelineární metody vyhlazování

Potíže s rozmazáváním hran mohou částečně eliminovat nelineární filtrační metody. Jejich společným principem je, že se v analyzovaném okolí σ snaží najít tu jeho část (oblast konstantního jasu), do které reprezentativní bod patří. Jen pixely této oblasti se využijí pro hledání jasové hodnoty (např. aritmetickým průměrem nebo výběrem hodnoty jasu jednoho pixelu), která bude reprezentovat celé okolí σ ve výstupním obrázku.

4.3 DETEKCE HRAN

Pixely, kde dochází ke změně hodnoty jasu, se nazývají hrany. Detekce hran (jako lokání předzpracování) slouží k automatickému nalézání takových míst v obraze. Hrana v obraze je dána vlastnostmi obrazového elementu a jeho okolí. Je určena tím, jak náhle se mění hodnota obrazové funkce *f(x,y)*.

Matematickým aparátem pro změny funkce dvou proměnných jsou parciální derivace. Změnu funkce udává její gradient, vektorová funkce ∇, určující směr největšího růstu funkce (směr gradientu) a strmost tohoto růstu (velikost, modul gradientu). Pro spojitou obrazovou funkci *f(x,y)* jsou velikost gradientu $|\nabla f(x,y)|$ a směr gradientu dány vztahy [5]:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad (4.1)$$

Pixely s velkým modulem gradientu se označují jako hranové body. Nalezené hranové body v obraze pomocí lokálních operátorů se pak používají pro hledání hranic objektů. Pokud objektu odpovídá oblast homogenního jasu, odpovídají body hranice pixelům s velkou hodnotou gradientu.

4.4 LAPLACIÁN

Pro určení významnějších hran, tedy míst, kde se jas prudce mění bez ohledu na směrové vlastnosti hran, se používá všesměrový lineární Laplaceův operátor – Laplacián ∇^2. Ten vychází z druhých parciálních derivací. Pro monotónně rostoucí jasovou funkci *f* v příslušném okolí je Laplacián ∇^2 roven nule tam, kde je velikost gradientu $|\nabla|$ maximální [5].

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad (4.2)$$

Hrany lze efektivně třídit podle jednorozměrného jasového profilu ve směru gradientu v daném pixelu. Gradientních operátorů se používá i pro ostření obrazu tak, aby v něm byly strmější hrany, tedy v případě dosažení většího kontrastu. Strmost může být dána modulem gradientu nebo Laplaciánem.

4.5 GRADIENTNÍ OPERÁTORY

Gradientní operátory udávající strmost obrazové funkce lze rozdělit do tří kategorií [5]:

- Operátory aproximující derivace pomocí diferencí. Některé operátory jsou invariantní vůči rotaci, např. Laplacián. Mohou být také počítány konvoluce s jedinou maskou. Jiné, aproximující první derivaci, využívají několik masek odpovídajících příslušné orientaci. Z těch se vybírá ta, která nejlépe lokálně aproximuje obrazovou funkci.
- Operátory založené na hledání hrany v místech tam, kde druhá derivace obrazové funkce prochází nulou, např. Cannyho hranový detektor.
- Operátory, které se snaží lokálně aproximovat obrazovou funkci jednoduchým parametrickým modelem, např. polynomem dvou proměnných.

4.6 KONVOLUČNÍ MASKY APROXIMUJÍCÍ DERIVACE OBRAZOVÉ FUNKCE

Jedná se o operátory, které je možné vyjádřit jako masky pro konvoluci. Jednotlivé operátory jsou pak definovány pomocí příslušného konvolučního jádra *h*. V případě směrových operátorů je pak tolik jader *h*, kolik směrů daný operátor rozlišuje[5].

Robertsův operátor

Jedná se o velmi jednoduchý operátor. Používá jen okolí 2 x 2 aktuálního pixelu. Jeho konvoluční masky jsou definovány vztahy:

$$h_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}; h_2 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Jeho nevýhodou je velká citlivost na šum, protože okolí použité pro aproximaci, je malé.

Laplaceův operátor

Diskrétní podoba tohoto filtru je již uvedena v kapitole *5.4*. Jedná se o gradientní operátor ∇^2, který aproximuje druhou derivaci. Tento operátor je

invariantní vůči pootočení a udává jen velikost hrany, nikoliv její směr. V digitálním obraze je aproximován pomocí diskrétní konvoluce. Konvoluční jádra v okolí 3 x 3 jsou definována vztahy:

$$h_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}; h_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Jeho nevýhodou je velká citlivost na šum a dvojité odezvy na hrany odpovídající tenkým liniím v obraze.

Operátor Prewittové

Tento operátor aproximuje první derivaci. Gradient je odhadován v okolí 3x3 pro osm směrů. Vybrána je vždy ta maska z osmi, které odpovídá největší modul gradientu. 8 základních masek má následující tvar:

$$h_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}; h_2 = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}; h_3 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}; h_4 = \begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

$$h_5 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}; h_6 = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}; h_7 = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}; h_8 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix}$$

Sobelův operátor

Tento operátor se často používá pro detekci vodorovných a svislých hran (k tomu postačí masky h_1,h_3). Základní masky mají následující tvar (ostatní je možné získat pootočením):

$$h_1 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}; h_2 = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}; h_3 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Kirschův operátor

První 3 masky jsou dány vztahy (ostatní je možné získat pootočením):

$$h_1 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{pmatrix}; h_2 = \begin{pmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{pmatrix}; h_3 = \begin{pmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{pmatrix}$$

4.7 DETEKCE HRAN NA ZÁKLADĚ PRŮCHODU DRUHÉ DERIVACE OBRAZOVÉ FUNKCE NULOU

Problémem těchto operátorů, které aproximující derivaci diferencemi v malém okolí, je velká závislost jejich chování na konkrétním obrázku. Velikost masky totiž musí odpovídat velikosti detailů v obrázku. Také jejich citlivost na šum je poměrně veliká.

Základem (tzv. Marrova přístupu) je hledání polohy hrany v obraze v místě průchodu druhé derivace obrazové funkce nulou. První derivace obrazové funkce nabývá svého maxima v místě hrany. Druhá derivace protíná v místě hrany nulovou hodnotu. Hledání polohy hrany v místě průchodu nulou, je díky strmosti přechodu mnohem spolehlivější, než u maxima první derivace [2].

Cannyho hranový detektor

Základní myšlenka tohoto detektoru vychází z představy, že skokovou hranu je možné hledat filtrem, využívá tedy hledání průchodů druhé derivace obrazové funkce nulou. Detektor je optimální pro skokové hrany vzhledem ke třem kritériím:

- Detekční kritérium požaduje, aby významné hrany nebyly přehlédnuty a aby na jednu hranu nebyly vícenásobné odezvy.
- Lokalizační kritérium požaduje, aby rozdíl mezi skutečnou a nalezenou polohou hrany byl minimální.
- Požadavek jedné odezvy zajišťuje, aby detektor nereagoval na jednu hranu v obraze vícenásobně. Tento požadavek je zaměřen zejména na zašuměné a nehladké hrany, což první požadavek nezajistí.

Nejprve byl hranový detektor formulován pro 1D signál a první dvě kritéria optimalizace. Po přidání třetího kritéria, tedy zabráněním vícenásobných odezev, bylo nutné hledat optimální odezvu filtru. Výsledný filtr lze aproximovat filtrací Gaussiánem.

Detektor hran je zobecněn na 2D prostor. Hrana je zde dána polohou, orientací a velikostí. Konvoluce s 2D Gaussiánem a derivace ve směru gradientu vytváří jednoduchý a účinný diferenciální operátor, který poskytuje i orientaci hrany.

Pokud označíme *G* jako 2D Gaussián a chceme obrázek podrobit konvoluci s operátorem G_n, který představuje první derivaci *G* ve směru gradientu *n*, pak:

$$G_n = \frac{\partial G}{\partial n} = n \nabla G \tag{4.3}$$

Pozice hrany pak odpovídá lokálnímu maximu konvoluce obrazové funkce *f* s operátorem G_n ve směru *n* a díky tomu získáváme vztah:

$$\frac{\partial}{\partial n} G_n * f = 0 \tag{4.4}$$

Do rovnice *(5.4)* dosadíme za G_n z rovnice *(4.3)*:

$$\frac{\partial^2}{\partial n^2} G_n * f = 0 \tag{4.5}$$

Tato rovnice ukazuje, jak najít lokální maxima ve směru kolmém na hranu. Protože jsou konvoluce a derivace asociativní operace, můžeme nejdříve realizovat konvoluci obrazové funkce *f* s Gaussiánem *G* a poté vypočítat směrovou druhou derivaci s využitím výpočtu ve směru *n* podle rovnice. Síla hrany (velikost gradientu intenzity *f*) se vypočte podle:

$$\left| G_n * f \right| = \left| \nabla \left(G * f \right) \right| \tag{4.6}$$

Výstup detektoru je obvykle podroben prahování, pomocí kterého se zvolí, které hrany jsou v digitálním obraze považovány za významné. Po prahování však bývají hrany často nesouvislé, ačkoliv by měly tvořit souvislou hranici. Tento problém je možné potlačit prahováním s hysterezí. Silné hrany s modulem gradientu nad vyšším prahem jsou pak považovány za hranové pixely pro dané měřítko. U osamocených slabých hran s menším modulem gradientu než vyšší práh se předpokládá, že pocházejí z šumu. Pokud jsou ale tyto slabé hrany souvislé se silnou hranou, je větší naděje, že mohou být hranovými body. Za ně jsou považovány, když příslušný modul gradientu přesahuje nižší práh. Vyšší a nižší práh se pak nastavuje podle odhadovaného poměru signálu k šumu.

Správné měřítko pro hranový operátor záleží na velikosti objektů v obrázku. Měřítko je dáno střední kvadratickou odchylkou Gaussiánu. Detektor může mít významnější odezvy na hranu ve více měřítkách, v tom případě se používá operátor v nejmenším měřítku, protože ten nejlépe lokalizuje hranu [5].

5. SEGMENTACE OBRAZU

Pomocí segmentace se obraz dělí do částí, které mají úzkou souvislost s předměty. Přínosem segmentace je výrazná redukce objemu zpracovávaných dat. Jedním z hlavních problémů ovlivňujících segmentaci je nejednoznačnost obrazových dat, často doprovázená informačním šumem v obraze.

5.1 SEGMENTACE PRAHOVÁNÍM

Segmentace prahováním je nejjednodušší segmentační postup. Mnoho objektů nebo oblastí obrazu je charakterizováno konstantní odrazivostí či pohltivostí svého povrchu. Potom je možné využít určené jasové konstanty prahu k oddělení objektů od pozadí. Výsledkem prahování je v ideálním případě kompletní segmentace do oblastí.

Prahování je transformace vstupního obrazu *f* na výstupní (segmentovaný) binární obraz *g* podle vztahu:

$$g(x,y)=\begin{cases}1 \rightarrow f(x,y)\geq T\\ 0 \rightarrow f(x,y)<T\end{cases}$$

Hodnotu prahu lze určovat pokusně nebo nějakou metodou automatického určování prahu. Zpravidla není možné úspěšně použít stejný práh na celé ploše obrazu. Jednou z modifikací základního prahování je prahování podle vztahu:

$$g(x,y)=\begin{cases}1 \rightarrow f(x,y)\in A\\ 0 \rightarrow jinak\end{cases}$$

kde *A* je jistá množina úrovní jasu. Takto definované prahování můžeme použít i pro detekci hranic objektů.

Modifikací metody je prahování s více prahy, kdy výsledkem již není binární obraz, ale obraz s velmi omezeným počtem jasových úrovní.

$$g(x,y)=\begin{cases}1 \rightarrow f(x,y)\in A_1\\ 2 \rightarrow f(x,y)\in A_2\\ \mathrm{M}\\ n \rightarrow f(x,y)\in A_n\\ 0 \rightarrow jinak\end{cases}$$

6. POPIS SCÉNY

Scéna je množina prostorových objektů, která je doplněná dalšími informacemi potřebnými pro jejich zobrazení. Smyslem zobrazování scén je nalezení a určení objektů, které se nachází v dané prostorové oblasti, popř. nalezení blízkých objektů, kdy může docházet ke kolizím.

Pro výrazné snížení časových nároků při zpracování scény je nutné vhodně uspořádat informace v prostoru. To přináší nutnost zavedení dalších, pomocných datových struktur. Tyto struktury sice nebývají součástí popisu scény, ale jsou vytvářeny při načtení scény. Struktury mají hierarchický charakter a zachycují 3D prostor. Při práci s pomocnými datovými strukturami rozlišujeme dva kroky [1]:

- Počáteční jednorázové vytvoření datových struktur a jejich naplnění informacemi o objektech ve scéně.
- Využití pomocných datových struktur pro vyhledávání informací o objektech.

6.1 HIERARCHIE OBÁLEK

Hierarchie obálek je jednou z možností reprezantace scény. Vznikají postupným shlukováním objektů a obálek, které je obklopují. Výsledkem hierarchie je stromová struktura. Kořen pak reprezentuje všechny objekty scény, tedy prostor, v němž se scéna nachází.

Ohraničení objektu obálkou spočívá v tom, že uvažovaný test polohy objektu lze provést rychleji s obálkou než s objektem, který uvažovaná obálka obklopuje. Tento test je obecně nazýván předběžný test (výsledkem je hodnota true/false). V případě negativního výsledku, již není potřeba testovat vlastní objekt.

Při optimalizaci algoritmů se používá několik druhů obálek viz *Obr.1* [1]. Nelze přitom zcela jednoznačně určit, která obálka je nejvhodnější, vždy záleží na řešené úloze. Obálky s jednoduchými tvary se snadno vytvářejí a aktualizují při transformaci obklopovaných objektů, ale hůře se těmto objektům přizpůsobují. Často totiž obklopí společně s objektem větší prázdný prostor. Předběžný test pak poskytne

pozitivní výsledek, přestože test s vlastním objektem je negativní. Složitější obálky se pak hůře aktualizují, ale jsou schopny se lépe přizpůsobit objektům.

Přehled vlastností a definice jednotlivých typů obálek [1]:

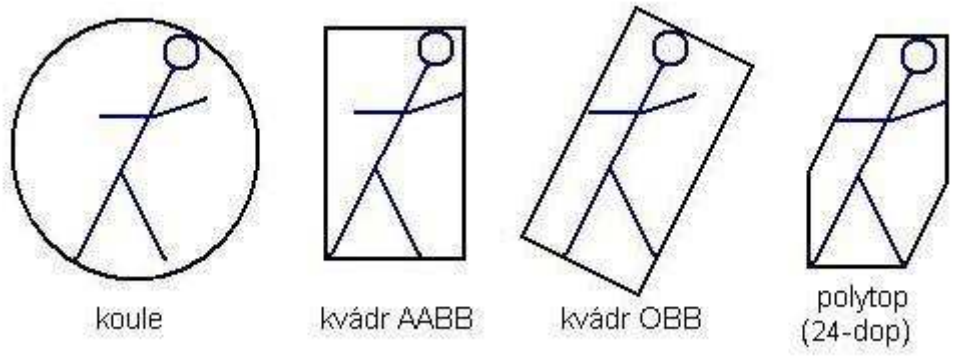
Koule – je invariantní vůči rotaci objektu.

Osově zarovnaný kvádr - AABB – stěny kvádru jsou kolmé na souřadnicové osy, je to současně min-max obálka všech bodů objeku (Axis-Aligned Bounding Box).

Orientovaný kvádr - OBB – obklopující kvádr je orientován tak, aby byl jeho obejm co nejmenší (Oriented Bounding Box).

Orientovaný rovnoběžnostěn - k-DOP – průnik několika pásů v prostoru určuje tzv. polytop (k-Discrete Orientation Polytop). Protilehlé roviny obálky jsou rovnoběžné a jejich normály jsou definovány v *k* diskrétních směrech. Nejčastější varianty jsou:

- 6-dop – kvádr (typu AABB)
- 14-dop – kvádr s oseknutými rohy
- 18-dop – kvádr s oseknutými hranami
- 24-dop – kvádr s oseknutými rohy a hranami



Obr. 1 Typy obálek používané v pomocných datových strukturách

Obálky se tedy seskupují do hierarchické stromové struktury. Obálky, které leží blízko sebe, nebo obálky které se překrývají, se postupně shlukují do větších celků a zapisují se do uzlů, které leží ve vyšších vrstvách stromu.

6.2 DĚLENÍ PROSTORU

Dělení prostoru je další možností reprezentace scény. Využívá systematické rozdělení prostoru scény. Existuje několik způsobů, jak 2D prostor rozdělit na menší části viz *Obr.2*.

Bližší popis technik dělení prostoru [1]:

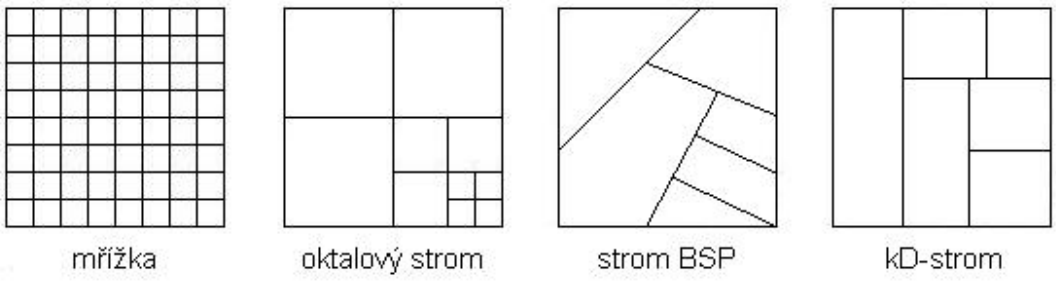
Pravidelná mřížka – svou podstatou není hierarchií, pouze dělí pravidelně prostor rovinami kolmými na souřadnice osy.

Oktalový strom – spočívá v dělení třemi řezy kolmými na souřadnicové osy a umístěnými v polovin daného prostoru. Dělením vzniká osm oktalů.

Strom BSP – binární strom s obecně umístěnými řezy (Binary Space Partitioning).

kD-strom – je speciální případ stromu BSP určený k popisu libovolného *k*-rozměrného (k-Dimensional) prostoru. Řezné roviny jsou vždy kolmé na některou ze souřadnicových os a orientace řezů se pravidelně střídají.

S výjimkou pravidelné mřížky se výsledek dělení prostoru zaznamenává do hierarchické datové struktury – stromu.



Obr. 2 Jednotlivé způsoby dělení prostoru

7. ANALÝZA POHYBU

V případě analýzy obrazů, které jsou proměnné v čase už nepracujeme s jedním, ale s celou sekvencí obrazů (v našem případě pořízených kamerou). Vstupem do systémů analyzující pohyb je posloupnost obrazů snímaných v po sobě následujících okamžicích. Díky tomu významně roste objem dat a objevují se požadavky na zpracování obrazu v reálném čase. Analýza pohybu dynamických obrazů se snaží získat informace o objektech, které se v obraze vyskytují. To zahrnuje jak objekty, které se pohybují, tak objekty, které jsou statické.

Existují přitom různé předpoklady řešených úloh. Zda je pohyb zachycen statickou, nebo pohybující se kamerou. Zda je posloupnost obrazů snímaná v dostatečně krátkých časových intervalech tak, že je možné považovat ji za reprezentaci spojitého pohybu.

Z hlediska pohybu rozlišujeme tři hlavní typy úloh [2]:

- Pouhá detekce pohybu, tedy zjištění detekovaného pohybu. Obvykle je neměnná poloha snímání.
- Nalezení pohybujících se objektů, případně rozpoznání a popis těchto objektů. Zde je obvyklé statické umístění kamery a pohybující se scény s objekty, nebo statická scéna a pohybující se kamera.
- Určení trojrozměrných vlastností objektů s využitím jejich dvojrozměrnou projekcí, které jsou pořízeny v různých časových okamžicích.

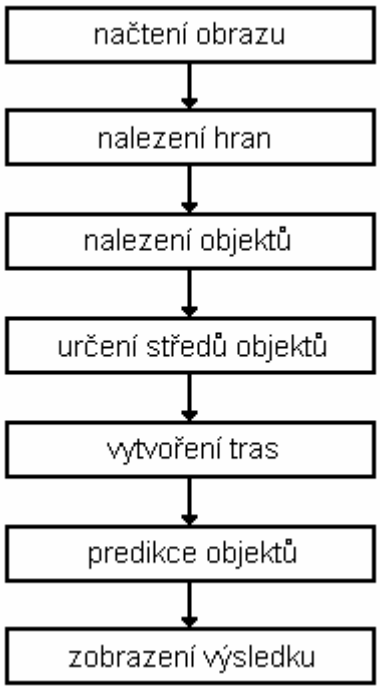
Přestože se zde v souvislosti s analýzou pohybu hovoří o dynamické analýze, velmi často se místo posloupnosti obrazů, které reprezentují pohyb, používá jen několik obrazů pohybu. Tento postup pak spíše odpovídá statické analýze obrazů. Je zde hledána souvislost mezi oblastmi či body jednotlivých obrazů. Těmito postupy se zabývají metody, které využívají srovnávání.

8. PROGRAM PRO VYHLEDÁNÍ OBJEKTŮ VE SCÉNĚ A NALEZENÍ JEJICH TRAS A PREDIKCÍ

Vytvořený program pro vyhledání objektů ve scéně a nalezení jeho tras a predikcí je realizován v prostředí Microsoft Visual Studio 2005 Professional Edition (v jazyce C++), za použití knihoven pro digitální zpracování obrazu OpevCV v1.0. Pro získání scén byla použita webová kamera Logitech QuickCam Messenger s rozlišením 320x240 pixelů.

Ve scénách se obvykle sledují různé skvrny, barvy, textury nebo určitá místa zájmu jako jsou hrany nebo rohy. Jak již bylo zmíněno v úvodu, program slouží pro zpracování a analýzu syntetických scén. Takové scény jsou poměrně jednoduché a je v nich možné sledovat přímo objekty, které bývají také poměrně jednoduché. Jejich parametry se navíc s časem příliš nemění (velikost, rotace, barva), což je výhodné i pro predikce, kde je velmi důležitá podobnost konkrétního objektu na sérii snímků.

Celý program lze rozdělit do několika funkčních částí viz *Obr.3.* V práci se pak dále zabývám důkladnějším popisem těchto jednotlivých funkčních celků.



Obr. 3 Jednotlivé funkční části programu

8.1 UŽIVATELSKÉ PROSTŘEDÍ

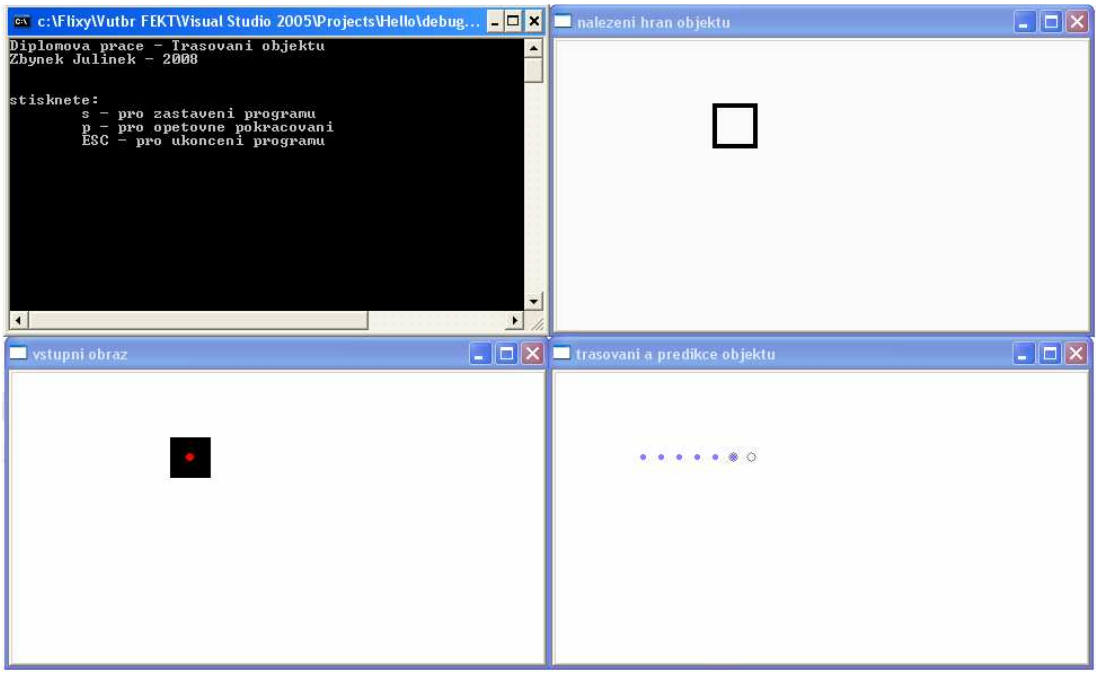
Celý program je vytvořen jako konzolová aplikace v prostředí Microsoft Visual Studio 2005. Na *Obr. 4* je zachycen běh programu, tak jak jej vidí uživatel.

V levém horním rohu je zobrazeno hlavní konzolové okno aplikace, jsou zde vypsány základní příkazy, kterými může uživatel řídit běh programu. Pro jednotlivé příkazy (zastavení programu, opětovné pokračování, ukončení programu) jsou implementovány příslušné klávesové zkratky.

V levém dolním rohu je umístěno okno, kde je zobrazován vstupní obraz. Do tohoto okna je možné dle potřeby také zobrazovat středy nalezených objektů a nebo predikce pro následující krok. Tyto parametry jsou sice zjišťovány v metodách, které jsou implementovány hluboko ve zdroji celého programu, ale může být vhodné porovnat si dosažený výsledek přímo se vstupním obrazem.

V pravém horním rohu se nachází okno, kde se zobrazuje obraz získaný metodami nalezení objektů pomocí hranových detektorů. Toto okno slouží pro kontrolu, jak kvalitní hrany se podařilo získat ze vstupního obrazu.

V pravém dolním rohu je okno, kde se zobrazují trasy a predikce jednotlivých objektů. Plná kolečka slouží k zachycení tras, prázdná kolečka reprezentují predikce.



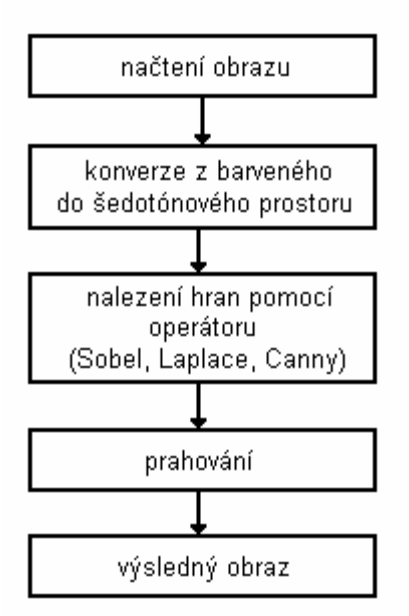
Obr. 4 Uživatelské prostředí programu

8.2 VÝBĚR VHODNÉHO OPERÁTORU PRO DETEKCI HRAN

Pro nalezení objektu ve scéně bylo nejprve nutné nalézt vhodný operátor pro detekci hran v objektu a prověřit jeho kvalitu na typových obrazech, které následně mají sloužit jako vstup pro další metody programu. Především pak zjištění citlivosti daného operátoru na šum obsažený ve snímku.

Jako vhodné operátory jsem zvolil Sobelův a Laplaceův gradientní operátory a také Cannyho hranový detektor. Pro všechny tyto metody jsem využil knihovny OpenCV, ve kterých jsou tyto funkce implementovány.

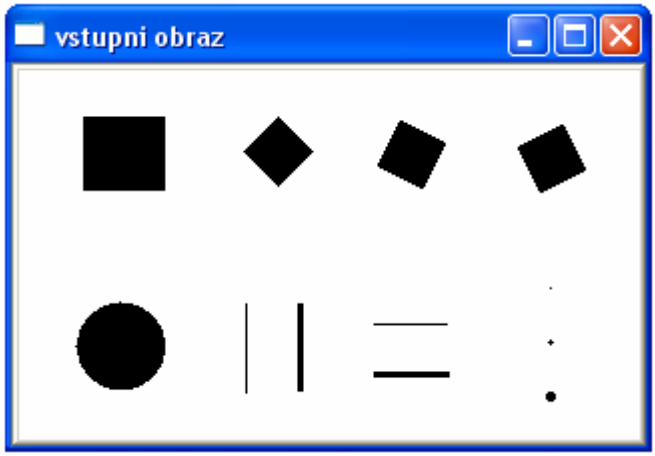
Velikost konvolučních masek všech operátorů jsem zvolil jako matici 3x3. Při pokusech o nastavení masky na velikost 5x5 bylo zatížení CPU tak značné, že se program zaseknul. U Sobelova operátoru bylo navíc nutné nejprve spočítat masky zvlášť pro osy *x,y* a tyto submasky následně sečíst. Cannyho detektor je implementován s hysterezí, jejíž parametry byly testovány s různým nastavením na různých snímcích. Pro testování jsem použil algoritmus viz *Obr.5*.



Obr. 5 Algoritmus pro testování hranových operátorů

Na vzorovém kontrastním snímku pro testování jsou umístěny jednoduché objekty viz *Obr.6*. Postup jsem zopakoval pro snímek ve formátu .bmp, a pro tentýž snímek, ale se zhoršenou kvalitou, ve formátu .jpg.

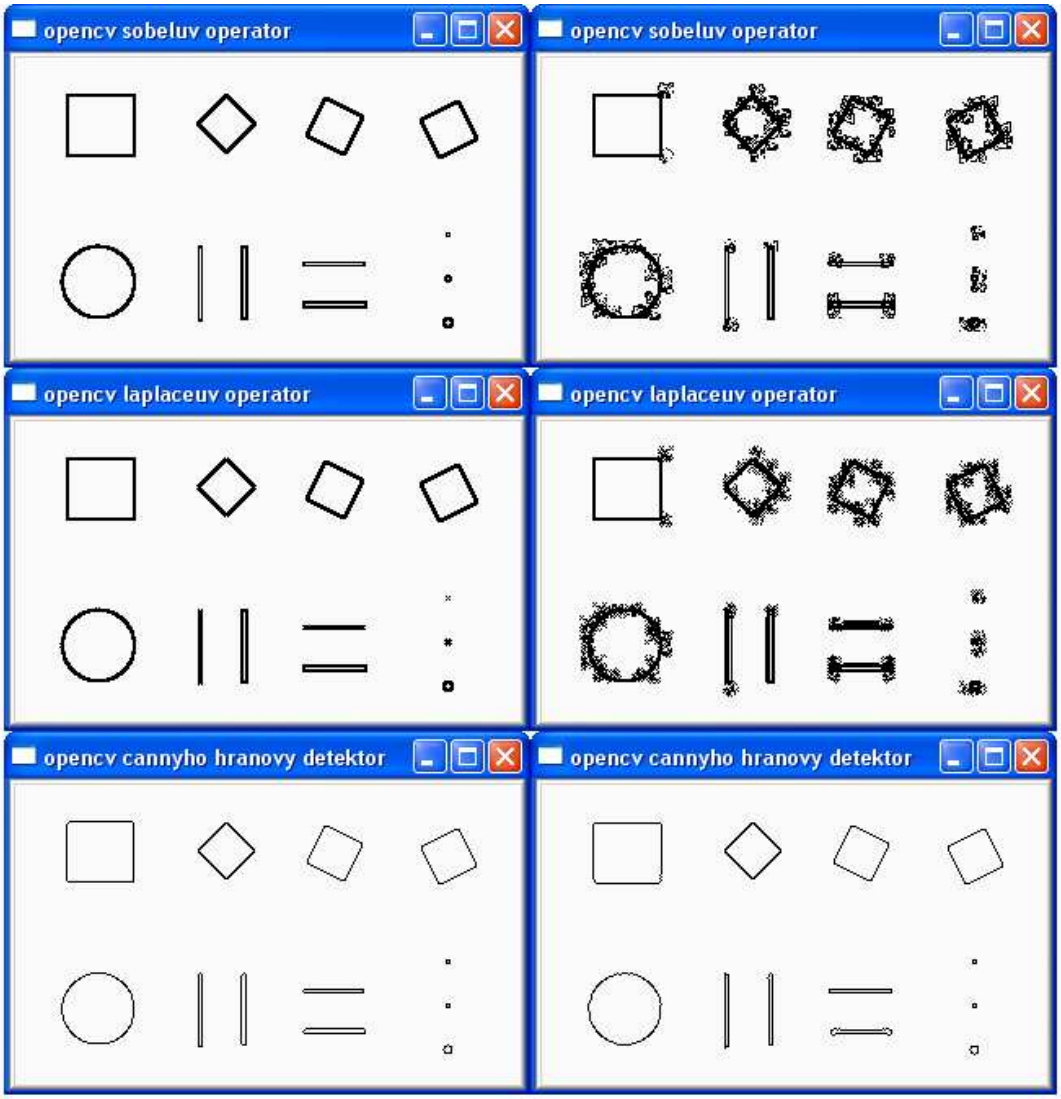
Komprese obrazu z formátu .bmp do .jpg probíhá v podstatě jako segmentace a transformace obrazu do bloků o velikosti 8x8 tak, že hodnoty v těchto blocích resp. maticích korespondují s důležitostí informace v obraze. Z tohoto důvodu na hranách obrazu ve formátu .jpg vznikají nežádoucí artefakty, které svým charakterem připomínají šum.



Obr. 6 Vzorový snímek použitý pro testování jednotlivých operátorů

Z testu viz *Obr.7* vyplývá, že všechny tři operátory kvalitně nacházejí hrany objektů na testovacím snímku ve formátu .bmp.

Na druhou stranu je zřejmé, že všechny operátory jsou citlivé na artefakty vznikající při kompresi do formátu .jpg, přičemž na hranách, které nejsou rovnoběžné s osami *x,y* je tento jev nejvíce patrný. Cannyho hranový detektor se v tomto případě jeví jako nejkvalitnější, protože u něj dochází pouze k velmi malým skreslením. Na druhou stranu je tato lepší kvalita vykoupena výrazně vyššími výpočetními nároky.



Obr. 7 Výsledky detekce hran na testovacím snímku (vlevo pro formát .bmp, vpravo pro formát .jpg)

Na základě výsledků tohoto testu jsem se rozhodl použít formát .bmp pro další části programu, které využívají k testování sekvenci statických obrazů.

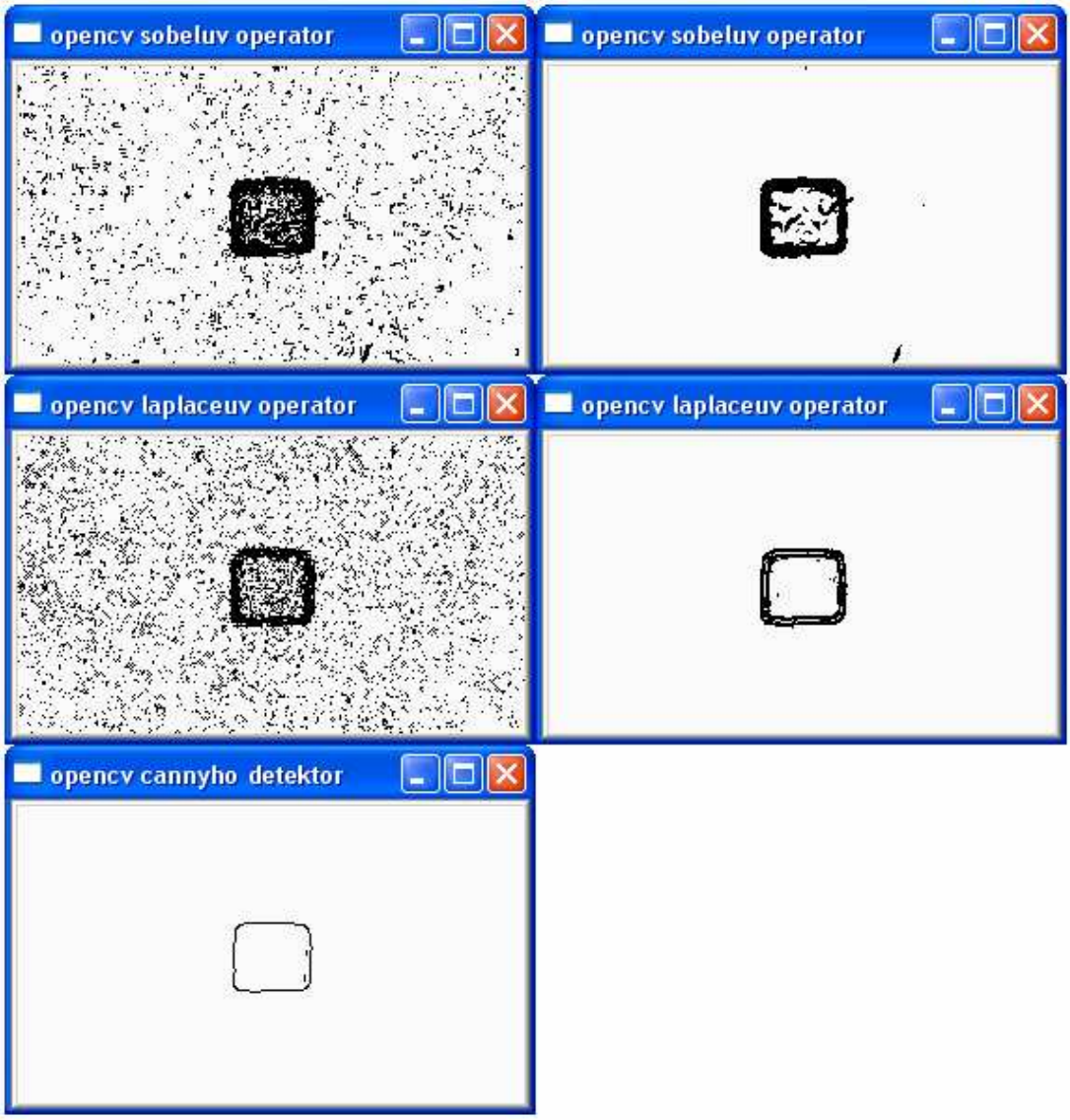
Dále jsem se rozhodl použité operátory otestovat na kontrastním snímku, který je na rozdíl od předchozího pořízen přímo digitální kamerou viz *Obr.8*. Výhodou takto pořízeného snímku je fakt, že lépe odpovídá situaci, kdy bude syntetická scéna snímaná digitální kamerou a dosažené výsledky budou mít větší váhu, než v případě uměle vytvořeného snímku.

Obr. 8 Testovací snímek pořízený digitální kamerou

Výsledky tohoto testu viz *Obr.9 - vlevo* byly překvapivé. Jako jediný uspěl výpočetně nejnáročnější Cannyho hranový detektor. V případě nalezení hran pomocí Laplaceova a Sobelova operátoru je výsledek zatížený značným šumem pocházejícím z okolního prostředí a z vybarvení objektu.

Proto jsem se rozhodl Sobelův a Laplaceův operátor vybavit předzpracováním obrazu v podobě křížového a mediánového filtru, oba o velikosti masky 3x3.

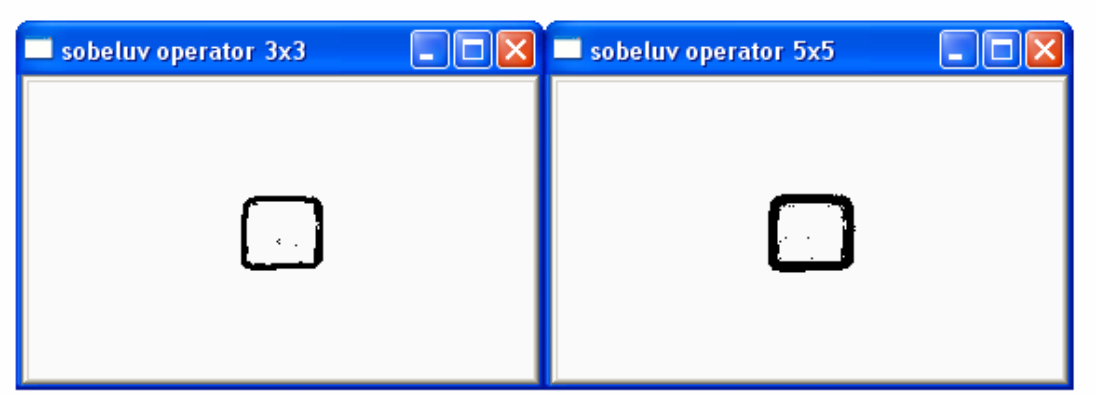
Dosažené výsledky takto upravených operátorů viz *Obr.9 – vpravo* jsou z hlediska množství výsledného šumu lepší, než v předchozím případě, ale nikoliv postačující. Zlepšená kvalita vyhodnocení detekce hran je navíc v tomto případě vykoupena vyššími výpočetními nároky na celý program. Stále je na obraze patrný šum, především v případě Sobelova operátoru, kdy nedošlo k odstranění šumu z povrchu objektu. U Laplaceova operátoru je zřejmé rozdvojení hran, které také není pro další zpracování obrazu žádoucí.



Obr. 9 Výsledky detekce hran na snímku z digitální kamery (vlevo pro operátory bez přídavných filtrů, vpravo s přidanými filtry)

Na základě výsledků tohoto testu jsem se rozhodl manuálně naprogramovat Sobelův operátor a také vyzkoušet různé nastavení velikosti masek (3x3 a 5x5). Z manuálu knihoven OpenCV, totiž není zcela jasné, jak konkrétně probíhá výpočet implementovaných funkcí pro jednotlivé operátory.

Tyto operátory jsem rovnou použil k dalšímu testu detekce hran na obrazu pořízeného digitální kamerou viz *Obr.8*.



Obr. 10 Výsledky detekce hran na snímku z digitální kamery

Velikost masky Sobelova operátoru se, v tomto testu viz *Obr.10*, logicky projevuje na síle detekované hrany, přesto je možné dosažené výsledky obou nastavení (3x3 a 5x5) považovat za totožné a především postačující. Myslím, že silnější hrana (maska 5x5) by mohla být vhodná pro snímky, kde by vlivem osvětlení nebo malého kontrastu mohlo docházet ke ztrácení, nebo rozpadání hran.

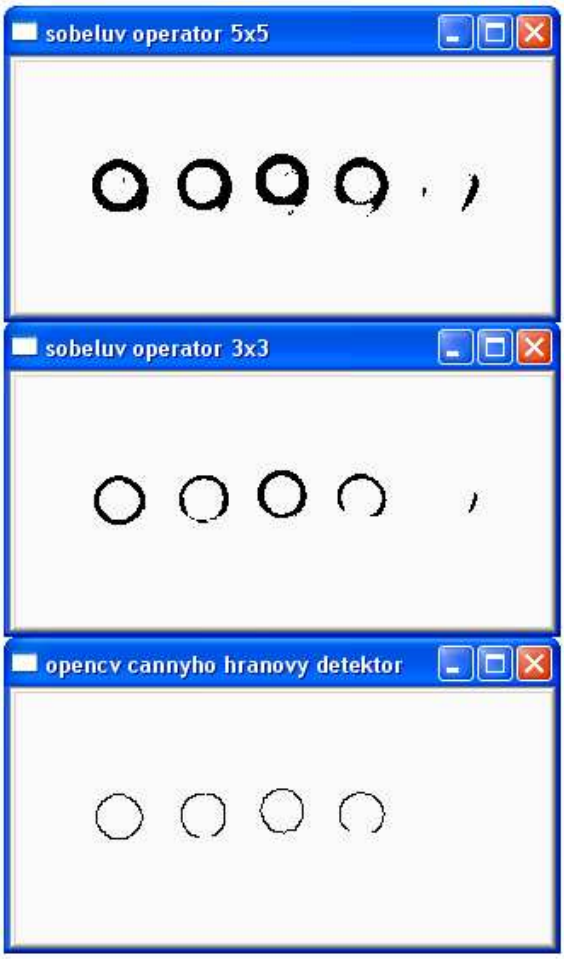
Z tohoto testu je zřejmé, že ručně naprogramovaný Sobelův operátor předčí Sobelův a Laplaceův operátor implementovaný v knihovnách OpenCV a to i v případě, kdy jsou oba tyto operátory vybaveny dalším předzpracováním obrazu v podobě křížového a mediánového filtru.

V rozhodnutí, který operátor nakonec použít jsem provedl ještě jeden test. Na bílé pozadí jsem umístil malé, různě barevné kuličky. Takto vytvořená scéna už velmi dobře reprezentuje syntetické scény, protože se zde nachází jednoduché reálné objekty. Snímek je přiměřeně kontrastní, ale už obsahuje různé barvy a nakonec tyto objekty vrhají stín, který se může také projevit na výsledku detekcí hran. Takto naaranžovanou scénu jsem vyfotil digitální kamerou viz *Obr.11* a podrobil ji detekci hran manuálně naprogramovaným Sobelovým operátorem, a Cannyho hranovým detektorem, jehož princip je implementován v knihovnách OpenCV (zbylé operátory jsem z dalšího testování vyřadil pro jejich nedostatky).

Obr. 11 Snímek pro finální test detekce hran

Z výsledků tohoto finálního testu viz *Obr.12* se potvrzuje důležitost vybrání vhodné velikosti masky Sobelova operátoru. V případě masky 5x5 jsou hrany poměrně silné, ale je také patrné, že operátor detekuje jako hranu i část stínu některých kuliček. V případě masky 3x3 jsou hrany o poznání slabší. V případě modré a červené barvy jsou nalezené hrany v pořádku, ale u zelené a oranžové už dochází k rozpadání hrany, což může způsobit značné chyby v dalším zpracování obrazu. U Cannyho hranového detektoru jsou detekované hrany objektů nejslabší a také se rozpadají. Hrany jsou v tomto případě tak tenké, že k případným nespojitostem a rozpadání hran bude docházet mnohem častěji, než v případě ostatních variant.

Zajímavostí tohoto testu je, že se nepodařilo v žádném z případů nalézt kvalitně hranu žluté kuličky. Je to patrně způsobeno malým kontrastem mezi žlutou barvou a bílým pozadím, kdy zjištěné hodnoty jasu zřejmě nepřesahují minimální hladinu prahování a tudíž jsou považovány za pozadí případně za šum.



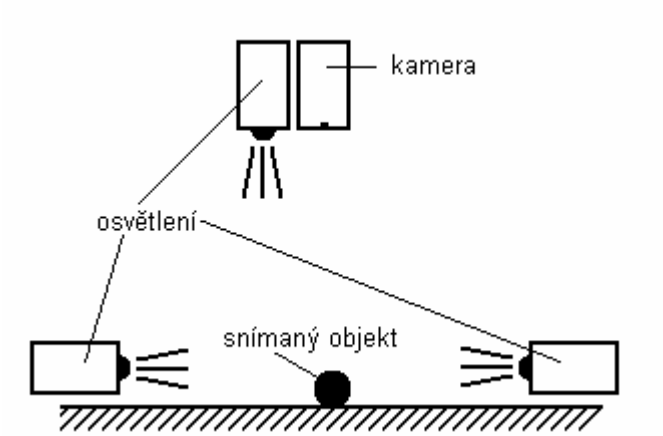
Obr. 12 Výsledky finálního testu detekce hran

Jako nejvhodnější způsob pro nalezení hran v obraze jsem pro další postup zvolil manuálně naprogramovaný Sobelův operátor s velikostí masky 5x5. Z provedených testů vyplývá, že je schopen nejlépe nalézt hrany objektů ve scéně a také, že klade nejmenší nároky na výpočetní náročnost.

Kvalita detekce hrany je na druhou stranu poznamenána tím, že operátor může detekovat stín objektu jako jeho hranu. Objekty, které jsou s pozadím v malém kontrastu nejsou tímto operátorem téměř detekovány. To je způsobeno nastavením parametrů prahování, které byly také testovány. V případě snižování této hranice tak, aby byl detekován i žlutý objekt postupně dochází k větší detekci stínů.

8.3 VYTVOŘENÍ SCÉN

Pro vytvoření syntetických scén byla použita webová kamera. Na *Obr.13* je zobrazen náčrtek rozmístění kamery a osvětlení při získávání scén. V těchto scénách jsou snímány barevné kuličky, které se pohybují po různých trajektoriích. Na těchto scénách jsou demonstrovány dosažené výsledky kvality trasování a predikce pohybu objektů v následujícím kroku.



Obr. 13 Umístění kamery a osvětlení při získávání scén

Takto vytvořené scény považuji za nejvhodnější zobecnění ve smyslu syntetických scén. Jak uvádím v úvodu práce, pokud se na těchto scénách podaří s dostatečnou přesností trasovat a predikovat pohyby objektů, je možné dalším rozšířením algoritmů postihnout i komplikovanější úlohy.

Při získávání scén jsou objekty kromě přirozeného světla navíc vystaveny umělému osvětlení viz *Obr.13*. Při přirozeném osvětlení objekty vrhají stíny, které pak mohou být detekovány jako části objektu viz *Obr.12*. Tato situace je však nežádoucí, a proto jsou použity další tři světelné zdroje rozmístěné tak, aby objekty vrhaly co nejmenší stíny.

Testovací scény byly vytvořeny jako statické obrazy, přičemž jejich seřazení do sekvencí je řešeno softwarově. Testovací scény slouží především k testování kvality predikce, takže obsahují jednoduché objekty (čtverce, kruhy), které se pohybují s různou rychlostí po různých trajektoriích. Mimo jiné tyto scény slouží k testování správné klasifikace objektů a nalezení jejich středů.

8.4 NALEZENÍ OBJEKTŮ A JEJICH STŘEDŮ

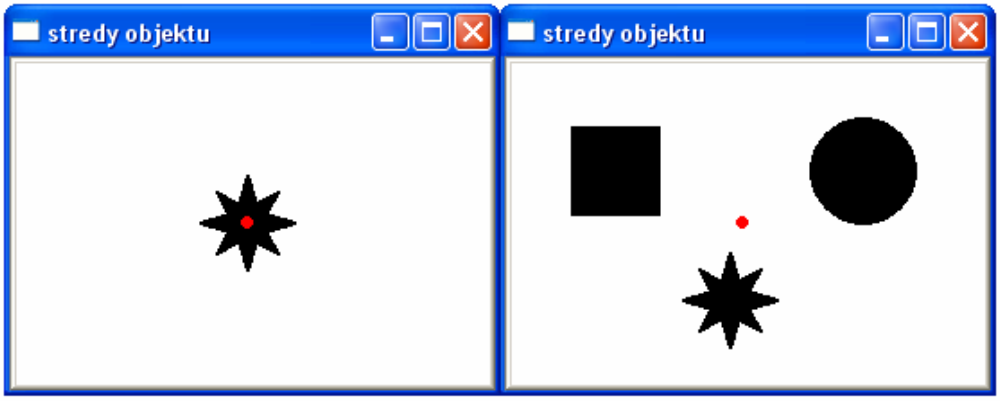
Jako nejvhodnější operátor pro detekci hran byl vybrán Sobelův operátor s velikostí konvoluční masky 5x5. Také jsou k dispozici scény, na kterých můžeme realizovat detekci objektu a nalézt jeho trasu. Algoritmus pro sledování objektu v sérii snímků a nalezení jeho trasy je zobrazen na *Obr.14*.

Obr. 14 Algoritmus pro nalezení objektu a jeho středu

Nejprve je načtena scéna z video souboru, ze kterého je získán aktuální snímek scény, případně jsou načítány jednotlivé snímky z testovacích sekvencí. Následně je provedena konverze do šedotónového prostoru. Poté jsou pomocí Sobelova operátoru nalezeny hrany objektu ve scéně a provedeno prahování.

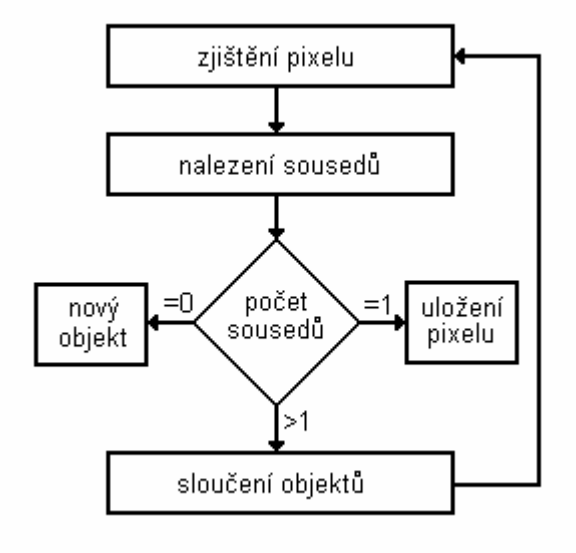
Nalezení středu objektu je prováděno algoritmem, který postupně prochází všechny pixely aktuálního snímku a vyhledává body, které odpovídají objektu. Střed objektu pak reprezentuje průměr souřadnic, všech takto zjištěných bodů.

Takto vytvořený algoritmus však postihuje pouze jeden objekt, jak je vidět na *Obr.15*, kde jsou nalezené středy vykresleny jako červené tečky do vstupního obrazu.



Obr. 15 Nalezení středu objektu průměrem ze všech bodů

Pro úspěšné nalezení více objektů ve scéně je nutné část tohoto algoritmu rozšířit o další mechanismy. Je totiž zřejmé, že nestačí pouze nalézt střed, kterým je průměr všech bodů odpovídající objektům. Rozšířená část algoritmu pro nalezení více objektů je zobrazena na *Obr.16*.



Obr. 16 Rozšířená část algoritmu pro nalezení objektů

Rozšířená část algoritmu začíná podobně jako její předchůdce. Systematicky jsou procházeny všechny pixely v obraze a je zjišťováno, jestli odpovídají objektu. Objektem je myšlen seznam *x,y* souřadnic všech bodů, které tomuto objektu náleží.

Jakmile je takový pixel zjištěn, probíhá hledání počtu jeho sousedů. Sousedem je myšlen bod objektu, který leží vedle jiného detekovaného bodu ve směru osy *x*,*y*.

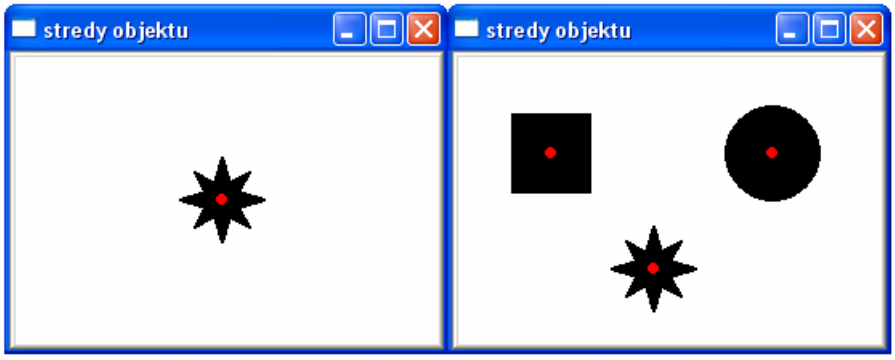
Při hledání počtu sousedů jsou postupně procházeny souřadnice všech nalezených objektů. Pokud žádní takoví sousedi neexistují, je založen nový objekt, kterému je přiřazen testovaný pixel. Pokud existuje jeden soused, znamená to, že zjišťovaný pixel náleží danému objektu a následně dochází k uložení souřadnic k tomuto objektu. Pokud je nalezeno víc sousedů, dochází ke slučování objektů.

V předchozím případě jsme měli na scéně pouze jeden objekt, nebyl tedy problém správně vyhodnotit jak konvexní tak konkávní objekty viz *Obr.15*. V situaci, kdy detekujeme více než jeden objekt, mohou být nekonvexní objekty popsány jako několik jiných objektů. Například hvězda by byla vyhodnocena tak, že pro každý cíp by byl nalezen jeden objekt. Proto je nutné takto nalezené objekty sloučit v případě, že mají více než jednoho souseda.

Sloučení sousedních objektů spočívá v procházení všech objektů, se kterými sousedí právě zpracovávaný objekt. Poté probíhá zkopírování všech bodů souseda do aktuálního objektu a nakonec je celý soused vymazán.

Nakonec jsou vypočteny středy, jako průměry souřadnic, všech takto detekovaných objektů. Aby nedocházelo k detekci možného šumu v obraze jako objektu, jsou za objekty považovány pouze shluky více než desíti pixelů.

Takto upravený algoritmus je tedy schopen rozlišit více objektů, jak je vidět na *Obr.17*, kde jsou nalezené středy vykresleny jako červené tečky do vstupního obrazu.



Obr. 17 Nalezení středů objektů rozšířeným algoritmem

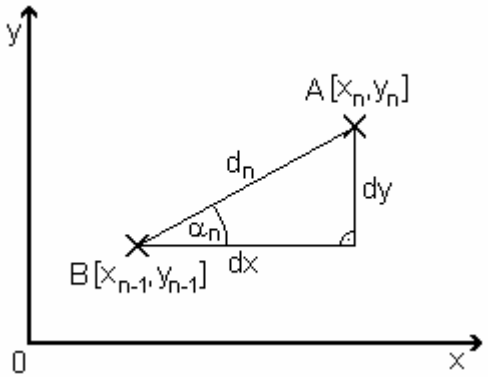
8.5 TRASOVÁNÍ NALEZENÝCH OBJEKTŮ

Pro další zpracování obrazu po nalezení objektů a jejich středů, přichází na řadu zaznamenání trajektorií jednotlivých objektů, které se také označují jako trasy. Jedná se v podstatě o souhrn všech bodů, kterými střed objektu doposud prošel.

Právě zjišťovaný střed je vždy uložen jako poslední prvek do řetězce, který obsahuje všechny zjištěné středy minulých kroků. V tomto řetězci je tedy uložena celá trasa, kterou objekt do této chvíle vykonal. Každý objekt má svůj vlastní soubor těchto prvků resp. každému objektu je přiřazena jeho trasa.

Trasa objektu je pak sledována tečkami, které reprezentují středy sledovaného objektu v jednotlivých krocích. Ty jsou posléze vykresleny do okna *trasování a predikce objektů* viz *Obr.4*.

Kromě tras jsou ještě pro daný objekt ukládány vypočítané vzdálenosti a úhly mezi současnou a minulou polohou viz *Obr.18*. Vzdálenosti *d* jsou počítány ze znalosti vztahů v pravoúhlém trojúhelníku viz rovnice *8.1*. Úhly *α*, pod kterými se objekt pohybuje jsou vypočteny ze znalosti goniometrických funkcí viz rovnice *8.2*.



Obr. 18 Výpočet vzdálenosti a úhlu mezi současnou a předchozí polohou

Současná poloha je na *Obr.18* označena jako bod *A*, předchozí poloha jako bod *B* ($dx=x_n-x_{n-1}$; $dy=y_n-y_{n-1}$). Vzdálenost *d* a úhel *α* mezi těmito body pak odpovídá:

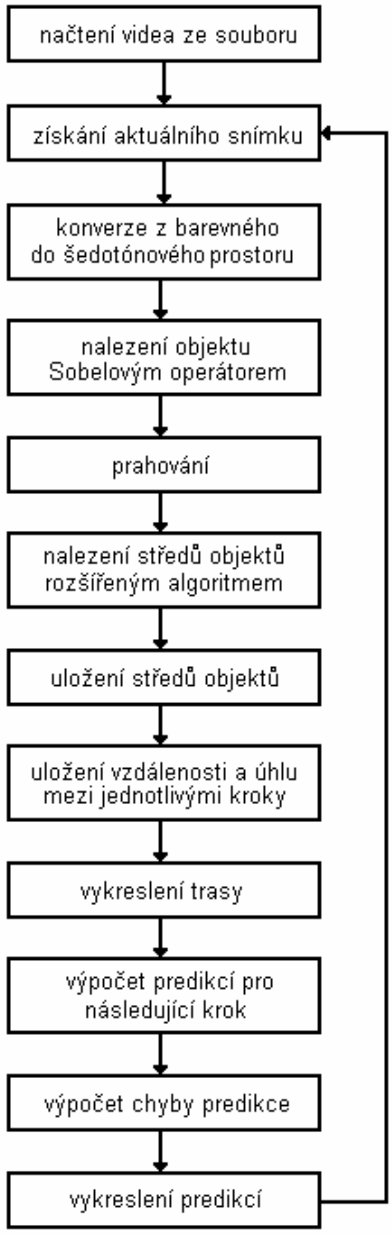
$$d_n = \sqrt{dx^2 + dy^2} \qquad (8.1)$$

$$\alpha_n = \tan(dy / dx) \qquad (8.2)$$

Všechny takto získané informace jsou následně využívány mechanizmy predikce objektů pro následující krok. Těmi se podrobně zabývám v další kapitole.

8.6 PREDIKCE OBJEKTŮ

Predikce objektů v programu slouží k vypočítání odhadu polohy objektů v následujícím kroku. Tyto predikce využívají informací z trasování objektů, především pak vzdálenosti a úhly mezi jednotlivými kroky pohybu objektu a také jeho současnou polohu. Celý algoritmus pro detekci, trasování a predikci objektů je zobrazen na *Obr.19*.



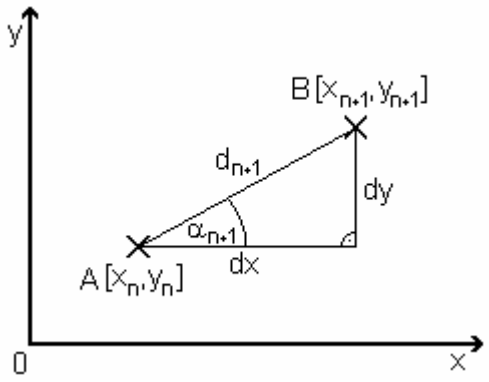
Obr. 19 Kompletní algoritmus pro detekci, trasování a predikci objektů

Pro výpočet predikce jsou navrženy dva algoritmy. Dá se říct, že úspěšnost predikce pohybu objektu pro následující krok, vždy záleží na způsobu pohybu objektu ve scéně. Predikce mívají tím lepší výsledky, čím víc si jsou podobné jednotlivé snímky, mezi kterými se predikce provádí. Zejména je důležitá podobnost objektů, které mohou být podrobeny různé rotaci nebo změně měřítka.

Zjednodušení ve smyslu syntetických scén spočívá v tom, že se objekty pohybují po různých trajektoriích, jejich velikost i natočení vůči kameře zůstává stejné, protože jsou snímány pohybující se barevné kuličky. V tomto případě je možné sledovat objekty na základě jejich trajektorií. Reálné scény bývají pestřejší, protože obsahují více různých objektů a dochází k jejich překrývání. V tomto případě by bylo nutné algoritmy rozšířit o metody rozlišení objektů podle tvarů, popř. barev.

V případě takových objektů, které dokáží zrychlit nebo zpomalit natolik, že se mohou v dalším kroku nacházet kdekoliv na scéně, podobně v případě tak rychlé změny směru, která má podobný následek, není predikce pro velkou dynamiku objektu možná. Trajektorie pohybu takovýchto objektů se dá považovat za náhodnou, a je proto nemožné ji predikčními algoritmy vystihnout.

Oba predikční algoritmy se tedy snaží co nejlépe odhadnout budoucí krok trajektorie objektu viz *Obr.20*. Z vypočtené vzdálenosti d a úhlu α mezi jednotlivými kroky jsou těmito algoritmy vypočteny predikce těchto parametrů (d_{n+1},α_{n+1}). Následně jsou ze znalosti goniometrických funkcí vypočteny souřadnice budoucího kroku (x_{n+1},y_{n+1}) viz rovnice *8.3, 8.4*.



Obr. 20 Výpočet predikce polohy objektu z predikované vzdálenosti a úhlu

Současná poloha je na *Obr.20* označena jako bod *A*, budoucí (predikovaná) poloha jako bod *B*. Souřadnice budoucího kroku x_{n+1}, y_{n+1} pak odpovídá:

$$\begin{aligned} dx &= d_{n+1} \cdot \sin(\alpha_{n+1}) \\ dy &= d_{n+1} \cdot \cos(\alpha_{n+1}) \end{aligned} \tag{8.3}$$

$$\begin{aligned} x_{n+1} &= x_n + dx \\ y_{n+1} &= y_n + dy \end{aligned} \tag{8.4}$$

Predikční algoritmy lze pojmenovat podle způsobu výpočtu jako:

- Plovoucí průměr s exponenciálním zapomínáním
- Statistické vyhodnocení a korekce absolutní chyby

Plovoucí průměr s exponenciálním zapomínáním

Jedná se o klasické počítání plovoucího průměru. Rozšíření v podobě exponenciálního zapomínání spočívá v tom, že váha starších kroků v průměrovacím okně exponenciálně klesá viz rovnice *8.5*, kde *k* je velikost průměrovacího okna.

$$\begin{aligned} d_{n+1} &= \sum_{i=1}^{k} \frac{1}{i} d_{n+1-i} \\ \alpha_{n+1} &= \sum_{i=1}^{k} \frac{1}{i} \alpha_{n+1-i} \end{aligned} \tag{8.5}$$

Statistické vyhodnocení a korekce absolutní chyby

Základem této metody je předpoklad, že se objekt pohybuje rovnoměrným přímočarým pohybem. Při změně rychlosti nebo směru pohybu objektu dochází k odchylce (e_d,e_α), která je rovna změně těchto parametrů mezi současným a předchozím krokem. Korekce výsledku pak spočívá v sečtení predikce a zjištěné odchylky viz rovnice *8.6*.

$$\begin{aligned} d_{n+1} &= d_n + e_d \\ \alpha_{n+1} &= \alpha_n + e_\alpha \end{aligned} \tag{8.6}$$

Pro testování úspěšnosti predikce obou těchto predikčních algoritmů slouží testovací scény. Ty obsahují jednoduché objekty, které se pohybují různou rychlostí po různých trajektoriích. Tyto scény jsou realizovány jako statické obrazy a jejich seřazení do sekvencí je řešeno softwarově. Přehled jednotlivých testů a zjištěných chyb je uveden v následující kapitole.

8.7 TESTY PREDIKČNÍCH ALGORITMŮ

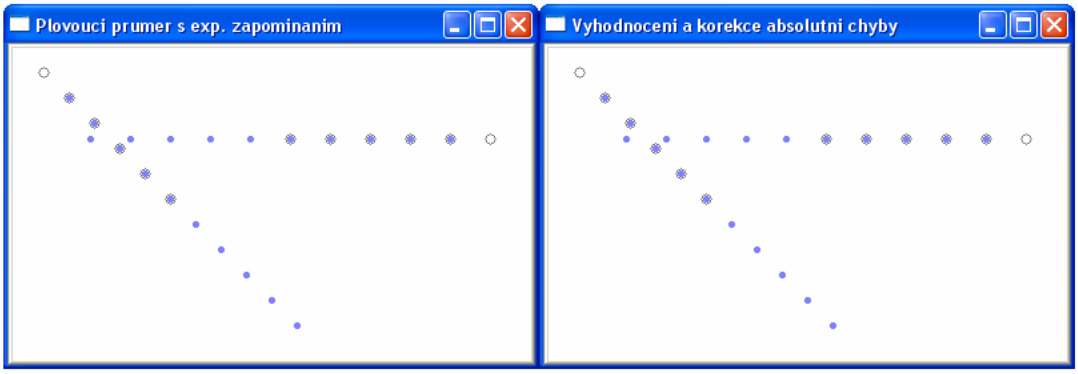
Pro zobrazení testů predikčních algoritmů slouží okno *trasování a predikce objektů* viz *Obr.4*. Plná kolečka zde reprezentují trasy jednotlivých objektů, prázdná kolečka odpovídají predikcím následujícího kroku.

Vypočtené chyby obou predikčních algoritmů pro jednotlivé testy jsou ukládány do externího souboru a pro přehlednost jsou uvedeny až na konci kapitoly. Konkrétně se jedná o absolutní a relativní chyby vzdálenosti d a rozdílu úhlů α mezi jednotlivými kroky. Protože jsou tyto chyby počítány pro každou trajektorii ve všech krocích, uvádím pouze minimální, maximální a průměrnou hodnotu těchto chyb.

Predikce v těchto testech jsou poprvé vypočítány až v pátém kroku. K výpočtu vzdálenosti a úhlu jsou nutné dva kroky, mezi kterými se tyto parametry počítají. K výpočtu rozdílu dvou úhlů, pomocí kterých algoritmy určují natočení objektu mezi jednotlivými kroky, je nutný další krok. Velikost průměrovacího okna v plovoucím průměru je rovna třem. Oba algoritmy predikují od stejného kroku, aby bylo k dispozici stejné množství hodnot k porovnání kvality predikce, v součtu tedy predikce začíná v obou případech až v pátém kroku.

Test č.1

V tomto testu se na scéně pohybují dva objekty přímočarým rovnoměrným pohybem. Výsledek testu je zobrazen na *Obr.21*.

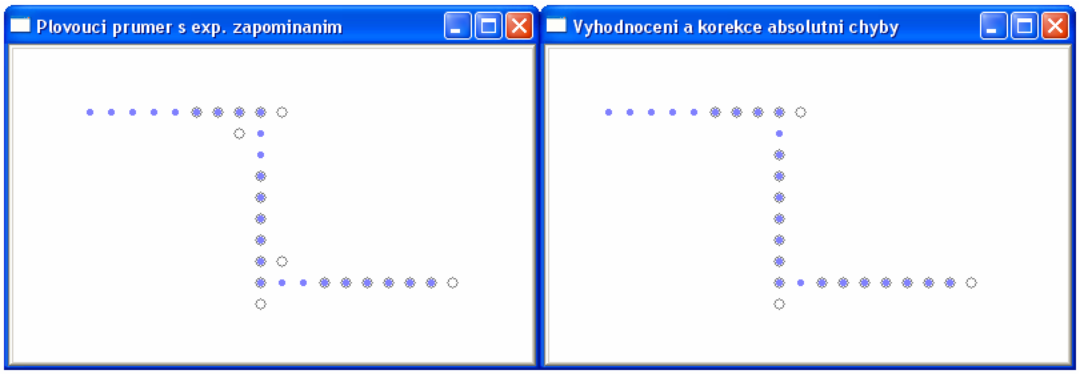


Obr. 21 Výsledky testu predikčních algoritmů č.1

Jak je z výsledků testu patrné, nedochází zde k žádné chybě. To je způsobeno tím, že jsou vzdálenosti i úhly mezi jednotlivými kroky konstantní.

Test č.2

V tomto testu se na scéně pohybuje jeden objekt rovnoměrným pohybem, přičemž dvakrát změní nečekaně směr. Výsledek testu je zobrazen na *Obr.22*.

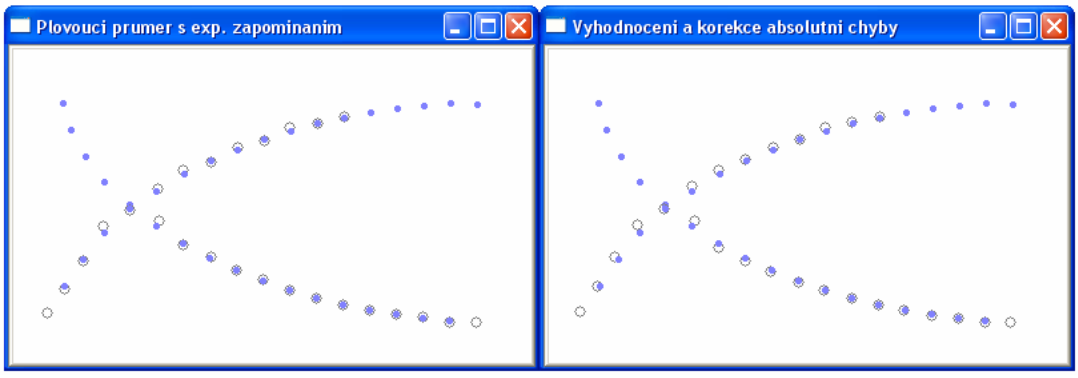


Obr. 22 Výsledky testu predikčních algoritmů č.2

Zde je situace podobná jako v předchozím případě. Zásadním rozdílem je změna trajektorie, kdy je patrné, že algoritmus plovoucího průměru s exponenciálním zapomínáním potřebuje k postihnutí této změny o jeden krok více, než algoritmus statistického vyhodnocení a korekce absolutní chyby, která úspěšně predikuje už v prvním kroku po změně pohybu.

Test č.3

V tomto testu se na scéně pohybují dva objekty proměnlivou rychlostí po trajektoriích s proměnlivým úhlem. Výsledek testu je zobrazen na *Obr.23*.

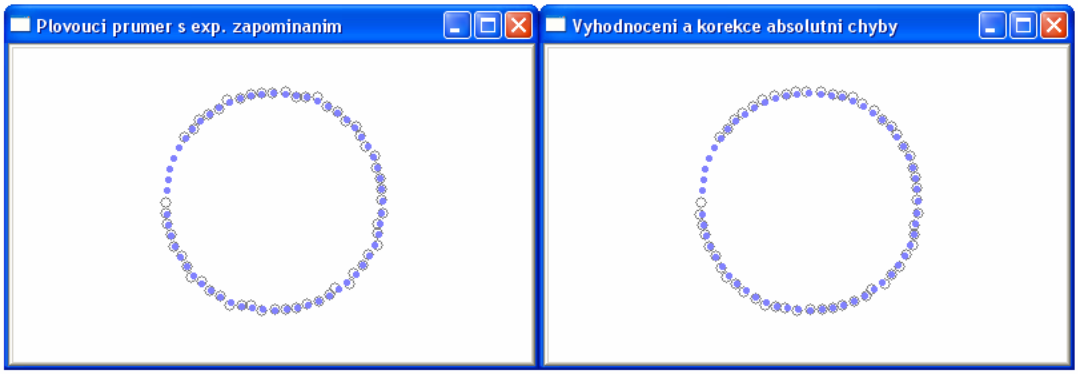


Obr. 23 Výsledky testu predikčních algoritmů č.3

Z výsledků tohoto testu je patrné, že oba predikční algoritmy pracují s jistou chybou v případě, že se objekt pohybuje proměnlivou rychlostí po trajektorii s proměnlivým úhlem. Tato trajektorie svým tvarem připomíná část elipsy.

Test č.4

V tomto testu se na scéně pohybuje objekt s proměnlivou rychlostí po kruhové trajektorii. Výsledek testu je zobrazen na *Obr.24*.

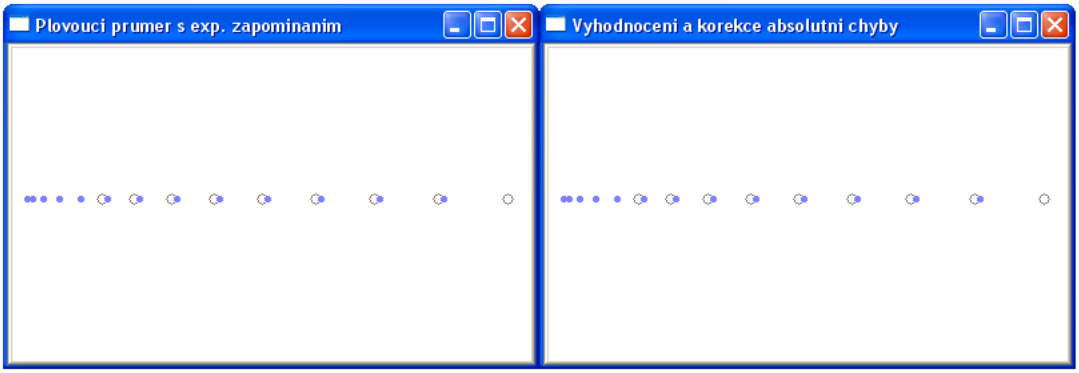


Obr. 24 Výsledky testu predikčních algoritmů č.4

Jednotlivé vzdálenosti a změny úhlů mezi jednotlivými kroky nejsou v tomto případě stejné, podobně jako v předchozím případě. Z tohoto důvodu není predikce zcela přesná a opět dochází k chybě predikce obou algoritmů.

Test č.5

V tomto testu se na scéně pohybuje jeden objekt zrychleným přímočarým pohybem. Výsledek testu je zobrazen na *Obr.25*.



Obr. 25 Výsledky testu predikčních algoritmů č.5

Z výsledků tohoto testu je zřejmé, že ani jeden z použitých predikčních algoritmů není schopen zcela přesně postihnout zrychlený pohyb. Vzhledem k tomu, že se jedná o přímočarý pohyb, nedochází k žádné chybě při predikci úhlu.

Chyby predikčních algoritmů na testovacích scénách

Jak je vysvětleno výše, uvádím pro oba predikční algoritmy pouze minimální, maximální a průměrnou chybu jednotlivých testů. V prvním a třetím testu se sice pohybují dva objekty, ale oba popisují typově stejnou trajektorii, proto jsou jejich výsledky sloučeny tak, jako by se jednalo o jediný objekt. Výsledky jednotlivých testů jsou zobrazeny v tabulkách na *Obr.26* a *Obr.27*.

	Plovoucí průměr s exponenciálním zapomínáním														
test č.	**1**			**2**			**3**			**4**			**5**		
chyby	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr
Δd [-]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.34	0.63	0.00	1.41	0.34	4.00	4.00	4.00
Δα [-]	0.00	0.00	0.00	0.00	90.00	18.00	0.04	9.16	2.91	0.00	13.86	5.56	0.00	0.00	0.00
δd [%]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	9.46	2.67	0.00	18.04	4.00	8.33	20.00	12.75
δα [%]	0.00	0.00	0.00	0.00	100.00	20.00	0.02	32.58	4.97	0.00	27.94	5.38	0.00	0.00	0.00

Obr. 26 Chyby predikce plovoucího průměru s exponenciálním zapomínáním

	Statistické vyhodnocení a korekce absolutní chyby														
test č.	**1**			**2**			**3**			**4**			**5**		
chyby	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr
Δd [-]	0,00	0,00	0,00	0,00	0,00	0,00	0,00	2,34	0,60	0,00	1,41	0,34	4,00	4,00	4,00
Δα [-]	0,00	0,00	0,00	0,00	90,00	9,00	0,00	10,43	3,35	0,00	13,86	5,45	0,00	0,00	0,00
δd [%]	0,00	0,00	0,00	0,00	0,00	0,00	0,00	9,46	2,59	0,00	18,04	4,00	8,33	20,00	12,75
δα [%]	0,00	0,00	0,00	0,00	100,00	10,00	0,00	32,58	9,88	0,00	25,25	4,88	0,00	0,00	0,00

Obr. 27 Chyby predikce statistického vyhodnocení a korekce absolutní chyby

Největší chyba je zjištěna u *testu č.2* (*δα = 100%*), kde objekt nečekaně změnil svoji dráhu, což nebylo možné předvídat. Průměrné hodnoty (*δα*, *Δα*) jsou u tohoto testu, v případě druhého algoritmu, poloviční (*δα = 9.00%*, *Δα = 10.00%*), protože správně predikuje už v prvním kroku po změně pohybu viz *Obr.22*.

Podobná situace se opakuje u *testu č.3*. Maximální chyby (*δα = 32.58%*) jsou u obou algoritmů totožné, ale celkově predikuje lépe první algoritmus, protože průměrná hodnota (*δα*) je poloviční. Maximální chyba je zde způsobena náhlým zvětšením úhlu, což je patrné z *Obr. 23*.

U *testu č.1* jsou chyby nulové. V případě *testu č.5* se chyba (*δd*) pohybuje v rozmezí *8.33-20.00%*, což potvrzuje, že oba algoritmy nejsou schopny predikovat zrychlený pohyb. Průměrná chyba je u obou testů totožná (*δd = 12.75%*).

Výsledky *testu č.4* ukazují, že se jednotlivé relativní chyby (δd, $\delta\alpha$) pohybují v poměrně velkých rozmezích ($\delta d \in 0.00\text{-}18.04\%$, $\delta\alpha \in 0.00\text{-}27.94\%$). Je nutné si uvědomit, že vzdálenost mezi jednotlivými kroky je maximálně $d = 10$ *pixelů*. To při nerovnoměrném nepřímočarém pohybu může snadno způsobit absolutní chybu predikce, která pak odpovídá $\Delta d = 1$*pixel*. Ačkoliv je možné považovat chybu jednoho pixelu za zanedbatelnou, zjištěná relativní chyba pak odpovídá $\delta d = 10\%$. Obdobná situace nastává v případě náhlé větší změny úhlu.

Ze zobrazených výsledků chyb je patrné, že oba predikční algoritmy pracují, až na uvedené vyjímky, s podobnou přesností. Predikční algoritmus statistického vyhodnocení a korekce absolutní chyby vykazuje nepatrně lepší výsledky (v řádech desetin procent u průměrných relativních chyb).

Dále je patrné, že zjištěné chyby potvrzují předpoklady, které vyplývají z vyhodnocení jednotlivých testů. S ohledem na všechny zmíněné okolnosti myslím, že oba predikční algoritmy pracují s velmi dobrou přesností.

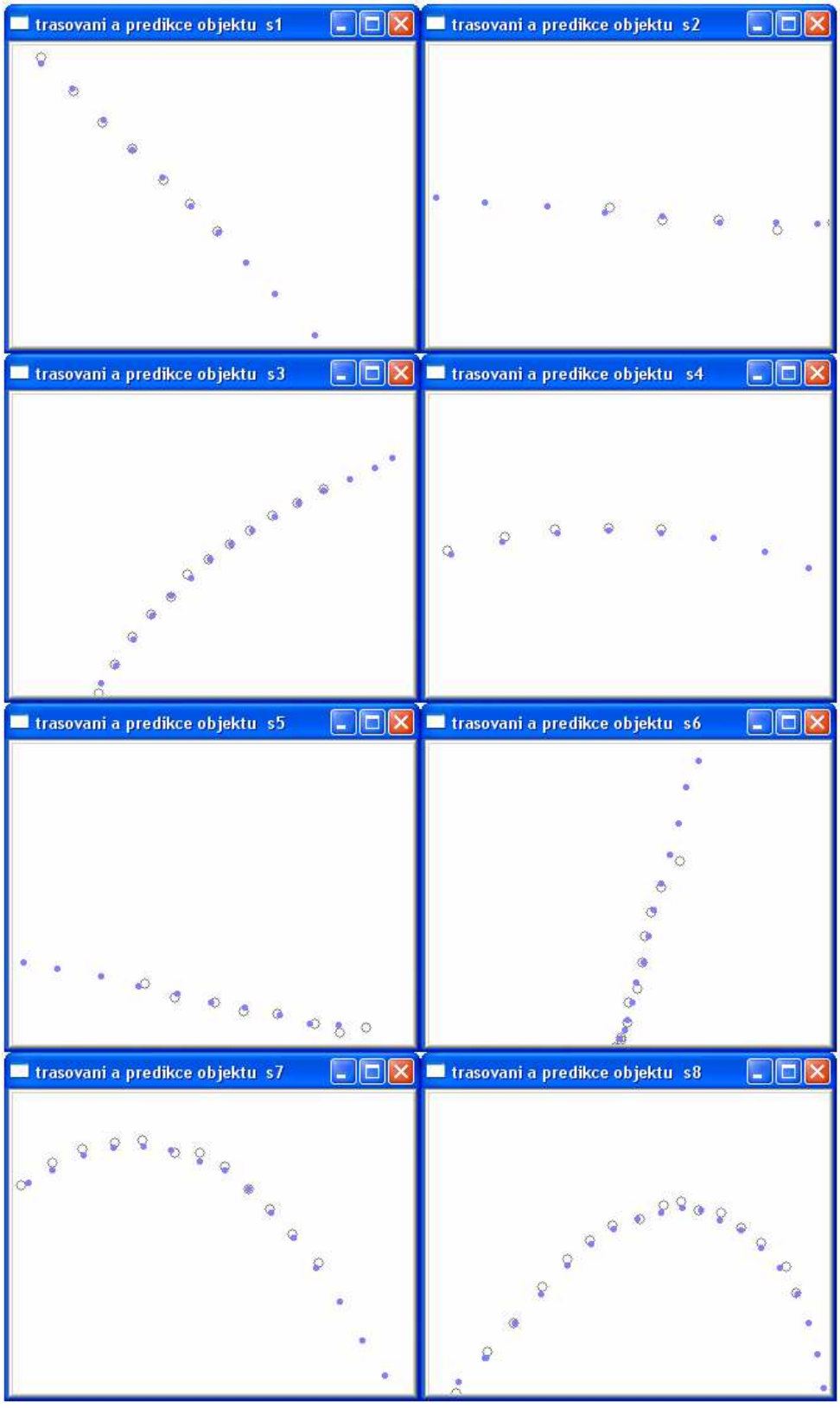
8.8 TRASOVÁNÍ A PREDIKCE OBJEKTŮ NA SYNTETICKÝCH SCÉNÁCH

V této kapitole je uveden přehled jednotlivých syntetických scén a také jsou zde popsány dosažené výsledky predikcí na těchto scénách. Postup získání těchto scén je uveden v *kapitole 8.3*.

Scény jsou digitální kamerou snímány tak, že zachycují postupně několik pohybujících se objektů. Pro přehlednější zobrazení výsledku, byly tyto scény nastříhány v programu Virtual Dub v1.8 do úseků, které obsahují jeden objekt viz *Obr.28*.

Jako vhodnější algoritmus pro predikce na takových scénách volím algoritmus statistického vyhodnocení a korekce absolutní chyby zejména proto, že je schopen lépe vystihnout proměnlivý pohyb.

První predikce polohy objektu je poprvé vypočtena v třetím kroku. Podobně jako v *kapitole 8.7*, jsou k výpočtu vzdálenosti a úhlu zapotřebí dva kroky, mezi kterými se tyto parametry počítají. K výpočtu rozdílu dvou úhlů, pomocí kterých algoritmy určí natočení objektu, je pak nutný třetí krok.



Obr. 28 Přehled tras objektů na syntetických scénách

Na *Obr.28* jsou zachyceny jednotlivé trasy malých kuliček, které se pohybující proměnlivou rychlostí po různých trajektoriích s proměnlivým úhlem (označení *s1-s8*).

Vypočtené chyby predikce v jednotlivých reálných syntetických scénách jsou zobrazeny v tabulce na *Obr.29*. Opět se jedná o absolutní a relativní chyby vzdálenosti *d* a rozdílu úhlu *α* mezi jednotlivými kroky. Ze stejných důvodů, jako v *kapitole 8.7*, uvádím pouze minimální, maximální a průměrnou hodnotu těchto chyb.

scéna č.	1			2			3			4		
chyby	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr
Δd [-]	0,03	3,34	1,69	0,17	11,98	3,48	0,05	6,80	1,28	0,14	3,50	1,45
Δα [-]	1,22	6,34	2,90	2,47	5,19	3,01	1,36	9,78	4,34	2,71	6,25	4,67
δd [%]	0,09	10,43	5,21	0,37	8,25	1,95	0,22	36,88	6,21	0,34	7,87	3,37
δα [%]	0,90	4,49	2,12	0,69	1,47	0,85	0,64	4,35	1,96	1,53	3,30	2,54

scéna č.	5			6			7			8		
chyby	min	max	průměr	min	max	průměr	min	max	průměr	min	max	průměr
Δd [-]	0,53	4,46	2,32	0,00	6,31	2,15	0,72	6,26	2,70	0,00	5,65	2,20
Δα [-]	3,58	13,77	5,89	0,00	17,73	6,78	1,46	19,00	7,87	0,00	20,84	8,37
δd [%]	1,67	14,54	8,46	0,00	25,10	8,64	2,81	29,16	10,93	0,00	18,14	9,51
δα [%]	1,04	3,85	1,68	0,00	6,97	2,70	1,02	11,82	4,79	0,00	11,88	4,93

Obr. 29 Chyby predikce na reálných syntetických scénách

Na všech reálných scénách dochází k určitým chybám. Maximální relativní chyba predikce vzdálenosti *d*, je zjištěna na *scéně č.2* (*δd = 36.88%*). Maximální relativní chyba predikce rozdílu úhlů *α* mezi jednotlivými kroky, je zjištěna na *scéně č.7* (*δα = 11.82%*). Průměrné hodnoty těchto predikcí nepřekračují u vzdálenosti *d* hranici *10%*, u rozdílu úhlů *α* pak nepřekračují hranici *5%*.

Zjištěné chyby predikce na reálných scénách potvrzují předpoklady, které vyplývají z vyhodnocení jednotlivých testovacích scén viz *Obr.27*. Účelem testovacích scén přitom bylo nastínit možné komplikace při predikci. Proto jsou zjištěné chyby na syntetických scénách cca dvakrát menší, než v případě chyb u testovacích scén.

Na základě dosažených výsledků se domnívám, že celý způsob řešení nalezení objektů, jejich tras a predikcí na syntetických scénách pracuje s velmi dobrou přesností.

ZÁVĚR

Účelem této práce bylo vytvořit vhodné algoritmy pro nalezení a predikci stejných objektů na sérii snímků a stanovit podmínky, za kterých má takováto predikce smysl.

V práci postupně popisuji jednotlivé funkční části navrženého programu. Jako vhodný operátor pro nalezení hran objektů byl vybrán Sobelův operátor s velikostí konvoluční masky 5x5. Dále uvádím popis samotného řešení nalezení trasy a predikci stejných objektů na sérii snímků. Nakonec jsou shrnuty dosažené výsledky nalezení, trasování a predikce objektů na syntetických scénách.

Testovací scény jsou tvořeny jako statické obrazy, přičemž jejich seřazení do sekvencí je řešeno softwarově. Tyto scény slouží k testování kvality predikce, takže obsahují jednoduché objekty, které se pohybují různou rychlostí po různých trajektoriích.

Digitální kamerou jsou snímány syntetické scény, na kterých se pohybují barevné kuličky po různých trajektoriích. Na těchto scénách jsou demonstrovány dosažené výsledky kvality trasování a predikce pohybu objektů.

Zjednodušení ve smyslu syntetických scén spočívá v tom, že velikost objektů a jejich natočení vůči kameře zůstává stejné. V tomto případě je možné sledovat objekty na základě jejich trajektorií. Reálné scény však bývají pestřejší, protože obsahují více různých objektů a může docházet k jejich překrývání. V tomto případě by bylo nutné algoritmy rozšířit o metody rozlišení objektů podle tvarů, popř. barev.

Tato práce se tedy zabývá výhradně syntetickými scénami, které lze chápat jako zobecnění světa. S průměrnou přesností predikce vzdálenosti $\delta d<5\%$ a úhlu $\delta\alpha<10\%$, se mi podařilo vyřešit predikce objektů pro následující krok na těchto scénách.

Dalším rozšířením algoritmů by tedy bylo možné postihnout i komplikovanější úlohy. Po úpravě programu tak, aby detekoval objekty jejichž povrch připomíná texturu, a aby byl schopen tyto objekty rozlišit na pozadí, které může obsahovat různé čáry nebo značky, by bylo možné program použít např. pro predikci pohybu jednotlivých hráčů soupeřova týmu u robosocceru.

POUŽITÁ LITERATURA

[1] Žára J., Beneš B.: Moderní počítačová grafika, Computer Press, Praha 1998

[2] Hlaváč V., Šonka M.: Počítačové vidění, Grada, Praha 1992

[3] Sojka E.: Digitální zpracování a analýza obrazů, VŠB, Ostrava 2000

[4] Petyovský P.: Reprezentace a vlastnosti počítačových dat, přednáška kurzu MPOV, VUT Brno 2006

[5] Zpracování obrazu a jeho statistická analýza - http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi

[6] Computer Vision online - soubor odkazů a článků - http://homepages.inf.ed.ac.uk/rbf/CVonline/

[7] Introduction to programing with OpenCV - http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

[8] OpenCV reference manual - http://www.comp.leeds.ac.uk/vision/opencv/opencvref_cv.html

[9] C++ library reference - http://www.cplusplus.com/reference/