Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Demonstrační aplikace klíčování videa

Plzeň, 2007 Václav Bystřický

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne …….. Václav Bystřický……..

Demonstrational application of Blue screen matting

Blue screen matting has become widely used technique in TV and film production in last decades. Main task of this work is to create an application, which demonstrates this process to laic public. The work contains basic overview of methods commonly used for blue screen matting. Color Difference Matting invented by Petros Vlahos was chosen for final implantation. Secondary task of this work is to implement point tracking algorithm suitable for creation of various visual effects. Some existing solutions of this problem are introduced. Simple thresholding algorithm is used for light saber effect is implementation.

Integral part of this work is also a list of hardware components, which are necessary for this application.

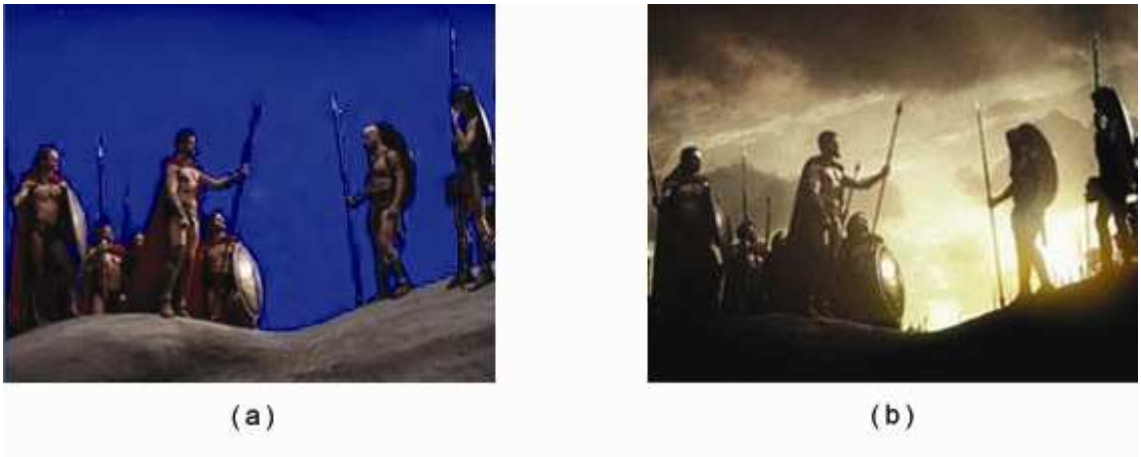
OBSAH:

1 Úvod

Maskování na modré pozadí je jedna z nejrozšířenějších technik ve filmovém a televizním průmyslu. Použití maskování na modré pozadí dovoluje režisérům umístit své postavy do v podstatě libovolného prostoru (viz obr.1) a vytvořit tak ruku v ruce se stále se rozvíjejícími se možnostmi počítačové animace projekty, které by dříve byly budˇ finančně extrémně náročné, nebo zcela nerealizovatelné. Televizní průmysl zase velmi často využívá maskování na modré pozadí v reálném čase pro vytvoření vizuálně působivých dynamických pozadí pro celou řadu pořadů (viz obr.2).

Přestože mnoho lidí se s výstupy, které jsou získány pomocí maskování na modrém pozadí, setkává na televizních obrazovkách a plátnech kin prakticky denně, většina z nich neví na jakém principu tento proces pracuje a jak studio zbývající se tímto procesem vypadá. Mým cílem bylo tedy vytvořit aplikaci, která by laické veřejnosti představila proces maskování na modré pozadí a umožnila i nezkušenému uživateli si proces maskování vyzkoušet. Součástí práce bylo i navrhnout vhodné hardwarové prostředky pro realizaci studia, ve kterém by bylo možné námi vytvořený software prezentovat. Při tomto návrhu byl kladen zvláštní důraz na cenu jednotlivých částí tak, aby byl celý systém dobře použitelný a zároveň byla jeho cena pokud možno minimální.

Pro zvýšení atraktivity pro potenciálního uživatele jsem se rozhodl implementovat nějaký jednoduchý vizuální efekt, jehož výstup by mohl uživatel ovlivnit svým pohybem ve scéně. Nakonec jsem se rozhodl vytvořit efekt světelného meče, notorický známý ze série filmů Hvězdné války George Lucase.



Obr. 1: Ukázka použití maskování na modré pozadí: (a) herci před modrým pozadím, (b) herci před virtuálním pozadím (zdroj <http://photizo.zaadz.com/blog>, upraveno)

Obr. 2: Ukázka použití maskování na modré pozadí: (a) ukázka virtuálního studia (počasí), (b)) ukázka virtuálního studia (diskusní pořad) (zdroj <http://blok.rozanek.cz/?p=1782>, upraveno)

2 Maskování a skládání

Maskování a skládání obrazu jsou jedny ze základních grafických operací. Proces maskování spočívá ve vyjmutí libovolného elementu z obrazu. V procesu skládání je pak tento element spojen s novým obrazem. V následujících dvou částech jsou definovány základní vztahy pro tyto operace.

2.1 Skládání

Skládání je proces při kterém je ze dvou vstupních obrazů nazývaných popředí a pozadí a masek určujících jejich průhlednost složen výsledný obraz zvaný kompozit. Tento proces lze popsat následujícími rovnicemi platnými pro jednotlivé pixely obrazů (převzato z [7]):

$$C' = \mathrm{F}' + (1-\alpha_f)B' \quad (2.1)$$

$$\mathrm{F}' = F\ \alpha_f$$

$$B' = B\ \alpha_b$$

$$\alpha_c = \alpha_f + (1-\alpha_f)\alpha_b$$

$$C' = \mathrm{C}\ \alpha_c$$

Kde C označuje pixel kompozitu, F pixel popředí, B pixel pozadí a α_c, α_f, α_b označují průhlednosti těchto pixelů. V následujících částech budu uvažovat pouze

skládání částečně průhledného popředí s neprůhledným pozadím. Rovnice skládání (2.1) pro tento případ přejde do tvaru

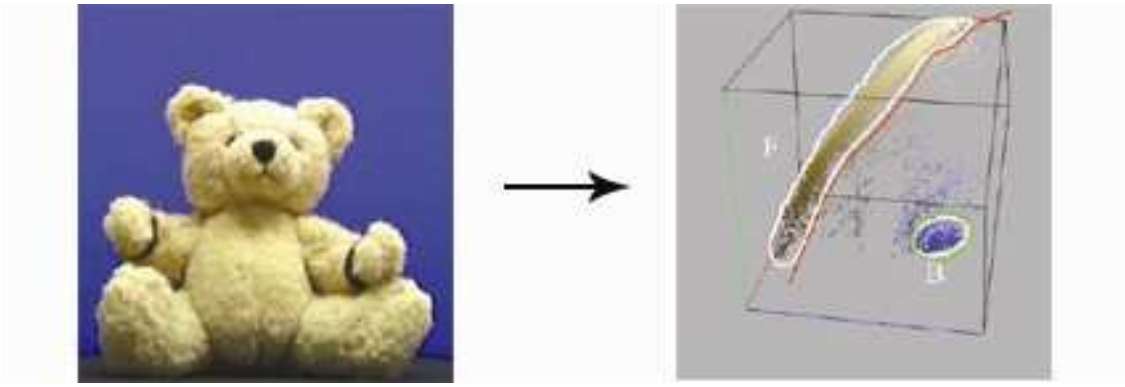
$$C = \alpha_f F + (1-\alpha_f)B \quad (2.2)$$

2.2 Maskování

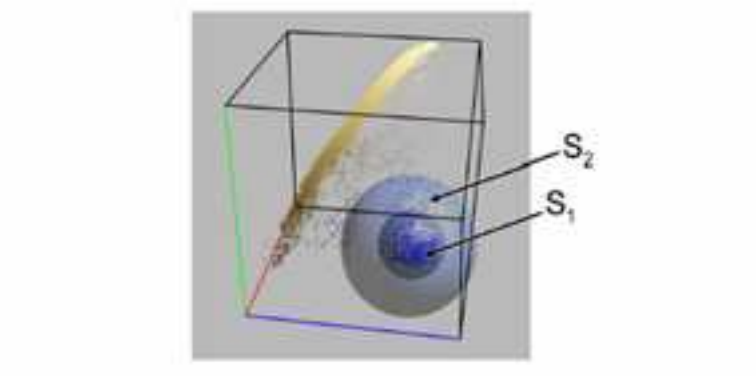
Maskování je inverzní proces k procesu skládání, kdy je dán kompozit C a snažíme se nalézt popředí F, pozadí B a masku α_f. Po dosazení do (2.2) získáme jednu rovnici pro tři neznámé. Je tedy zřejmé, že úloha maskování obecně nemá řešení, viz [6]. Proto byly vyvinuty speciální metody, které se snaží daný problém řešit. Takových metod existuje celá řada (pěkný přehled těchto metod je dán v [1]). Jednou z těchto metod je maskování na modré pozadí.

Jedná se o speciální příklad maskování, kdy se snažíme oddělit obraz popředí od pozadí, které má známou barvu. Tento proces byla poprvé představen v roce 1964 Petro Vlahostem, který použil modré pozadí, odtud název maskování na modrém pozadí. Nicméně v praxi se běžně používají i pozadí jiných barev (žlutá, červená, zelená).
Máme tedy dán kompozit C a barvu pozadí B. Pokud se podíváme na rovnici (2.2) je zřejmé, že stále nemá řešení. Proto byly vyvinuty metody, které se snaží řešit tento problém odhadnutím parametru α_f na základě jednoduchých heuristik a uživatelem nastavitelných konstant. Mezi tyto metody patří 3D Matting.

3D Matting je obecně soubor maskovacích metod založených na representaci obrazu ve třírozměrném barevném prostoru. Na obr.3 je jasně vidět, že pokud presentujeme obraz v prostoru RGB (každému pixelu obrazu odpovídá bod v RBG krychli s pozicí definovanou jednotlivými složkami jeho barvy), pak pozadí vytvoří shluk (modrých) bodů. 3D maskovací algoritmy se snaží definovat 3D tělesa, která oddělí shluk bodů představujících pozadí od ostatních. Pro vytvoření měkkých masek, jsou pak obvykle potřeba tyto objekty dva (kdy oblast mezi prvním a druhým tělesem obsahuje částečně průhledné body). Obr.4 ukazuje příklad použití nejjednoduššího obalového tělesa, tedy koule (všechny body uvnitř koule S1 jsou zcela průhledné, body vně koule S2 jsou zcela neprůhledné a body mezi S1 a S2 jsou částečně průhledné). Další metodou používanou pro maskování na modré pozadí je Chroma Matting



Obr. 3: RGB obraz převedený do 3D prostoru (zdroj <http://www.csie.ntu.edu.tw/~cyy/courses/vfx /07spring/lectures/handouts/lec10_matting.pdf> , upraveno)

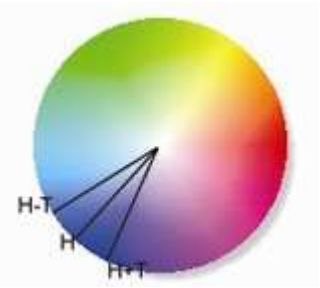


Obr. 4: Shluk pixelů pozadí obalený koulí S1 určující oblast zcela průhledných bodů a koulí S2 určující oblast částečně průhledných pixelů (zdroj <http://www.csie.ntu.edu.tw/~cyy/courses/vfx/07spring/ lectures/handouts/lec10_matting.pdf> , upraveno)

Termínem Chroma Matting jsou někdy nesprávně označovány všechny metody maskování na pozadí s jednolitou barvou. Ve skutečnosti tento termín označuje jen ty metody maskování na jednobarevném pozadí, které pro získání masky používají pouze informaci o odstínu barvy H, zatímco hodnoty saturace S a světlosti L jsou ignorovány.

Jednoduchá metoda pro získání masky využívající barevného prostoru HSL vypadá následovně. Nejprve je určen odstín pozadí H_{screen}. Protože ani v profesionálních studiích není možné dosáhnout obrazu, který má pozadí s konstantním odstínem, je třeba definovat hodnotu T, která určí rozsah tolerance viz obr.5. Masku α_{pixel} v libovolném bodu obrazu pak můžeme určit dle následujícího vztahu.

$$\alpha_{pixel} \begin{cases} 1 \text{ když } (H_{screen} - T) < H_{pixel} < (H_{screen} + T) \\ 0 \text{ jinak} \end{cases} \quad (2.3)$$



Obr. 5: Příklad vymezení oblasti okolo H (odstím pozadí) pomocí tolerance T (zdroj <http://www1.sapdesignguild.org/resources/glossary_color/index2.html>, upraveno)

Posledním příkladem metod pro maskování na modré pozadí, které uvádím je Color Difference Matting .

Tato metoda byla vyvinuta v roce 1964 Petro Vlahosem. Vlahos založil svoji metodu na předpokladu, že pro všechny barvy obsažené v objektu, který chceme vymaskovat ze scény, platí (převzato z [8]):

$$B > \max(R, G) \quad (2.4)$$

Kde R,G,B značí modrou, zelenou a červenou složku barvy. Pokud tedy zvolíme referenční barvu pozadí $C = [R_c, G_c, B_c]$, můžeme masku obrazu určit dle následujícího vztahu.

$$\alpha_{pixel} = \frac{B_{pixel} - \max(G_{pixel}, R_{pixel})}{B_c - \max(G_c, R_c)} \quad (2.5)$$

Protože v obraze se mohou vyskytnout i pixely pro něž ($B_{pixel} - \max(G_{pixel}, R_{pixel})$) < 0, nebo ($B_{pixel} - \max(G_{pixel}, R_{pixel})$) > ($B_c - \max(G_c, R_c)$) je nutné získanou hodnotu α_{pixel} oříznout na interval $\langle 0,1 \rangle$.

V předchozích odstavcích jsem uvedl tři nejpoužívanější metody používané pro maskování na modré pozadí. Následuje porovnání těchto metod z hlediska v hodnosti pro můj projekt.

Nejprve je nutné definovat požadavky, které na hledanou metodu klademe. Je nutné, aby hledaná metoda byla dostatečně rychlá, pokud možno snadno

implementovatelná, nevyžadovala vůbec nebo jen minimálně vstupy od uživatele a samozřejmě, aby bylo s její pomocí možné generovat co nejrealističtější komposity.

Metody ze skupiny Chroma keiyng jsou sice snadno implementovatelné i dostatečně rychlé, ale komposity generované pomocí nich nejsou dostatečně kvalitní. Důvodů, proč je tomu tak, je několik. Především H je pro amatérské DV kamery v tmavých místech špatně definovaný, další problémy přináší poměrně značný barevný šum čipů těchto kamer. DV data navíc procházejí kompresí podobnou JPEG, která dále barvonosné kanály zhoršuje.

Metody používající 3D maskování sice generují velmi realistické komposity, ale pouze při použití komplexních obalových těles, jejichž implementace je poměrně složitá a při jejichž použití nejsou tyto metody dostatečně rychlé.

Metoda Color Difference Matting je poměrně rychlá, implementačně jednoduchá, výsledné komposity mají dostatečnou kvalitu a je možné implementovat jednoduché uživatelské rozhraní pro jejich ovládání. Metoda tedy splňuje všechny základní požadavky, proto jsem ji zvolil pro realizaci mého projektu.

3 Trasování bodu

Trasování bodu je standardní úlohou počítačového zpracování obrazu s uplatněním v mnoha rozličných aplikacích. Obecně se úlohou trasování bodu rozumí snaha o nalezení pozic jednoho či více bodů v sekvenci většinou po sobě jdoucích snímků.

Existují minimálně dvě skupiny trasovacích algoritmů. Každá z nich byla vyvinuta pro řešení jiného okruhu problémů a tedy jejich přístupy k trasování bodu jsou zcela jiné.

První okruh jsou metody pro trasování tzv. význačných bodů (feature points tracking). Ty byly vyvinuty pro oblasti jako navigace, monitorování scény a s ní související identifikace pohybujících se předmětů atd. Společným znakem všech těchto metod je, že předem nejsou známy žádné nebo minimum informací o konkrétní fyzické podobě scény, kterou zpracováváme. Jinak řečeno, pokud například chceme trasovat pohybující se automobily na silnici, tak víme, že hledáme pohybující se automobil, nevíme však nic o jeho tvaru či barvě. Tento okruh problémů se řeší ve dvou krocích, nejprve jsou vhodným algoritmem detekovány tzv. význačné body obrazu a ty jsou po

té trasovány. Body nejsou trasovány na základě jejich fyzického vzhledu (porovnáním jejich okolí se známým vzorem), ale je k nim přistupováno jako ke shluku od sebe nerozeznatelných bodů, jejichž trasování je prováděno pouze na základě předem určených kinematických podmínek. Pěkný přehled základních algoritmů je dán v [5].

Druhá skupina trasovacích algoritmů se snaží řešit zcela jiný okruh problémů. Tyto algoritmy byly vyvinuty pro užití v oblastech jako je robotika, smíšená realita (kombinace reálného a virtuálního obrazu) a další. Jedním ze základních problémů, které se v těchto oblastech řeší, je určení pozice pozorovatele (příklady těchto algoritmů viz [4], [3]). Toto je možné pokud ve snímaném obraze dokážeme identifikovat dostatečný počet objektů se známou polohou v reálném světě. Tyto objekty se většinou označují jako „markery“ . Úlohou trasovacích algoritmů je v tomto případě nalézt pozici všech těchto markerů ve snímané scéně. Typický postup je následující. Nejprve je nutné definovat vhodné markery. Typicky se používají dvojrozměrné objekty s definovanými vlatnostmi jako tvar, barva, homogenita okolí a další (např. červený kroužek na černém pozadí). Na základě těchto předem definovaných vlastností jsou pak v obraze nalezena místa potenciálního výskytu hledaných markerů. V posledním kroku jsou pak tyto potenciální oblasti porovnány s předem známými šablonami markerů. Podle těchto šablon jsou jednotlivé markery identifikovány a tím je proces trasování dokončen (jednoznačně jsme identifikovali pozice bodů ve zpracovávaném obraze).

Možné jsou i kombinace obou předchozích přístupů. Například použít více identických markerů a po jejich nalezení v obraze provést trasování pomocí některého algoritmu pro význačné body.

Naším cílem je trasování bodů pro vytvoření efektu světelného meče. Je zřejmé, že první skupina algoritmů je pro tento účel nevhodná, protože se snažíme trasovat dva body, což na základě pouhé pozice těchto bodů bez omezení jejich pohybu nelze. Použití některého algoritmu ze druhé skupiny se rovněž nejeví nerealisticky, protože kladou na markery požadavky ohledně tvaru a homogenity okolí, které nelze v našem příkladě realizovat v praxi. Rozhodl jsem se tedy realizovat pouze detekci bodů na základě barvy (zvolil jsem modrou) a využít symetričnosti světelného meče (kdy pro náš účel není nezbytně nutné znát který bod je který) a body netrasovat. Návrh obecnějšího postupu trasování nechávám jako jeden z námětů pro další práci.

4 Planární projektivní transformace

Při vytváření světelného meče jsem narazil na problém jak namapovat texturu na oblast definovanou čtyřmi body (čtyřúhelník). Jedním z možných řešení je nalezení transformace, která převede souřadnice každého z těchto čtyř bodů na souřadnice jednotlivých rohů textury. Pomocí takové transformace pak lze pro všechny body zadaného čtyřúhelníku určit odpovídající body v textuře a provést tak namapování textury do vstupního obrazu. Postup jak nalézt takovou transformaci je popsán dále.

Představme si tedy, že máme dány dvě čtveřice bodů ve dvojrozměrném prostoru a známe jejich vzájemné korespondence a naším úkolem je spočítat transformaci, která převede všechny body z první čtveřice na jim odpovídající body z druhé čtveřice. Pokud převedeme zadané body do homogenních souřadnic, můžeme danou úlohu zapsat pomocí následující rovnice:

$$\mathbf{x}' = \mathbf{T}\mathbf{x}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (4.1)$$

Kde $\mathbf{x}$ značí bod libovolný z původní čtveřice, $\mathbf{x}'$ značí jemu odpovídající bod ze druhé čtveřice a $\mathbf{T}$ je transformační matice, kterou chceme nalézt. Pro každou dvojici bodů můžeme sestavit následující rovnice:

$$x' = \frac{t_{11}x + t_{12}y + t_{13}}{t_{31}x + t_{32}y + t_{33}} \quad (4.2) \qquad y' = \frac{t_{21}x + t_{22}y + t_{23}}{t_{31}x + t_{32}y + t_{33}} \quad (4.3)$$

po roznásobení :

$$x'\left(t_{31}x + t_{32}y + t_{33}\right) = t_{11}x + t_{12}y + t_{13} \quad (4.4)$$

$$y'\left(t_{31}x + t_{32}y + t_{33}\right) = t_{21}x + t_{22}y + t_{23} \quad (4.5)$$

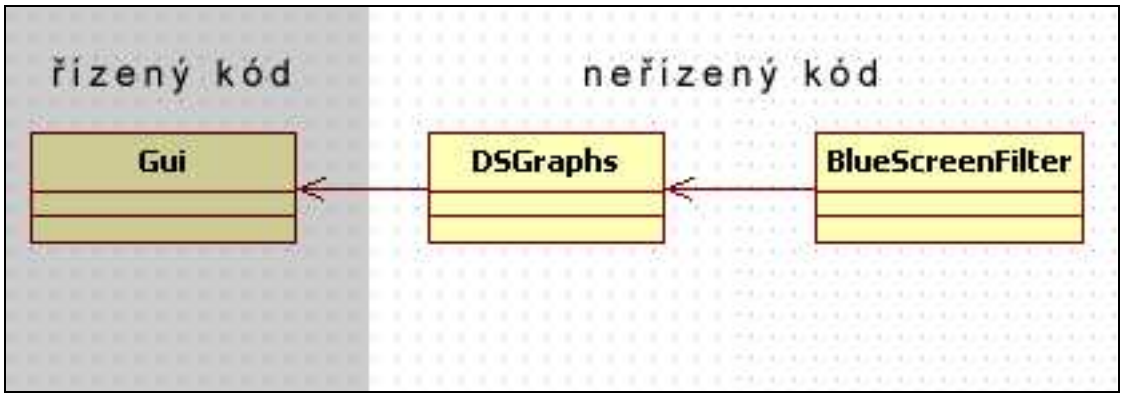
Máme k dispozici čtyři dvojice bodů, můžeme tedy sestavit pouze osm rovnic pro devět neznámých. Řešením je zvolit parametr $t_{33}=1$, což si můžeme dovolit pro všechny body, které se nenachází v nekonečnu. Můžeme tedy sestavit soustavu rovnic:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 \\ x_2 & y_2 & & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 \\ x_3 & y_3 & & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3'x_3 & -y_3'y_3 \\ x_4 & y_4 & & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4'x_4 & -y_4'y_4 \end{pmatrix} \begin{pmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix} \quad (4.6)$$

Jejímž vyřešením získáme všechny zbývající členy transformační matice.

5 Realizační část

V této části nejprve popíšu strukturu celého programu a prostředky s jejichž pomocí jsem program realizoval. Dále pak bude následovat detailnější popis jednotlivých částí systému, jejich funkcionalita, implementace, případně poznámky k alternativním postupům řešení.



Obr. 6: objektový model programu

5.1 Použité Prostředky

Program jsem realizoval v prostředí .NET v jazyce VC++. Toto prostředí jsem zvolil z několika důvodů. Nejdůležitější pro mě byla schopnost .NETu kombinovat řízený a neřízený kód. Navíc použití jazyka VC++ mi umožnilo kombinovat řízený a neřízený kód uvnitř jediného projektu (což je obzvlášť výhodné při debbugování programu).

Části, které nejsou kritické pro rychlost běhu celého programu, jsem napsal v řízeném kódu a mohl tak využít všech výhod, které nabízí .NET Framework (Garbage collector, visuální návrhář atd.). Naproti tomu části programu obsahující většinu funkcionality jsem napsal v neřízeném kódu, jehož běh je efektivnější (jedním z důvodů byla i absence oficiálního řízeného rozhraní pro architekturu DirectShow, kterou jsem použil pro zpracování video vstupů).

Jak jsem zmínil výše, pro práci s multimediálními vstupy jsem použil architekturu DirectShow, která je standardem pro zpracování proudů multimediálních dat na platformě Windows.

Program má klasickou dvouvrstvou strukturu (viz obr.6), kdy je od sebe jasně oddělena část obsahující uživatelské rozhraní (třída Gui) a část provádějící získávání, zpracování a ukládání dat (třídy DSGraphs, BlueScreenFilter).

5.2 Popis jednotlivých tříd

V následujících podčástech jsou popsány jednotlivé třídy programu (jak jejich celková funkcionalita, tak některé důležité funkce)

5.2.1 Třída Gui

Tato třída využívá standardních možností .NET Frameworku pro vytvoření grafického uživatelského rozhraní. Vytvořené rozhraní má dvě verze. Základní verze umožňuje uživateli pomocí tří tlačítek spouštět jednotlivé grafy vytvořené třídou DSGraphs. Po přepnutí do rozšířené verze může uživatel přímo nastavovat některé parametry procesů maskování a detekce bodu implementovaných ve třídě BlueScreenFilter. Tato verze rozhraní není určena pro koncového uživatele. Je učena pouze pro demonstrování správné funkce programu. Pro více informací o funkci jednotlivých uživatelských prvků viz uživatelská příručka.

Pokud se v programu vyskytne nějaká chyba, je tato skutečnost uživateli oznámena pomocí standardního MessageBoxu.

5.2.2 Třída DSGraphs

Tato třída využívá architektury DirectShow pro získávání, zpracování, vykreslování a ukládání multimediálních dat (předpokládám, že čtenář má alespoň základní znalost struktury DirectShow).

Třída obsahuje tři druhy funkcí (podle činnosti, kterou vykonávají) . První z nich jsou funkce, které jsou využívány při sestavování jednotlivých grafů. Další jsou funkce, které řídí již sestavené grafy. Poslední jsou funkce předávající parametry zadané uživatelem v rozšířené verzi grafického rozhraní instanci třídy BlueScreenFilter.

Filtry, ze kterých jsou grafy sestavovány, lze rozdělit do dvou skupin. První skupina filtrů je vytvořena při vzniku instance třídy a existuje po celou dobu existence této instance. Druhá skupina obsahuje filtry pro načítání a ukládání dat. Tyto filtry je nutné vytvořit pokaždé, když je sestavován nový graf (viz popis funkcí sestavujících jednotlivé grafy). Následuje popis důležitých funkcí.

```
void InitCaptureGraphPreview(HWND  owner)
```

Funkce sestaví a spustí graf, který pomocí instance třídy BlueScreenFilter zpracovává data z DV kamery připojené k počítači a výsledný obtraz vykresluje do okna, jehož handle je předán jako vstupní parametr. Před sestavením tohoto grafu je nutné zjistit, zda je k počítači připojena DV kamera a případně získat filter, který tuto kamaru reprezentuje. Detekce kamery a případné vytvoření příslušného filtru je prováděna pomocí fce `HRESULT FindDevice(const GUID deviceCategory, IBaseFilter **device).` Pokud k počítači není připojena požadovaná kamera, funkce pouze zastaví a rozpojí aktuální graf.

```
void InitCaptureGraph()
```

Funkce sestaví a spustí graf, který přijímá data z DV kamery a ukládá je do souboru na pevném disku počítače. Data nejsou během ukládání nijak modifikována. Opět je prováděna detekce kamery pomocí fce `HRESULT FindDevice(const GUID deviceCategory, IBaseFilter **device).` Pokud k počítači není připojena žádná kamera, funkce pouze zastaví a rozpojí aktuální graf.

```
void InitVideoGraph(HWND  owner1 ,WCHAR* path);
```

Funkce sestaví graf , který pomocí instance třídy BlueScreenFilter zpracovává data ze souburu, jehož jméno a cesta jsou specifikována jako vstupní parametr path. Data dále komprimuje a ukládá do souboru .//blueScreen.avi. Komprese a ukládání na disk může být v závislosti na hodnotě boolovské proměnné render nahrazena vykreslováním do okna, jehož handle je předán jako vstupní parametr (tato funkce je přístupná pouze při použití rozšířené verze uživatelského rozhraní). Proměnnou render lze nastavit pomocí fce `void RenderFile(BOOL render).`

```
HRESULT FindDevice(const GUID deviceCategory, IBaseFilter **device)
```

Tato funkce prochází všechny kamery připojené k počítači, dokud nenajde DV kameru připravenou k přenosu dat. Nalezenou kameru předá zpět volající fci v parametru device. Pokud k počítači žádná taková kamera připojena není, vrátí E_FAIL.

```
HRESULT SetBimaps()
```

Tato funkce předá instanci třídy BlueScreenFilter absolutní cesty k obrázkům reprezentujícím pozadí pro vytvoření kompozitu a textury světelného meče. Cesty je nutné vytvářet a předávat z této třídy. Při pokusu o použití relativních cest přímo ve filtru BlueScreenFilter jsem zjistil, že jeho pracovní adresář je bez ohledu na místo spuštění aplikace vždy C:\\.

```
void ProcessEvents()
```

Tato funkce je spouštěna v samostatném vlákně a zpracovává události generované jednotlivými grafy.

5.2.3 Třída BlueScreenFilter

Tato třída implementuje filtr, který provádí maskování na modré pozadí a detekci bodů, pomocí nichž se snaží vytvořit efekt světelného meče. Pro získání funkcionality požadované od všech filtrů určených pro DirectShow dědí tato třída od třídy `CTransInPlaceFilter`. Pro vytvoření funkčního filtru je navíc nutné implementovat dvě virtuální funkce ze třídy `CTransInPlaceFilter`. První z nich je funkce `HRESULT CheckInputType(const CMediaType *pmt),` která určuje formát vstupních dat filtru. Tento filter akceptuje pouze nekomprimované video ve formátu

RGB24. Druhou funkcí, kterou je nutné implemntovat, je funkce `HRESULT Transform(IMediaSample *pSample)`. Tato funkce provádí požadovanou transformaci vstupních dat pomocí volání dalších funkcí této třídy. Transformace dat probíhá v několika krocích. Každému kroku odpovídá volání jedné funkce. Následuje popis jednotlivých funkcí v pořadí, jak jsou volány při zracovávání vstupního obrazu.

```
void  Preprocess(BYTE* image,  int imageWidth, int imageHeigth,
                 int *maxDifference)
```

Tato funkce provádí dvě operace. Za prvé z obrazu odstraňuje prokládání metodou zvanou „Blur“. Při tom zároveň ve všech sudých řádcích obrazu hledá pixel, jehož barva má největší rozdíl mezi zeleným kanálem a větším ze dvou zbylých kanálů. Největší nalezený rozdíl je pak vrácen ve výstupní proměnné maxDifference. Hledat největší rozdíl jen v sudých řádcích si mohu dovolit vzhledem k tomu, že při odstranění prokládání metodou „Blur“ jsou všechny liché řádky spočítány jako průměr dvou sousedních (sudých) řádků. Z toho vyplívá, že hodnoty hledaného rozdílu nemohou být pro tyto řádky vyšší než nalezené maximum.

```
short*  Process(BYTE *image, int thresh, int *label_num, int **area,
                float **pos, int **clip, int **label_ref,
                int **differences, int sampleWidth, int sampleHeigth,
                float *alphaArray, int maxDiff);
```

Na prvním místě je nutné zmínit, že tato funkce je modifikací funkce obsažené v [5], podléhá tedy podmínkám uvedeným v „General Public Licence“ a je možné ji využít pouze pro nekomerční účely.

Ta to funkce opět provádí dvě operace. První z nich je detekce modrých oblastí v obraze. Při této detekci je pro každý pixel zjištěna hodnota rozdílu mezi modrým kanálem a větším ze zbylých dvou kanálů (dále jen rozdíl). Pokud je tento rozdíl vyšší než prahová hodnota, program předpokládá, že pixel je součástí nějaké modré oblasti a je proveden „labeling“ tohoto pixelu. Při „labelingu“ je prohledáno již zpracované okolí pixelu a je určeno, zda se pixel přiřadí k již existující oblasti (pokud pixel sousedí s jiným „modrým“ pixelem), nebo zda vytvoří novou oblast. Pro tyto oblasti jsou průběžně počítány statistiky jako velikost oblasti, střed oblasti atd. Pokud se v okolí nově detekovaného modrého pixelu nachází dvě oblasti (pixel sousedí s oběma), jsou tyto oblasti sloučeny. Při sloučení jsou samozřejmě přepočítány všechny statistiky. Výstupem této části algoritmu je pak počet detekovaných oblastí (`label_num`) a pole určující jejich vlastnosti. Mezi sledované vlastnosti patří střed oblasti (`pos`), velikost

oblasti (area), průměrný rozdíl (`differences`) a největší a nejmenší hodnota souřadnic x,y prvků oblasti (clip).

Zároveň s detekcí modrých oblastí jsou počítány hodnoty masky pro jednotlivé pixely. Tyto hodnoty jsou určeny podle následujícího jednoduchého vztahu

$$\alpha_{\text{pixel}} = \frac{\text{maxDifference}}{\text{difference}_{\text{pixel}}}$$

kde maxDifference je hodnota největšího rozdílu mezi zeleným kanálem a větším ze dvou zbylých kanálů obsažená ve zpracovávaném obraze (je to jeden ze vstupních parametrů této fce) , $\text{difference}_{\text{pixel}}$ značí hodnotu tohoto rozdílu pro aktuální pixel a α_{pixel} je hodnota masky pro tento pixel.

Předtím než je hodnota α_{pixel} uložena na příslušné místo do pole alphaArray, je zespodu ořízнuta na interval $\langle 0,1 \rangle$.

```
void  FilterAlpha5x5(float** alpha, int width, int heigth);
```

Tato funkce aplikuje na všechny prvky pole `alpha,` s výjimkou krajních dvou řádek a sloupců, Gausůvský filtr rozměrů 5x5. Důvodem pro aplikaci tohoto filtru je značné zlepšení výsledné masky.

```
void  DifferenceMatte2(BYTE* image, int numOfPixels,
                       float* aplhaArray);
```

Tato funkce vytváří pomocí vstupního obrazu (image) a jeho masky (`aplhaArray`) , výsledný kompozit. Při skládání obrazu popředí s novým pozadím jsou z popředí odstraněny zelené odplesky aplikací pravidla, které dovoluje aby zelený kanál přesahoval modrý kanál o maximálně 50% hodnoty o kterou červený kanál přesahuje modrý, viz následující pseudo kód.

G = zeleny kanál barvy popředí
B = modrý kanál barvy popředí
R = červený kanál barvy popředí
pokud G > B
    pokud B > R // pokud je B větší než R nepoužívá se rozdíl (který je záporný) a
                //zelený kanál se prostě ořízne na velikost modrého
        G = B
    jinak

G = min (G, (R-B)*0,5 +G) // operace min se využíva protože v žádném případě
//nechceme hodnotu zeleného kanálu zvětšit

konec podmínky

konec podmínky

Důsledky této operace jsou posány v části 7.1.

```
void  Draw(BYTE *image, int imageWidth, int imageHeigth,
           int label_num,int *area, int *clip, float *pos,
           int* differences)
```

Tato funkce provádí vykreslení světelného meče. Z detekovaných oblastí jsou nejprve vybrány dvě, které mají největší hodnotu v poli differences. Středy těchto oblastí (pos) jsou pak vzaty jako krajní body meče. Z těchto dvou bodů jsou pomocí fce

void CreateLinesFromTwoPoints(Point* points, Line * lines)

vytvořeny čtyři body a z nich čtyři úsečky. Aby bylo možné na oblast vytyčenou těmito body namapovat texturu, je třeba najít transformaci mezi nimi a rohy textury. Proto je sestavena příslušná soustava rovnic (viz 4), která je vyřešena pomocí Gausovy eliminace. Po získání transformační matice je vykreslen světelný meč metodou řádkového plnění.

Kromě funkcí zpracovávajících vstupní obraz obsahuje třída další pomocné funkce

```
HRESULT LoadBitmap(const char* fileName,  const char* mode,
                   BYTE bitmap[]);
HRESULT AllocMemory();
void  FreeMemory();
```

jejichž funkce je zřejmá.

6 Návrh hardwaru

Návrh realizace hardwarového vybavení je nedílnou součástí této práce. V následující části je nejprve popsána možná realizace studia a poté je dán návrh prostředků pro vytvoření světelného meče.

6.1 Realizace studia

Pro vytvoření studia je potřeba několik základních komponent. První potřebnou částí je vhodné pozadí.

Na pozadí použité při nasazení metody Color Diference Matting je kladeno několik základních nároků. Pozadí by mělo mít pokud možno co „nejčistější barvu“. Tím se rozumí, že pro modré pozadí by modrý kanál barvy vyjádřené v RGB měl co nejvíce převyšovat ostatní dva kanály (analogicky pro zelené pozadí). Typická hodnota rozdílu mezi modrým kanálem a větším ze dvou zbývajících kanálů, kterou by mělo splňovat dobré pozadí, je 50 (za předpokladu že rozsah hodnot jednotlivých kanálů je 0..255). Pozadí by mělo dále mít matnou povrchovou úpravu, mělo by být barevně stálé a dostupné ve vhodném formátu (ve smyslu rozměrů).

V praxi se při amatérském maskování na modré pozadí používá celá řada různých materiálů, lišících se dostupností i cenou. Nejčastěji jsou uváděny tři a to papír, látka a fólie z PVC. Jako první z nich jsem testoval papír. Papír se vyrábí ve velké barevné škále, matné povrchové úpravě a je jednoznačně nejlevnější za všech testovaných materiálů. Při testování papírů v různých odstínech modré a zelené jsem nalezl odstíny, které zhruba splňovaly všechny požadavky na dobré pozadí s výjimkou posledního. Největší formát, ve kterém lze papír v těchto barvách koupit, je A3 (297mm x 210mm). To se ukázalo jako značný problém a v podstatě to znemožňuje nasazení do praxe.

Dalším zkoumaným materiálem byla látka. Zde jsem narazil na problém nedostatečné nabídky jednobarevných látek (drtivá většina nabízených látek obsahuje různé vzory což vylučuje jejich použití). Další podstatnou nevýhodou je cena, která se pohybuje od 150 Kč/m2 výše. Po předchozí zkušenosti s papírem, kdy jsem si udělal představu jaké barevné odstíny jsou pro pozadí vhodné, jsem se rozhodl neinvestovat peníze do koupě barevných látek, jejichž barvy by se s největší pravděpodobností ukázaly jako nevhodné.

Posledním testovaným materiálem bylo PVC. Fólie z PVC se prodávají především pro účely výroby reklamních předmětů, proto jsou nabízeny v široké škále barev, v matné i lesklé povrchové úpravě. Výrobce zaručuje barevnou stálost po nejméně čtyři roky. Fólie lze odebírat v pásech šíře 0,5m a 1m.

Testoval jsem několik vzorků fólií různých odstínů modré a zelené. Jako nejlepší z nich z testu vyšla fólie firmy Avery Denison s označením 35/14. Pro tuto barvu

vycházel poměr zeleného kanálu ku většímu ze zbylých kanálů (modrému) v průměru kolem dvou, což se blíží hodnotě udávané pro „typické dobré pozadí“. Navíc PVC lze snadno udržovat (je omyvatelné a nemačká se). Protože tento materiál, narozdíl od ostatních, splňuje všechny základní požadavky na dobré pozadí, zvolil jsem ho pro realizaci projektu. Srovnání cen jednotlivých pozadí viz tab. 6.1

Materiál	**Cena (Kč/m2)**
Papír	15
Fólie z PVC	100
Látka	150

Tab. 6.1 Přehled cen používaných pozadí

Použitá kamera je další část systému, která zásadně ovlivňuje kvalitu výsledného kompositu. Při výběru kamery pro mě byla zásadním omezením cena, ta by se měla pohybovat v rozmezí 10000 – 15000Kc. Požadavek minimálních nákladů mi dovolil vybírat pouze z amatérských DV kamer.

Pro maskování je dobré mít kameru s chroma podvzorkováním nejlépe 4:4:4 nebo 4:2:2.Všechny amatérské kamery nabízejí pouze 4:2:0, ale vzhledem k omezením popsaným výše jsem neměl na výběr.

Kamera by měla splňovat několik další požadavků. Měla by nabízet možnost manuálního nastavení délky expozice, manuálního ostření a vyvážení bílé barvy. Celá řada na trhu běžně dostupných amatérských kamer splňuje tyto požadavky.

Pro realizaci studia jsem měl k dispozici kameru Canon MV8300i, která se ukázala jako postačující. Vyzkoušení dalších typů kamer a zjištění zda není možné jejich použitím ještě zlepšit kvalitu výsledného kompositu, je jedním z námětů pro další práci. Další nezbytnou součástí studia jsou světla.

Světla jsou často označována za vůbec nejdůležitější součást celého systému a správné nasvícení scény za faktor nutný k vytvoření vysoce kvalitních kompositů. Profesionální studia jsou proto vybavena světly, jejichž cena se pohybuje v řádech desítek až stovek tisíc korun. To je samozřejmě mimo naše finanční možnosti. Ze světel běžně dostupných na trhu, se nabízejí především dvě varianty. První z nich je použití běžných halogenových reflektorů, druhá použití fluorescenčních zářivek.

Pokud porovnáme vlastnosti světla produkovaného těmito zdroji, jeví se jednoznačně lépe zářivky, které produkují měkké bílé světlo oproti poměrně ostrému

žlutému světlu halogenových reflektorů. Zářivky se jeví lépe také co se týče ekonomičnosti jejich provozu, protože jejich spotřeba je podstatně nižší než je tomu u halogenových reflektorů. V použití zářivek mi zabránila především nedostupnost vhodného nosiče, do kterého by bylo možné zářivku umístit (pro vytvoření vlastního mi chyběly odborné znalosti i finanční prostředky). Proto jsem pro realizaci vybral halogenové reflektory, jejichž vlastnosti nejsou ideální, avšak postačují. K němu je však nutné dokoupit ještě halogenovou žárovku, přívodní kabel a vidlici. Ceny těchto komponent jsou uvedeny v tab 6.2.

Komponenta	**Cena/jednotka**
halogenový reflektor	289 Kč/ks
kabel (guma)	75Kč/m
halogenová žárovka	90Kč/ks
postraní vidlice	35Kč/ks

Tab. 6.2: Přehled cen komponent potřebných pro osvětlení studia určeného pro maskování na modré pozadí

Nezbytnou součástí studia je samozřejmě počítač na kterém bude prováděno zpracování obrazu. Pro realizaci by měla být dostatečná běžná počítačová sestava pohybující se do 20 000Kč. Jediným speciální požadavkem na počítač je vstup pro rozhraní FireWire.

6.2 Realizace světelného meče

Navrhnutý trasovací algoritmus je schopen detekovat modré oblasti vyskytující se ve vstupním obrazu. Pro realizaci světelného meče bylo tedy nutné navrhnout a otestovat vhodný objekt, který by bylo možné trasovat na základě jeho modré barvy.

Nejprve jsem testoval tyč s koncem obarveným na modro. Při testování jsem narazil na zásadní problém ve chvíli, kdy na konec tyče nedopadal dostatek světla. V takovém případě se barva objektu jevila jako téměř černá a trasovací algoritmus selhával. Proto jsem se rozhodl otestovat různé LED a žárovky. Při testování jsem použil nejprve modré led, různých rozměrů vydávající jak difusní tak ostré světlo různých intenzit. U všech testovaných diod jsem narazil na dva problémy. Prvním z nich bylo, že vyzařovací úhel většiny diod se pohybuje v rozmezí od 60 do 120 stupňů. Pokud tedy byly v pozici, kdy svítily směrem od kamery, nebylo možné je

navrženým postupem detekovat. Druhý problém je, že diody se v obraze jeví jako téměř bílé body, které jsou jen lehce namodralé. Jejich barva se navíc značně mění v závislosti na úhlu, ze kterého jsou kamerou snímány (od jasně bílé po jasně modrou). Vyzkoušel jsem tedy barevné žárovky. Při jejich použití sice odpadl problém s vyzařovacím úhlem, ale jejich barva v obraze byla podobná jako u diod. Tento problém jsem nakonec vyřešil umístěním modré diody do obalu z modrého plexiskla, které rozptýlí světlo vycházející z diody a sníží jeho intenzitu. Barva LED umístěné v tomto obalu je pak snadno detekovatelná navrženým postupem.

Další komponenty nutné k vytvoření meče jsou tři tužkové baterie, spínač, pouzdro na baterie, dráty na spojení všech elektrických částí a plexisklová trubka. Při testování jsem tuto trubku nahradil z finančních důvodů dřevěnou tyčí. Použití dřevěné tyče pro finální realizaci je nevhodné, protože při určitých úhlech pohledu překrývá jednu z diod (viz část 7). Při umístění diod do plexisklové trubky navíc můžeme libovolně zvolit jejich pozici v „meči“, což není při použití dřevěné tyče možné. Přehled cen všech komponent je uveden v tab. 6.3.

Komponenta	**Cena/jednotka**
LED	21Kč/ks
tužková baterie nabíjecí	75Kč/ks
spínač	25Kč/ks
pouzdro na baterie	11Kč/ks
trubka z plexiskla	200Kč/m

Tab. 6.3: Přehled cen komponent potřebných pro vytvoření světelného meče

7 Ověření výsledků

V této části jsou ukázky výstupů jednotlivých částí algoritmu. Jsou v ní také jasně definovány podmínky, za kterých celý systém funguje. Na závěr této kapitoly uvádím tabulku rychlostí běhu programu změřenou na různých počítačích.

Metoda použitá pro maskování na zelené pozadí funguje spolehlivě, pokud jsou splněny všechny následující podmínky. Snímaná scéna musí být dobře nasvícená. Pokud scéna obsahuje ostré stíny nebo velmi jasné odlesky světla, vznikají ve výsledném obraze nepříjemné artefakty, se kterými se program nedokáže vypořádat (viz

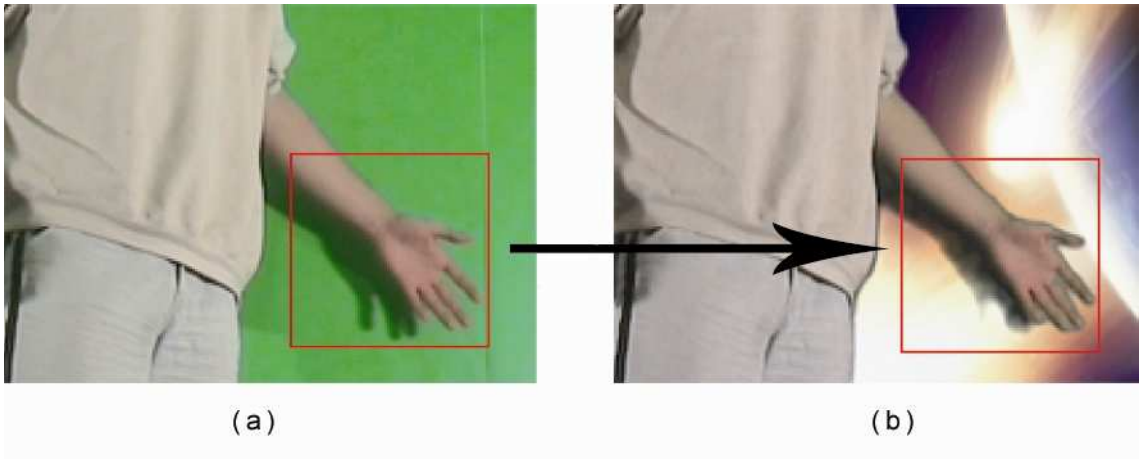
obr.7). Objekty v popředí, které chceme maskovat, samozřejmě nesmí obsahovat zelenou barvu. Neměly by obsahovat ani žádné průhledné či průsvitné části.

Při procesu odstranění zelených odlesků z předmětů v popředí je na všechny jejich pixely uplatněno následující pravidlo. Zelený kanál může přesahovat modrý kanál maximálně o 50% hodnoty, o kterou červený kanál přesahuje modrý. Uplatnění tohoto pravidla výrazně zlepšuje kvalitu výsledného kompositu (viz obr. 8 a obr.9), zároveň však způsobuje problémy při maskování žlutých předmětů. Pokud předmět popředí obsahuje žlutou barvu nebo barvy jí podobné, dostávají tyto barvy po uplatnění výše uvedeného pravidla mírný červený nádech. Vzhledem k výhodám, které odstranění zelených odlesků přináší, jsem se tento barevný posun u žluté barvy rozhodl přijmout jako „nutné zlo“.

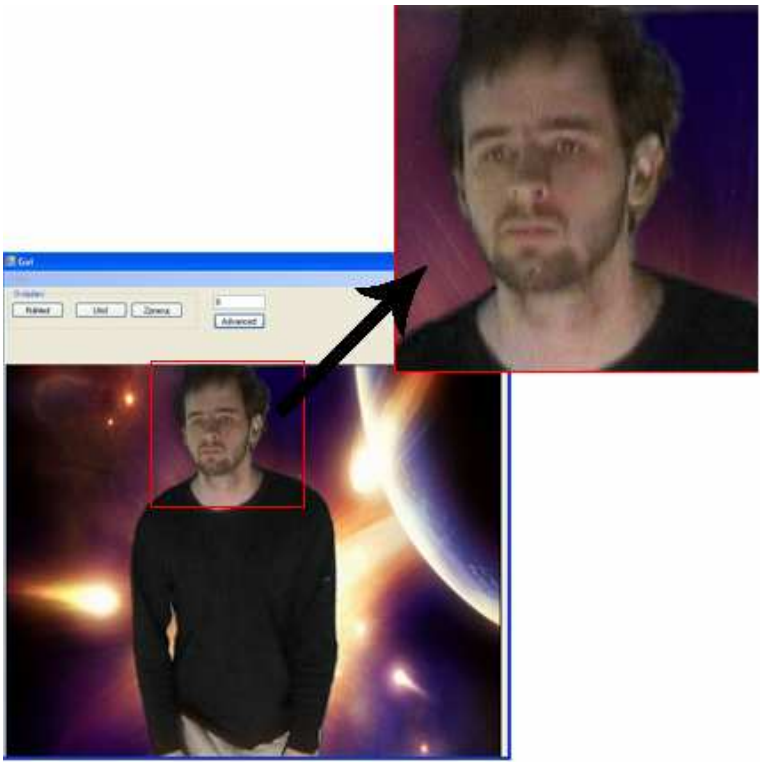
Metoda použitá pro vytvoření meče funguje (viz obr. 10) pokud jsou splněny dvě podmínky. První z nich souvisí s detekcí koncových bodů meče. Pro správnou funkci programu je nutné, aby se v obraze nevyskytovaly žádné jiné modré objekty. Pokud algoritmus detekuje více než dva modré objekty, vybere z nich dva s největším průměrným rozdílem mezi modrým kanálem a větším ze zbylých dvou kanálů. Tento postup funguje dobře pokud se část světla vydávaného diodami odráží od oděvu člověka, který meč drží. Pokud se však ve scéně vykytuje další modrý objekt (mimo diod), algoritmus selhává.

Druhé omezení souvisí s mapováním textury meče do výsledného obrazu. Aby bylo možné texturu správně mapovat, je potřeba v obraze detekovat oba koncové body meče. Tato podmínka odpovídá realitě s jednou výjimkou a to pokud meč míří přímo na kameru. V takovém případě je vidět pouze jeden koncový bod meče. Během vývoje aplikace jsem se pokusil ošetřit tuto možnost pomocí určení hranice modré oblasti detekované algoritmem. Detekované hranice se však ukázaly, vlivem rozmazání vstupního obrazu, jako nepoužitelné (nepodařilo se mi na takto definované oblasti namapovat texturu tak, aby byl výsledný efekt vizuálně přijatelný) .

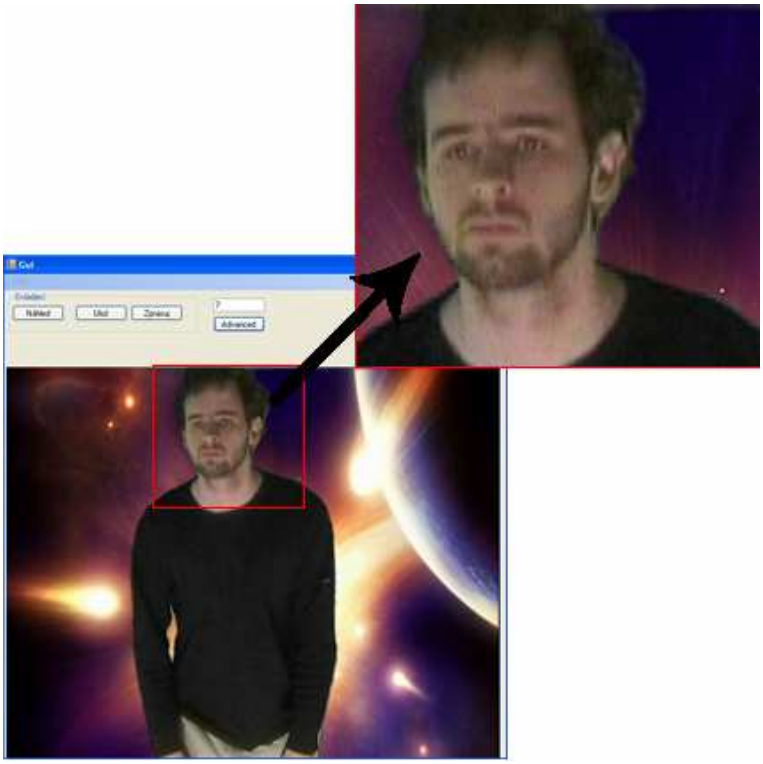
Pro konečnou realizaci jsem pro vytvoření těla světelného meče navrhl trubku z průhledného plexiskla. Tu jsem při testování musel nahradit dřevěnou tyčí (z finančních důvodů). Program tedy samozřejmě nefunguje správně, pokud je jedna z diod překryta touto tyčí (viz obr. 11). Při použití plexiskla tento problém odpadá.



Obr. 7: (a) snímek špatně nasvícené scény, (b) kompozit vytvořený ze snímku (a)

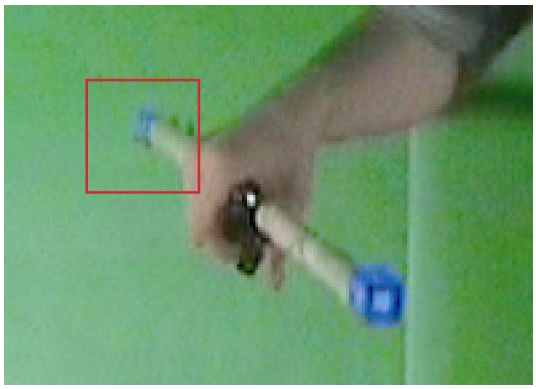


Obr. 8: Ukázka výstupu programu s odstraněním zelených odlesků



Obr. 9: Ukázka výstupu programu bez odstraněním zelených odlesků

Obr. 10: Ukázka výstupu programu, který obsahuje světelný meč



Obr. 11: Tyč překrývající jedenu z diod (ta tak nemůže být detekována)

Při testování rychlosti jsem se omezil na část programu zpracovávající vstupní data z pevného disku počítače. Výsledky testů jsou uvedeny v tab7.1.

Počítačová sestava (operační paměť, procesor, základní deska)	Rychlost programu (fps)
512 MB RAM, Intel Pentium 4 2,8 GHz, ASUS P4P800 SE	11
1024 MB RAM, Intel Pentium D 2.6 GHz, MSI 945PL	14
1024 MB RAM, AMD Athlon XP 3200++, ASUS A7V600	10

Tab. 7.1: rychlosti běhu programu (ve snímcích za sekundu) na různých počítačových sestavách

8 Závěr

Vytvořil jsem fungující aplikaci představující základní možnosti procesu maskování na zelené pozadí. Myslím, že kvalita výsledných kompositů je s ohledem na použité hardwarové prostředky velmi dobrá.

Navrhl jsem všechny prostředky nutné k vytvoření funkčního studia pro maskování na zelené pozadí. Při návrhu jsem se soustředil na minimální cenu jednotlivých komponent. Navržené hardwarové prostředky, tedy vzhledem k jejich ceně, nedosahují kvalit profesionálního vybavení. Přesto testování ukázalo, že jsou poměrně dobře použitelné.

Tato aplikace nabízí celou řadu námětů na další práci. Dle mého názoru by se další úsilí mělo zaměřit především na zvýšení rychlosti programu, vylepšení efektu světelného meče a navržení obecnějšího trasovacího algoritmu, který by umožnil implementaci dalších vizuálních efektů.

Literatura

[1] Yung-Yu, Chuang. New *Models and Methods for Matting and Compositing*. 2004 [cit. 2007-05-01] Dostupné z <http://grail.cs.washington.edu/theses/ChuangPhd.pdf>.

[2] Yung-Yu Chuang. *Matting and compositing*. 2007-05-1 [cit. 2007-05-05] Dostupné z <http://www.csie.ntu.edu.tw/~cyy/courses/vfx/07spring/lectures/handouts/lec10_matting.pdf>.

[3] Kato, Hirokazu. *ARToolKit* Ver 2.72. 2007 Dostupné z <http://www.hitl.washington.edu/artoolkit/>.

[4] Cho, Youngkwan. – Park, Jun. - *Neumann, Ulrich, Fast Color Fiducial Detection and Dynamic Workspace Extension in Video See-through Self-Tracking Augmented Reality. 1998.* [cit. 2007-05-05] Dostupné z <http://graphics.usc.edu/cgit/pdf/papers/PacificGraphics98-AR.pdf>

[5] Verestóy, Judit. – Chetverikov, Dmitry. *Feature Point Tracking Algorithms*. *1998-12-28.* [cit. 2007-05-05] Dostupné z <http://visual.ipan.sztaki.hu/psmweb/ >.

[6] Kheng, Leow Wee. *Theory of Matting. 2006.* [cit. 2007-05-05] Dostupné z <http://www.comp.nus.edu.sg/~cs5245/lecture/matte.pdf >

[7]Porter, Thomas. - Duff, Tom. *Compositing Digital Images.* 1984. [cit. 2007-05-05] Dostupné z <http://keithp.com/~keithp/porterduff/>

[8] Smith, Alvy Ray. - Blinn, James. *Blue Screen Matting*. 1996 [cit. 2007-05-05] Dostupné z <graphics.cs.cmu.edu/courses/15-463/2006_fall/www/Papers/smith-blinn.pdf>

Přílohy

A. Uživatelská příručka

Program má jednoduché ovládání. Požadovaného výsledku zle dosáhnout ve čtyřech krocích:

1. Nejprve je nutné se přesvědčit, že kamera připojená k počítači je zapnutá a není v režimu uspání. Pokud je kamera v tomto režimu, je nutné ji „vzbudit“ (provedení této operace závisí na druhu použité kamery).

2. Dalším krokem je spuštění náhledu pomocí tlačítka „Náhled“ (na obr. 12 označené jako číslo 1). Po stisknutí tohoto tlačítka program začne zpracovávat data z kamery a zobrazí je uživateli (viz obr. 13). Běh náhledu není časově omezen.

3. Dalším krokem je uložení videa na disk počítače. To se provede stisknutím tlačítka „Uložit“ (na obr. 12 označené jako číslo 2). Po jeho stisknutí se zastaví náhled (program přestane zobrazovat video) a začne ukládání dat z kamery na pevný disk počítače trvající 30 sekund. Skončení ukládáni je uživateli oznámeno pomocí hlášení „Ukládání ukončeno“.

4. Posledním krokem je pracování videa uloženého v předchozím kroku. Pro start tohoto procesu je nutné stisknout tlačítko „Zpracuj“ (na obr. 12 označené jako číslo 3). Dokončení této fáze ohlásí program pomocí hlášení „Zpracování souboru bylo dokonceno“.

Kromě výše popsaných tlačítek obsahuje uživatelské rozhraní ještě dva prvky. Prvním z nich je políčko zobrazující počet snímků vykreslených za sekundu (na obr. 12 označené jako číslo 4), toto políčko je aktivní pouze v režimu „Náhled“. Druhým je tlačítko „Advanced“ (na obr. 12 označené jako číslo 5). Toto tlačítko přepne uživatelské rozhraní do rozšířeného módu.

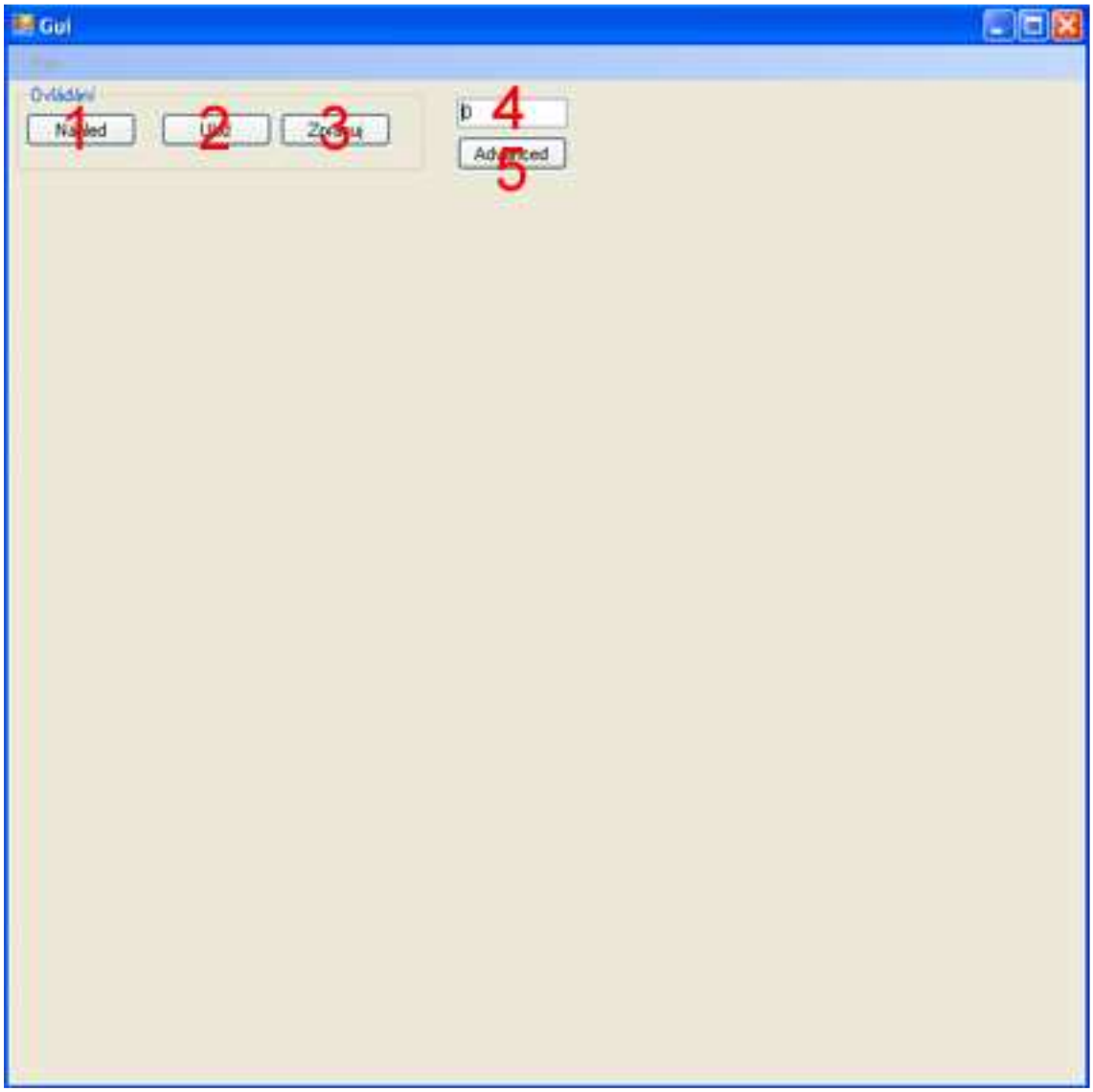
Při běhu programu může dojít k několika chybovým stavům. Jednotlivá chybová hlášení jsou popsána dále:

1. „Kamera byla odpojena“ toto hlášení se oběví, pokud program během režimu „Náhledu“ nebo „Ukládání“ detekuje odpojení kamery (to může znamenat buď, že se kamera uspala, nebo byla odpojena od počítače). Po opětovném připojení kamery lze pokračova v běhu programu.

2. „Soubor nelze zpracovat“ toto hlášení se oběví, pokud uživatel stiskne tlačítko Zpracovat a před tím neprovedl krok 3 uložení videa (viz víše). Nebo pokud při zpracování uloženého souboru došlo k chybě.

3. „Program nenalezl všechny soubory nutné pro zpuštění“ toto hlášení se objeví při spuštění aplikace, pokud program nenalezne na soubory s obrázky pozadí a textur světelného meče.

4. „V programu se vyskytla chyba, program bude ukončen“; toto hlášení oznamuje ostatní chyby



Obr. 12: Grafické uživatelské rozhraní programu s očíslovanými jednotlivými prvky: 1 tlačítko „Náhled“, 2 tlačítko „Ulož“, 3 tlačítko „Zpracuj“, 4 políčko „Počet snímků za sekundu“, 5 tlačítko „Advanced“ (popis jednotlivých prvků viz uživatelská příručka)

Obr. 13: Ukázka aplikace v režimu „Náhled“

Následuje popis „Advanced“ uživatelského rozhraní. Ovládací prvky tohoto rozhraní lze rozdělit do tří kategorií.

Do první kategorie patří prvky, které slouží k sestavení tzv. „ukázkového grafu“. Tento graf načítá vstupní data z uživatelem zadaného souboru, zpracovává je pomocí filtru BlueScreenFilter a následně je vykresluje do okna aplikace, nebo ukládá na pevný disk počítače. Ukázkový graf lze sestavit a spustit následujícím způsobem. Nejprve je nutné vybrat soubor, který má být zpracován. To lze provést pomocí stisknutí položky menu Main/file/load (viz obr 14 číslo 20). Další věc, kterou je nutné udělat, je nastavit, zda se mají data vykreslovat ,či ukládat. Tento výběr lze provést pomocí zaškrtávacího políčka „render“ (viz obr. 14 číslo 21). Pokud je políčko zaškrtnuté, data se vykreslují,

jinak se ukládají. Po výběru zdrojového souboru a určení, zda se mají data vykreslovat, lze graf spustit pomocí položky menu Main/Strart (viz obr14 číslo 1).

Druhou skupinou jsou ovládací prvky, které slouží primárně pro kontrolu běhu „ukázkového grafu“ (některé z nich lze použít i při ovládaní ostatních grafů viz dále). Následuje popis jednotlivých prvků této skupiny (prvky jsou číslovány dle obr. 15) :

7. tlačítko „Run“; pomocí tohoto tlačítka lze spustit graf který byl před tím zastaven (viz tlačítko 8), či pozastaven (viz tlačítko 9). Pokud graf běží, či není sestaven žádný graf, stisknutí toho tlačítka nemá žádný účinek.
8. tlačítko „Stop“; stisknutí tohoto tlačítka zastaví běžící, či pozastavený graf. Opět pokud je aktuální graf zastaven, či není sestaven žádný graf, nemá stisknutí tohoto tlačítka žádný vliv.
9. tlačítko „Pause“; stisknutí tohoto tlačítka pozastaví běžící, či zastavený graf. Opět pokud je aktuální graf pozastaven, či není sestaven žádný graf, nemá stisknutí tohoto tlačítka žádný vliv.
10. tlačítko „Step>“; stisknutí tohoto tlačítka posune graf na následující snímek. Pokud graf před stisknutím tohoto tlačítka běžel, je pozastaven.
11. tlačítko „Step<“; stisknutí tohoto tlačítka posune graf na předchozí snímek. Pokud graf před stisknutím tohoto tlačítka běžel, je pozastaven.
12. tlačítko „Refresh“; stisknutí tohoto tlačítka způsobí znovu vykreslení aktuálního snímku. Pokud graf před stisknutím tohoto tlačítka běžel, je pozastaven.
13. tlačítko „Resolution“; stisknutím tohoto tlačítka lze přecházet mezi plným a polovičním rozlišením výstupního videa.
14. „TrackBar“ pomocí tohoto ovládacího prvku lze nastavovat aktuální pozici ve videu.

Zde je nutné zmínit, že ovládací prvky 10 až 14 jsou aktivní pouze při použití „ukázkového grafu“ a vykreslování videa do okna aplikace. Pro všechna ostatní nastavení jsou tyto prvky neaktivní.

Třetí skupinou ovládacích prvků jsou prvky, pomocí nichž lze přímo nastavovat vlastnosti procesů maskování na modré pozadí a detekce bodu. Do této skupiny patří (číslováno dle obr. 16) :

15. ovládací prvek označený „Treshold“ . Pomocí tohoto prvku lze nastavovat hodnotu rozdílu mezi modrým kanálem a větším ze dvou zbylých, kterou musí pixel dosáhnou, aby byl při detekci bodů označen jako „modrý“.

16. ovládací prvek „Green Spill“. Pomocí tohoto prvku lze natavit o kolik může zelený kanál přesahovat modrý kanál (viz část 5.2.3 popis funkce DifferenceMatte2) .
17. ovládací prvek „Křivka. Tento nástroj nastavuje transformaci hodnot masky pro všechny pixely obrazu a výrazně tak ovlivňuje kvalitu výsledného kompozitu. Prvek funguje na podobném principu jako nástroj Křivka ve Photoshopu. Na vodorovná osa představuje vstupní hodnoty masky (v rozmezí 0..1). Svislá osa představuje hodnoty masky po transformaci (opět v rozmezí 0..1). Červená křivka, jejímiž koncovými body lze pohybovat pomocí myši, pak určuje transformaci vstupních hodnot na vodorovné ose na výstupní hodnoty svislé ose.
18. zaškrtávací políčko „useFilter“ určuje, zda má být při vytváření masky použit Gausovský filter (viz část 5.2.3 popis funkce FilterAlpha5x5) .
19. zaškrtávací políčko „mate“ určuje, zda má být při zpracování dat zobrazena maska (pokud je políčko zaškrtnuté), či zda se má generovat kompozit.

Pro všechny tyto prvky platí, že jejich nastavení je platné pouze v rozšířeném módu aplikace a po přepnutí do standardního módu se všechny parametry na staví na výchozí hodnoty.

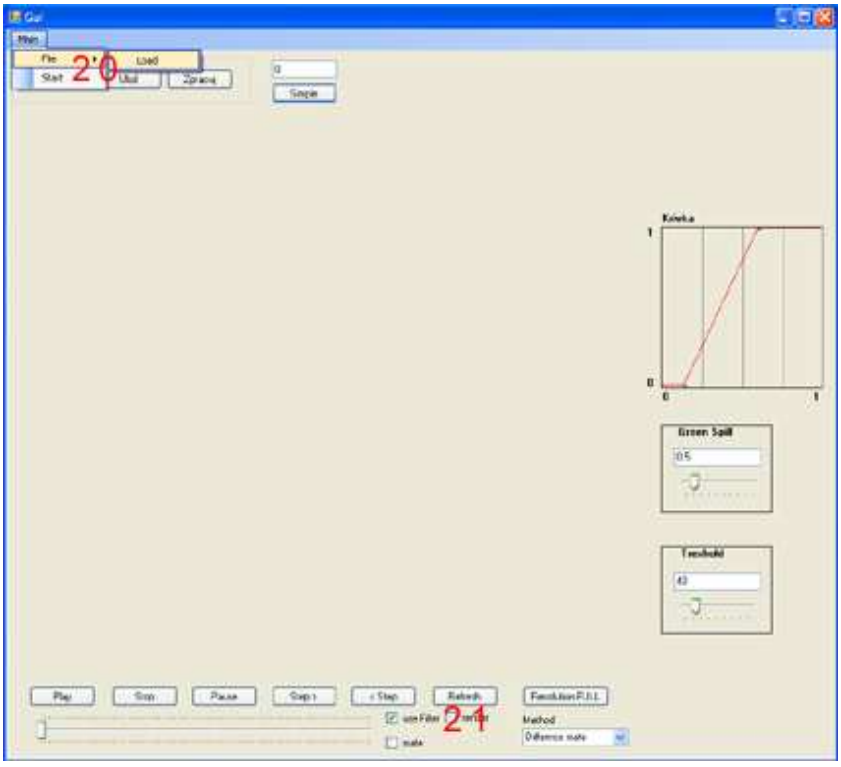
V poslední části uživatelské příručky popíši, kdy se sestavují a rozpojují jednotlivé grafy (určeno pouze pro zkušené uživatele). Aplikace obsahuje tři různé typy grafů. Jejich sestavování a rozpojování se provádí v přesně určenou chvíli a má některé důsledky, které musí vzít uživatel pracující v rozšířeném módu na vědomí.

První typ grafu je určen pro „náhled v reálném čase“. Tento graf je zpuštěn stisknutím tlačítka „Náhled“ a běží, dokud není sestaven jiný graf, nebo není stisknuto tlačítko „Stop“, nebo není odpojena kamera, ze které jsou data získávána. Při vytvoření jiného grafu je tento graf zastaven a rozpojen (to platí i pro všechny ostatní grafy). Stejné operace jsou provedeny i v případě, že je dekováno odpojení kamery. Po stisknutí tlačítka „Stop“ či „Pause“ je graf pouze zastaven respektive pozastaven a je možné jej znovu spustit tlačítkem „Run“.

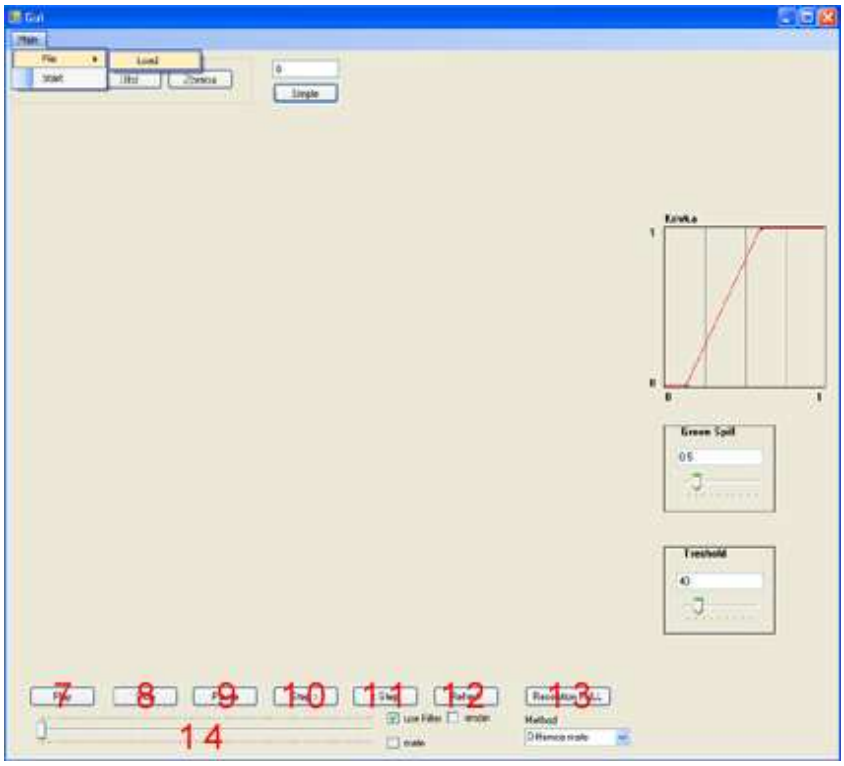
Další typ grafu je určen pro ukládání „nezpracovaného“ videa na pevný disk. Tento graf je spuštěn tlačítkem „Uložit“ a běží po dobu 30 vteřin. Po uplynutí této doby je graf zastaven a rozpojen. Při použití tlačítek „Stop“, „Pause“ a „Run“ platí stejná pravidla jako pro předchozí graf. Je ale nutné si uvědomit, že časový limit běží bez

ohledu na to, zda byl graf zastaven, či běží bez přerušení, a graf je vždy po 30 vteřinách rozpojen.

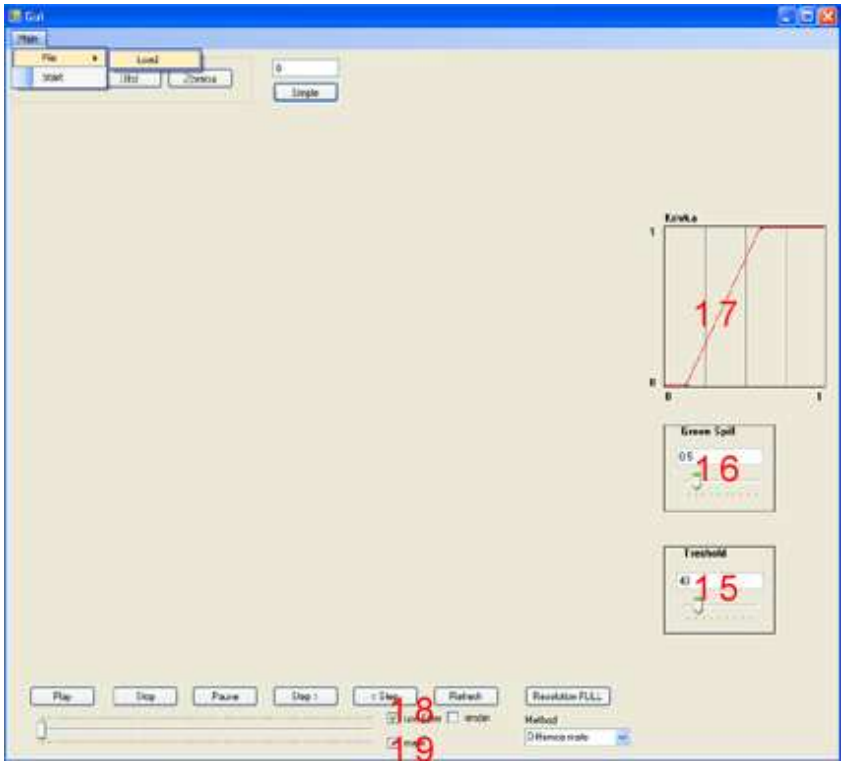
Poslední typ grafu zpracovává data z pevného disku počítače. Tento graf lze spustit tlačítkem „Zpracuj“ nebo v rozšířeném módu pomocí menu „Main“(viz předchozí část uživatelské příručky). Tento graf běží do doby, než zpracuje všechna data ze vstupního souboru, nebo není zastaven pomocí tlačítek „Stop“ či „Pause“. Pro případ zastavení či pozastavení grafu pomocí příslušných tlačítek platí pravidla popsaná v předchozích odstavcích. Je nutné si uvědomit, že bez ohledu na to zda jsou data ukládána na disk, či vykreslována do okna aplikace (v rozšířeném módu), po zpracování všech dat je graf rozpojen.



Obr. 14: Ukázka rozšířeného uživatelského rozhraní s označenými prvky pro sestavení a zpuštění „ukázkového grafu“



Obr. 15: Ukázka rozšířeného uživatelského rozhraní s označenými prvky pro kontrolu jednotlivých grafů



Obr. 15: Ukázka rozšířeného uživatelského rozhraní s označenými prvky pro kontrolu parametrů procesů maskování na modré pozadí a detekce bodu

B. Překlad

Přiložené CD obsahuje zdrojové soubory spolu se „solution file“ Visual Studia 2005 (dále jen VS) BlueScreen.sln.Pomocí tohoto souboru lze snadno spustit příslušný projekt a po té zdrojové soubory přeložit ve VS 2005. Před překladem je nutné nainstalovat DirectX SDK a DirectShow SDK a nastavit ve VS cesty k jejich knihovnám a hlavičkovým souborům. Dále je nutné přeložit DirectShow BaseClasses (pro přeložení debug verze programu jejich debug verzi, analogicky pro release verzi) a nastavit ve VS cesty k přeložené knihovně (knihovnám)a hlavičkovým souborům DirektShow BaseClasses. Ostatní nastavení jsou již obsažena v projektu kterého jsou zdrojové soubory součástí (obě konfigurace Debug i Release jsou otestované a plně funkční). Některé verze VS generují při překladu programu varování číslo 4793 oznamující, že je generován neřízený kód. Neřízený kód je generován zcela záměrně a toto varování by se nemělo zobrazovat. Jedná se o chybu ve VS (ověřeno přímo u vývojového týmu VS) a proto by toto hlášení mělo být ignorováno.