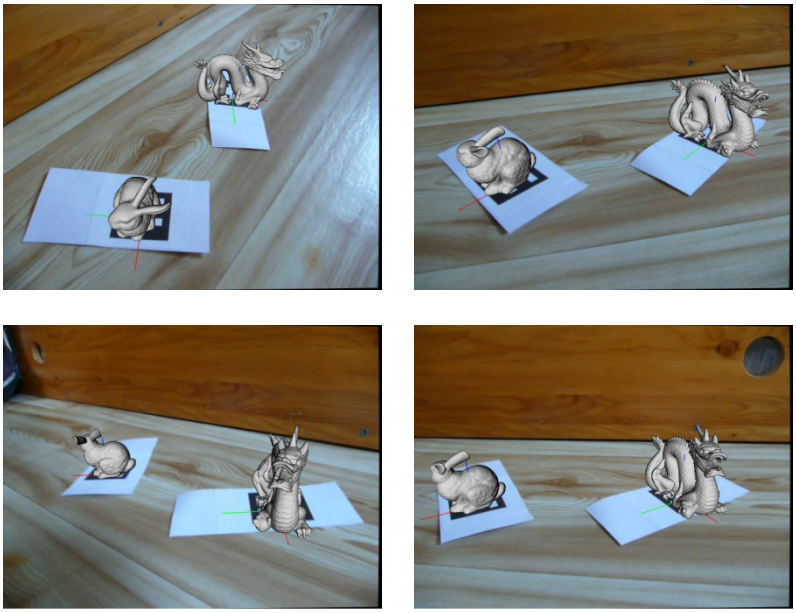
Zpracování multimediálních dat

Podpůrný text pro tutoriál:

Virtuální realita a její aplikace – rozšířená realita



Obrázek 1: Ukázka video sekvence s doplněným 3D modelem (ukázky z výsledné aplikace)

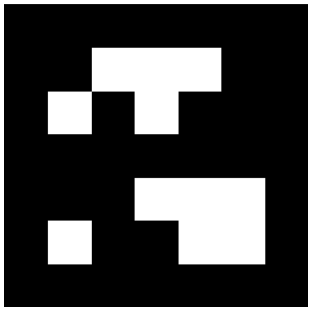
Jednou z praktických aplikací z oblasti virtuální reality je tzv. rozšířená realita. Cílem této metody je rozšíření reálného obrazu o virtuální prvky, jakými mohou být počítačem generovaný obraz, zvuk nebo např. GPS data. V tomto tutoriálu se seznámíme s postupem pro vytvoření aplikace, která do reálného záběru kamery umístí virtuální předmět na místo označené speciální značkou. Tato značka (dále marker) umožní určit typ i orientaci objektu, který chceme do scény virtuálně umístit.

V následujících bodech shrneme jednotlivé kroky vedoucí k umístění 3D modelu předmětu do video sekvence. Vzhledem k tomu, že se předpokládá vypracování tohoto tutoriálu v prostředí jazyka C++ a s využitím knihovny OpenCV, jsou u jednotlivých kroků uvedeny také názvy odpovídajících funkcí, řešících daný problém. Detailní popis těchto funkcí je nad rámec tohoto textu a čtenář je odkazován na podrobnou dokumentaci knihovny OpenCV.

Postup:

1. Detekce markerů v obrazu kamery

Předpokládejme, že se nám podařilo úspěšně získat snímek z připojené kamery a pokusíme se v něm nalézt příslušný marker. Markerem rozumíme černý čtverec s vycentrovanou mřížkou tvořenou černobílou šachovnicí (např. 5x5 čtverců), která reprezentuje informaci v binárním kódu (černá odpovídá hodnotě 0 a bíla hodnotě 1).



Obrázek 2: Ukázka značky (markeru) o rozměrech 7x7

Pro následnou snadnější manipulaci s obrazem, tento nejprve převedeme z barevného prostoru do stupňů šedi (cv::cvtColor). Dále je nutné obraz binarizovat pomocí prahování. Jednoduché prahování nebude v našem případě fungovat, protože obraz bude vykazovat různé úrovně jasu z důvodu problematických světelných podmínek. Lepší volbou bude adaptivní prahování (adaptiveThreshold) pracující s průměrnými hodnotami v určitém okolí každého pixelu. Výsledkem by měl být černobílý obrázek, na kterém budou dobře viditelné všechny části markerů (tzn. jejich obvod i vnitřní vzor) a případně další nepotřebné.

2. Detekce kontur

Nyní potřebujeme nalézt samotný marker v černobílém obraze, který jsme získali v předchozím kroku. Jelikož oblast markeru představuje černý čtverec, je nejjednodušší tuto oblast aproximovat pomocí polygonu o čtyřech vrcholech. Nalezení kontur řeší metoda cv::findContours a našim úkolem je ze získaných kontoru vhodným způsobem odfiltrovat chybné detekce. Před samotným filtrováním je nutné konturu nahradit polygonem (cv::approxPolyDP). Platný polygon musí mít právě čtyři vrcholy, musí být konvexní (cv:isContourConvex) a vzdálenost každých dvou vrcholů polygonu musí být větší než vhodně zvolená prahová hodnota. Tímto krokem jsme získali kandidáty na markery a musíme dále ověřit, zdali jsou alespoň některé ze získaných polygonů skutečně konturami markerů.

3. Ověření potenciálních markerů

Pro získání obsahu jednotlivých polygonů, je nutné nejprve odstranit perspektivní projekci a získat jejich obraz z čelního pohledu. Toho docílíme pomocí transformační matice, kterou vypočteme pomocí funkce cv::getPerspectiveTransform na základě znalosti čtyř korespondujících párů bodů.

První čtveřice bodů jsou souřadnice nalezených vrcholů polygonu v obraze a druhá čtveřice představuje body v lokálním souřadném systému samotného markeru. Samotný warping obrazu provedeme pomocí funkce cv::warpPerspective. Zdrojovým obrazem pro warping je originální barevný snímek z kamery a výsledný obraz potencionálního markeru je tedy také barevný obrázek. Pro čtení binárních symbolů je pro nás vhodnější jeho černobílá reprezentace. Tu získáme prahováním pomocí Otsuova algoritmu (cv::threshold(… | CV_THRESH_OTSU), který předpokládá bimodální rozložení jasů v obraze (to je přesně náš případ) a snaží se nalézt takovou hodnotu prahu, která maximalizuje varianci mezi třídami při zachování co nejmenší variance uvnitř tříd.

Po provedeném prahování se můžeme pokusit odečíst binární hodnoty z příslušných míst obrazu markeru a získat tak jeho kód. Každý „bit“ markeru je reprezentován jedním ze 49 čtverců rozmístěných v pravidelné mřížce nad každým markerem (viz. Obrázek 2). Předpokládejme, že obvodové bity jsou vždy 0, tj. černé a užitečnou informaci nese pouze vnitřní mřížka 5x5 bitů. Pro převedení hodnot pixelů jednotlivých bitů markeru na bity lze využít funkce cv::countNonZero nad příslušným regionem obrazu markeru (tzv. ROI).

4. Rozpoznání markerů

Jednou z možností, jak zapsat požadovanou informaci (typ objektu a orientaci) do markeru, je přímo zakódovat např. nějaké ID do 25-bitové hodnoty. To však není zcela idální, protože např. taková hodnota ID=0 by vedla na černý čtverec, ze kterého by evidentně nebylo možno získat informaci orientaci markeru. Rovněž symetrické vzory by nám neumožnily jednoznačně zrekonstruovat jeho orientaci. Dále také nemůžeme spoléhat na to, že se nám vždy podaří každý marker rozpoznat zcela bezchybně. Z těchto důvodů je vhodné hodnoty markerů zakódovat např. pomocí lineárního Hammingova kódování pro opravu jedné chyby.

5. Přesné dohledání poloh markerů v obraze

Po dekódování ID markeru a určení jeho správné orientace, můžeme dohledat jeho vrcholy se subpixelovou přesností pomocí funkce cv::cornerSubPixel. Tento krok nám zajistí co nejpřesnější nalezení umístění vrcholů markeru v obraze a následně zpřesní odhad jeho umístění v 3D prostoru. Tento krok je poměrně časově náročný, a proto je zařazen až zde, jelikož teď již pracujeme jen s malým počtem markerů.

6. Poloha markeru v 3D prostoru

Po provedení předcházejícího kroku známe 2D polohy rohů ($\mathbf{p}_s$) nalezených markerů v obrazovém prostoru. Jejich umístění v 3D světovém prostoru ($\mathbf{p}_w$) a korespondující projekce do 2D obrazového prostoru je svázána rovnicí

$$\mathbf{p}_s = \mathbf{A}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{p}_w\,, \qquad (1)$$

kde **A** představuje tzv. vnitřní matici kamery popisující parametry jako ohnisková vzdálenost, hlavní bod kamery a geometrickou distorzi objektivu kamery. Rotační matice **R** o rozměrech 3x3 spolu s translačním 4D vektorem **t** v homogenních souřadnicích představuje Euklidovskou transformaci. Z rovnice (1) známe pouze polohy rohů $\mathbf{p}_s$ (můžeme si určit libovolně) a $\mathbf{p}_w$ () vše ostatní musíme určit.

Kalibrací kamery se zde zabývat nebudeme, jen poznamenáme, že knihovna OpenCV je vybavena funkcemi pro získání matice **A** ze série 10 – 20 snímků šachovnice. Pro tuto chvíli předpokládejme, že matice **A** je nám pro naši kameru k dispozici a můžeme pokračovat dále.

Procedura nalezení odhadu transformace mezi prostorem kamery a globálním prostorem, ve kterém je umístěn sledovaný marker, se nazývá odhad pózy z 2D-3D korespondence a jejím výsledkem je nalezení matice Euklidovské transformace [**R**|**t**]. Tuto korespondenci lze snadno nalézt pomocí funkce cv::solvePnP vracející obě komponenty uvažované Euklidovské transformace, je však nutné si uvědomit, že nalezená transformace popisuje pozici kamery vůči markeru ve světovém prostoru. Pro další zpracování je vhodné získat inverzní transformaci.

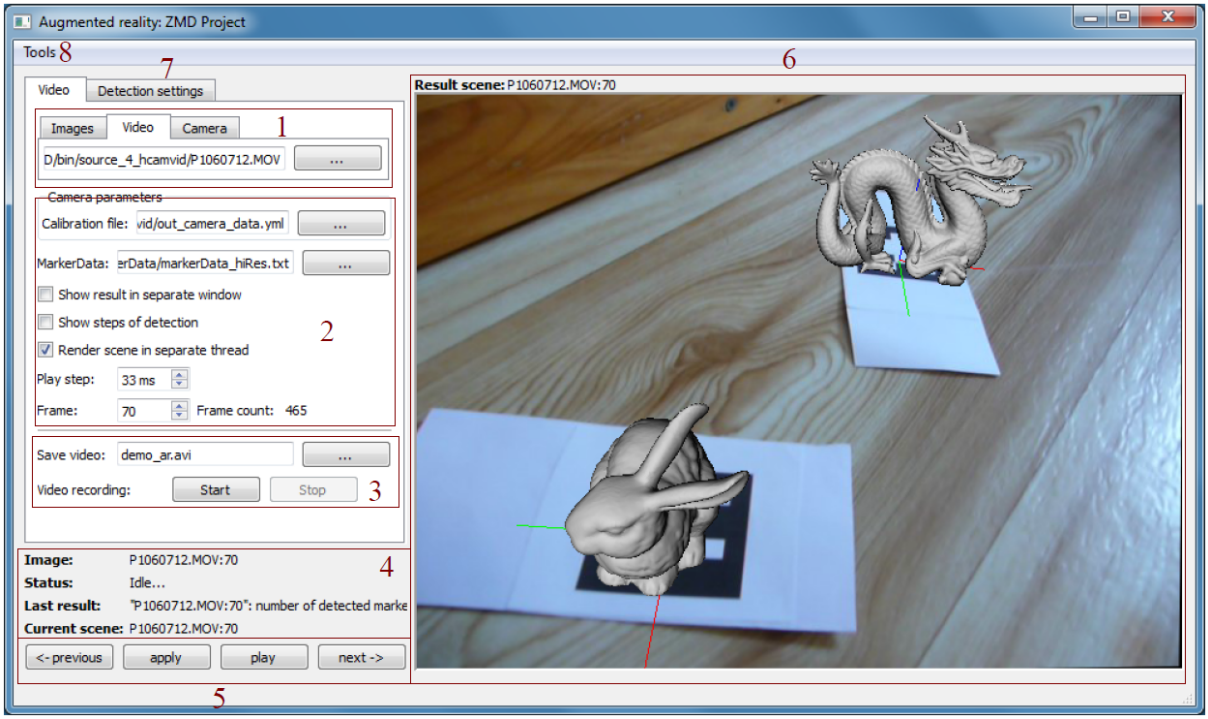
7. Doplnění virtuálního modelu do reálného obrazu

Posledním krokem je s využitím získaných transformací provést vykreslení modelu do obrazu kamery. Toho je možné dosáhnout např. pomocí knihovny OpenGL.

Následujícím cílem je uvedený postup prakticky realizovat v jazyce C++ za využití knihovny OpenCV. Jako návodný příklad může posloužit následující úspěšná implementace uvedeného postupu.

Příklad výsledné aplikace:

Autoři: Tomáš Bartošek, Jan Křístek, Peter Drábik, školní rok 2012/2013



Obrázek 3: Ukázka výsledné aplikace

Popis jednotlivých ovládacích prvků hlavního okna aplikace (viz. Obrázek 3):

1. Zdroj snímků: zde se nastavuje, co bude zdrojem snímků pro detekci markerů. Přepínáním mezi záložkami (tabulátorem) se nastavuje typ zdroje:
   - Obrázky: vytvoří seznam obrázků z disku. Lze vybrat soubory pouze z jedné složky. Pomocí klávesy ctrl lze vybrat selektivně nebo pomocí ctrl+a lze vybrat všechny soubory.
   - Video: video soubor. Pokud obsahuje více proudů, bude použit první a zvuková stopa je kompletně ignorována.
   - Kamera: připojená kamera k PC (např. webkamera). Parametrem je číslo, které je použito pro metodu cv::VideoCapture::open().

2. Nastavení detekce a vykreslování: nastavuje různé nastavení pro vykreslování a detekci. Dále je zde možno nastavit kalibrační soubor kamery, který je potřeba pro správnou detekci. Také je zde možno nastavit soubor s metadaty ohledně markerů, případně nechat zobrazit výsledky v samostatném okně a průběh jednotlivých kroků detekce.

3. Nahrávání do souboru: umožňuje nahrát renderované snímky do videa. Je použit MJPG kodek. Po kliknutí na „Start“ se budou průběžně ukládat všechny vyrenderované snímky až do stisknutí „Stop“.

4. Informace o průběhu: zobrazuje informace o tom, který snímek se detekuje, status vlákna detekce apod.

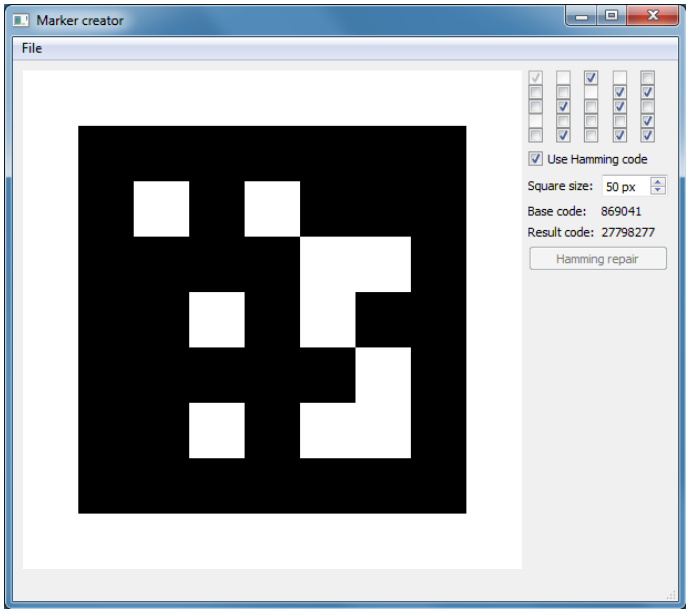
5. Renderování: ovládá renderování. Prostřednictvím tlačítek „previous“ a „next“ se lze přepínat mezi jednotlivými snímky, které se ihned detekují, renderují a zobrazí. Kliknutím na play se automaticky budou přehrávat jednotlivé snímky. Pozn.: tlačítka „previous“ a „next“ si pamatují snímek, který byl renderován naposled. Přehrávání je ale ignorováno. Snímek, kde

se má přehrávání začít a ke kterému se vztahují tlačítka „previous“ a „next“ lze nastavit v položce „Frame“ v sekci 2.

6. Výsledek renderování: vizualizuje výsledek detekce a renderování.

7. Parametry detekce: Zde lze nastavit parametry OpenCV funkcí použité pro renderování.

8. Položka nástroje: obsahuje nástroj pro tvorbu markerů. Je možné si naklikat marker a nechat zobrazit jeho kód (viz. Obrázek 4). Marker lze uložit jako obrázek.



Obrázek 4: Editor markerů

Kalibrace kamery

Aby bylo možno provést detekci markerů správně, je potřeba zjistit parametry kamery, tj, kalibrovat ji. Kalibrace probíhá tak, že se nejprve kamerou zachytí několik (10 – 20) snímků šachovnice známých rozměrů, a poté se nechá proběhnout výpočet, který dané parametry zjistí. Ukázka aplikace pro kalibraci kamery (viz. Obrázek 5) pochází z diplomové práce Tomáše Bartoška. Program umožňuje kalibraci provést pomocí videa nebo samostatných obrázků. V pravé části je možno nastavit parametry kalibrace včetně velikosti šachovnice.

Metadata markerů

Program rozšířené reality používá textové dokumenty s metadaty markerů pro jejich detekci a vykreslení správného modelu.

Formát metadat:

- Textový formát, lze libovolně oddělovat bílými mezerami s jednou výjimkou
- Když je řečen řetězec, je myšlen bez uvozovek a bez bílých mezer
- Struktura dat jednoho markeru:
    - Uvozen symbolem ">" (bez uvozovek)

- Řetězec: jméno markeru
- Číslo: indikátor, zda marker používá hammingovo kódování (1 - ano, 0 – ne)
- Číslo: zakódovaný kód markeru; v případě hammingova kódování se vynechávají paritní bity (číslo spotřebovává tedy 20 bitů místo 25) – lze získat z nástroje pro tvorbu markerů
- následují volitelné parametry:
- model formátu ply:
  - řetězec: "PLY_MODEL" (bez uvozovek)
  - adresa souboru; jako adresa se bere následující celý zbytek řádku, takže může obsahovat bílé znaky

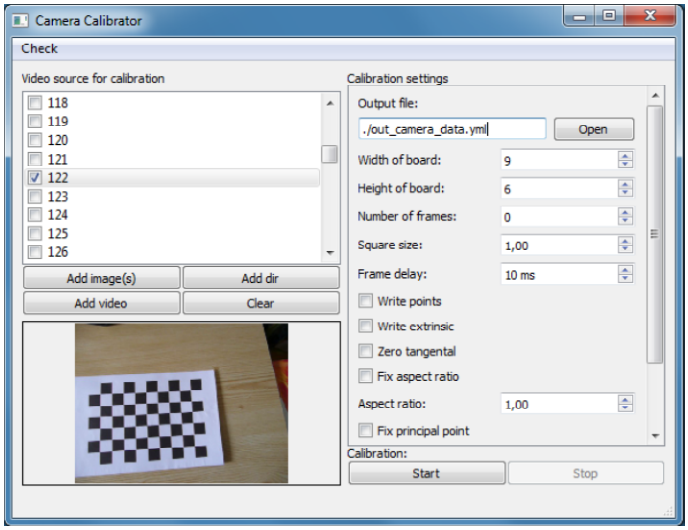
- Neuvádí se počet markerů ani ukončovací token, ale je potřeba označit všechny markery symbolem ">".

Příklad:

```
> marker_01
0 22757246
PLY_MODEL markerData/dragon_vrip_res4.ply

> marker_03
1 1040743
PLY_MODEL markerData/dragon_vrip_res4.ply

> marker_04
1 1043519
PLY_MODEL markerData/bun_zipper_res4.ply
```



Obrázek 5: Kalibrace kamery

Postup použití aplikace:

1. Je potřeba nasnímat šachovnici pro kalibraci a snímky pro samotné markery.
2. Kalibrace kamery.
3. Vytvoření souborů s metadaty markerů.
4. Spuštění hlavního programu a spuštění rozšířené reality:

a) Ve zdroji snímků nastavit video.
b) Nastavit kalibrační soubor kamery.
c) Nastavit soubor s metadaty markerů.
d) Je doporučeno nechat renderování v samostatném vlákně. Volitelně je možné nastavit zobrazování výsledků v samostatném okně a případně zobrazování průběhu detekce.
e) Volitelně lze nastavit výstupní video. Kliknutím na „Start“ se připraví jeho nahrávání.
f) Použitím tlačítek „apply“, „previous“ a „next“ lze postupně nechat zpracovat jednotlivé snímky nebo tlačítkem „play“ přehrát celé video.
g) Jestliže jsme nahrávali video s výslednými snímky, kliknutím na „Stop“ se nahrávání dokončí a výsledný video soubor bude možné přehrát.

Použitá literatura:

[1] Baggio, Daniel a kol. Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing. ISBN 978-1-84951-782-9. 2012.