Rychlá 3D grafika pro Matlab

Jan Černík, Tomáš Pajdla
České vysoké učení technické, fakulta elektrotechnická
Centrum strojového vnímání
121 35 Praha 2, Karlovo náměstí 13, Česká Republika
cernik@cmp.felk.cvut.cz

Abstrakt

Tento příspěvek popisuje řešení problému zobrazení 3D scény, reprezentované množinou referenčních 2D snímků a informací o jejich korespondenci, z libovolného bodu v prostoru.

Syntéza 3D modelu scény pouze na základě množiny 2D intenzitních snímků je těžký problém, jehož obecné řešení zatím nebylo nalezeno. Při řešení je nutné využít kromě referenčních snímků také informaci o jejich korespondenci. Tato práce využívá výsledků výzkumu v rámci Centra strojového vnímání a soustřeďuje se na efektivní metody zobrazení výsledné 3D scény.

Jádrem příspěvku je popis nového toolboxu, nazvaného Geomview Server, který umožňuje spolupráci mezi Matlabem 5.x a 3D zobrazovacím programem Geomview. Prostřednictvím tohoto toolboxu se dostává uživateli do rukou výkonný, plně programovatelný nástroj pro vizualizaci rozsáhlých 3 i více rozměrných objektů. Zobrazované objekty jsou popsány speciálním, objektově orientovaným programovacím jazykem OOGL (Object Oriented Graphic Language). Kromě popisu modelu, textury a jiných atributů je možné programově řídit např. polohu objektu, osvětlení a vytvářet tak animované sekvence v reálném čase. Geomview Server zajišťuje také zpětnou vazbu, tj. interaktivně přenáší informace o vybraných objektech zpět do Matlabu.

Celý toolbox je určen pro prostředí UNIX (Linux a SGI IRIX).

Úvod

Na příkladu motivačního problému, kterým je syntéza 3D modelu scény ze sekvence 2D snímků, představím nový toolbox Geomview Server, určený pro rychlé a interaktivní zobrazení 3 a více rozměrných objektů. Text článku je logicky rozdělen do dvou částí:

- motivační problém: syntéza 3D modelu scény,
- popis toolboxu: Geomview Server.

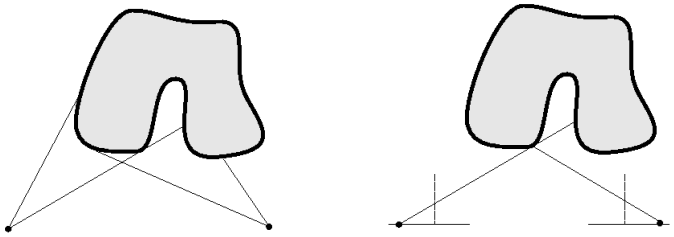
Motivační problém

Syntéza 3D modelu scény

Základem algoritmu je vstupní množina snímků. Kvalita a způsob sejmutí snímků jsou pro úlohu syntézy velmi důležité. Podle velikosti scény lze využít 2 rovnocenných postupů:

- statická scéna, pohybující se kamera,
- statická kamera, dynamická scéna.

Frekvence, s jakou získáváme snímky scény, je libovolná. Existuje ovšem přímý vztah mezi složitostí řešené úlohy a frekvencí snímků. Čím vyšší je počet snímků za jednotku času, tím menší je změna scény mezi jednotlivými snímky. Optimum představuje množina snímků sejmutá s frekvencí blízkou k nekonečnu, pak lze problém zobrazení 3D scény převést na výběr snímku, který zachycuje scénu z požadovaného místa. Dalším důvodem, proč snímat scénu s co možná největší snímkovou frekvencí, je problém samozakrytí objektů na scéně (obr. 1). Pokud je scéna snímána s nízkou frekvencí a navíc dochází k samozakrytí objektů, mění se charakter scény mezi jednotlivými snímky kvalitativně, skokem a úloha interpolace se stává prakticky neřešitelnou.



Obr. 1: Samozakrytí a Nejednoznačnost bodů

Popisovaná metoda vyžaduje kromě vstupní množiny snímků také informaci o korespondenci bodů. Přestože algoritmy pro nalezení korespondence nejsou předmětem tohoto článku, je snadné ukázat, že samozakrytí znemožňuje správnou funkci libovolně složitého algoritmu pro korespondenci. Na obrázku (obr. 1) je vidět, že bodu sledovanému na pravém snímku odpovídají na levém snímku 2 body.

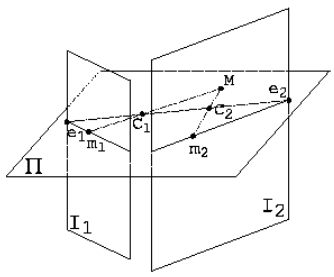
Syntéza 3D modelu slouží v rámci tohoto článku pouze jako motivační problém, a proto popíši zjednodušené řešení, které pro syntézu využívá pouze 2 referenčních snímků a informaci o jejich korespondenci.

Schématicky lze funkci algoritmu zapsat takto:

- **Vstup:** 2 referenční snímky scény, bodová korespondence referenčních snímků.
- **Zpracování:** rekonstrukce hloubky, syntéza 3D modelu.
- **Výstup:** úplný objemový 3D model, včetně textury.

Algoritmus syntézy 3D modelu

Jak bylo řečeno výše, vstupem algoritmu jsou 2 referenční snímky a souřadnice korespondujících bodů. Situace viz obr. 2. Na souřadnice korespondujících bodů je aplikována Delaunayho triangulace.



Obr. 2: Epipolární geometrie

Intenzitní funkci v referenčním resp. virtuálním snímku (novém pohledu na syntetizovaný model scény) budu značit $f(\mathbf{u})$ resp. $f'(\mathbf{u}')$. Předpokládám, že intenzity korespondujících bodů v referenčním a virtuálním snímku jsou shodné, tj.

$$f'(\mathbf{u}') = f(\mathbf{u}).$$

Intenzity je možné kopírovat mezi referenčním a virtuálním obrazem bod po bodu. Je samozřejmě nutné aplikovat příslušnou transformaci. Aby ve virtuálním snímku nevznikly diskontinuity vlivem rozdílných vzorkovacích frekvencí, je třeba provádět zpětné mapování, tj. procházet postupně body jednotlivých trojúhelníků ve virtuálním obrazu a hledat jim vzory zpětně v referenčním obrazu.

Efektivnější metodou je transfer celých trojúhelníků vzniklých triangulací mezi referenčním a virtuálním obrazem. Nejprve přeneseme vrcholy a pak výplň jednotlivých trojúhelníků. Pracujeme-li s akcelerovaným grafickým subsystémem, je právě trojúhelník základní jednotkou pro mapování textur a z-bufferingu. Většinu operací tak přebírá hardware.

Obecně je mezi texturou trojúhelníka v ref. a virtuálním snímku vztah homografické transformace $\mathbf{u}' \approx \mathbf{H}\mathbf{u}$ (mapování rovina→rovina, bod→bod). Pro intenzitu virtuálního bodu platí

$$f'(\mathbf{u}') = f(\mathbf{H}^{-1}\mathbf{u}) . \qquad (1)$$

Viditelnost bodů je s výhodou řešena z-bufferingem. Souřadnice hloubky z bodu $\mathbf{u}'$ je aproximována funkcí vzdálenosti

$$d(\mathbf{u}) = |\mathbf{u} - \mathbf{e}| .$$

Pro transformaci hloubky do virtuálního obrazu platí (viz 1)

$$d(\mathbf{u}') = d(\mathbf{H}^{-1}\mathbf{u}') . \qquad (2)$$

Pro výpočet matice $\mathbf{H}$ je třeba 4 bodů: 3 vrcholy trojúhelníka a epipól (viz obr. 2) původní dvojice snímků. Matici $\mathbf{H}$ však lze aproximovat, potom pro intenzitu bodu ve virtuálním snímku platí

$$f'(\mathbf{u}') = f(\mathbf{u}) = f(\alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 + \alpha_3\mathbf{u}_3) , \qquad (3)$$

kde hodnoty α_i, i = 1, 2, 3 získáme vyřešením soustavy

$$\begin{aligned} \alpha_1\mathbf{u}'_1 + \alpha_2\mathbf{u}'_2 + \alpha_3\mathbf{u}'_3 &= u', \\ \alpha_1 + \alpha_2 + \alpha_3 &= 1. \end{aligned}$$

Hodnotu $d(\mathbf{u}')$ lze získat podle vztahu 2 takto

$$d(\mathbf{u}') = d(\mathbf{u}) = d(\alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 + \alpha_3\mathbf{u}_3) . \qquad (4)$$

Aby nebylo nutné vyčíslovat $d(\mathbf{u}')$ v každém bodu trojúhelníka, je výhodnější vztah 4 aproximovat:

$$d(\mathbf{u}') = \alpha_1 d(\mathbf{u}_1) + \alpha_2 d(\mathbf{u}_2) + \alpha_3 d(\mathbf{u}_3) . \qquad (5)$$

Analýza problému

Cílem popsané motivační úlohy je sestavit 3D model scény, zobrazit jej, umožnit uživateli s ním volně manipulovat a získat tak nové, virtuální pohledy na scénu. Navíc jsou na zobrazovací systém kladeny tyto požadavky:

- **flexibilita:** systém musí být navržen tak, aby bylo možné za běhu měnit maximum parametrů, např. počet virtuálních kamer, osvětlení scény, parametry geometrických objektů: materiál, stínování, texturu apod.,
- **odezva v reálném čase:** dostatečně rychlá odezva, aby bylo možné např. simulovat průlet virtuální kamery kolem scény,
- **přenositelnost:** systém musí být přenositelný mezi různými platformami a operačními systémy, zejména PC-Linux a SGI-IRIX,
- **otevřenost:** jasná architektura systému, aby v případě potřeby bylo snadné doplnit dodatečné funkce nebo modifikovat systém tak, aby byl schopen spolupracovat i s programy, které v době návrhu neexistovaly,
- **interaktivita:** zobrazovací systém, musí být přístupný okamžitým zásahům a experimentování uživatele,
- **zpětná vazba**: systém musí být schopen nejen akceptovat data na vstupu, ale také zpětně poskytovat informace o svém stavu, výsledcích interakce s uživatelem a jiných neočekávaných událostech.

Navržené řešení

Proč pro vyřešení problému nepoužít pouze prostředků, které nabízí Matlab ? Problémů je několik, zejména triangulace a zobrazení výsledné scény.

Triangulace

Matlab ve verzi 5.x konečně obsahuje funkce pro 2D triangulaci. Přestože je lákavé využít této funkce, nesplňuje požadavky kladené řešenou úlohou. Funkce Matlabu neumožňují specifikovat žádné omezení na výsledek triangulace. Řešená úloha přitom vyžaduje, aby výsledná triangulace obsahovala hrany tvořící konturu (obálku) korespondujících bodů. Proto jsem pro triangulaci zvolil externí program Triangle (shareware, J. R. Shewchuk, Carnegie Mellon University). Tento program umožňuje nejen zadat hrany, které se musí ve výsledku triangulace objevit, ale řadu dalších podmínek jako např. minimální úhly v trojúhelnících apod.

Zobrazení

Zobrazení výsledku syntézy pomocí vestavěných grafických funkcí Matlabu se jeví jako přímočaré a přirozené řešení. Existuje však několik překážek, které toto řešení komplikují až znemožňují:

- **Textura:** Pro práci s texturou nabízí Matlab poměrně omezené funkce. Mapování textury je pouze lineární. Texturu lze aplikovat pouze na čtvercové plochy. Výsledný, syntetizovaný 3D model scény však vzniká jako drátový model složený z trojúhelníků vzniklých triangulací. Trojúhelníků jsou řádově desetitisíce a veškeré další zpracování tohoto objemu dat před vlastním zobrazením nepřijatelně prodlužuje výpočet.
- **Rychlost zpracování:** Vazba mezi Matlabem a grafickým subsystémem konkrétní výpočetní platformy není zatím dostatečně těsná. I jednoduché modely obsahují tisíce uzlových bodů a desetitisíce trojúhelníků, z nichž každý může teoreticky obsahovat jinou texturu. Doba zobrazení se požadavku na odezvu v reálném čase nepřibližuje. Z chování Matlabu je zřejmé, že i na platformách vybavených akcelerovaným grafickým subsystémem, využívá Matlab pro manipulaci s 3D objekty především softwarové prostředky místo dedikovaného hardwaru.

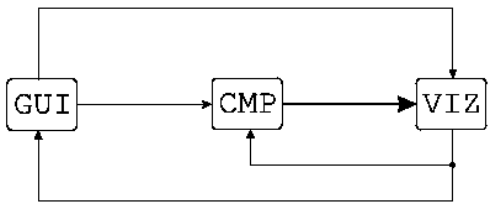
Z těchto důvodů jsem pro zobrazení zvolil externí program Geomview (shareware, The Geometry Center). Geomview je univerzální program pro interaktivní prohlížení a manipulaci s geometrickými objekty. Pokud je počítač vybaven akcelerovaným grafickým subsystém, Geomview jej využije. Program je k dispozici na platformách SGI IRIX, NeXT, dále prakticky na kterémkoli UNIXu s podporou X-Window nebo Open GL.

Popisu toolboxu Geomview Server, který tvoří propojení mezi Matlabem a Geomview, věnuji druhou část článku.

Popis toolboxu

Architektura

Při návrhu architektury toolboxu jsem předpokládal určité typické schéma aplikace (viz obr. 3), grafické rozhranní **GUI** (Matlab), výpočetní modul **CMP** (Matlab a externí programy), vizualizační modul **VIZ** (Geomview).



Obr. 3: Logické schéma výpočetního řetězce

Komunikace mezi jednotlivými moduly je následující:

- **GUI→CMP:** interakce s uživatelem. Data jsou uložena buď ve formě globálně přístupných proměnných nebo přímo ve výpočetním modulu. Příkazy typu *NačtiData(soubor)*, *InterpolujObraz(data)*,
- **CMP→VIZ:** přenos vypočtených 3D dat v předem definovaném formátu, navíc je možno např. v závislosti na typu dat dodat konfigurační parametry pro vizualizaci. Příkazy typu *VykresliObjekt(data)*, *NastavPerspektivníZobrazení()*, *NastavKameru(parametry)*,
- **VIZ→CMP:** požadavky na opakovaný nebo modifikovaný výpočet 3D dat např. při natočení objektu po interakci uživatele přímo s vizualizačním modulem. Objem přepočítávaných dat nemusí být statický, ale lze jej dynamicky měnit technikou zvanou *Level Of Detail*. Tzn. přizpůsobit rozsah přepočítávaných dat vzdálenosti objektu od kamery. Je-li objekt komplikovaný s mnoha detaily, zrychlí tato metoda významným způsobem celý výpočet. Příkazy typu *PřepočtiData(úhly_natočení)*, *DopočtiDetaily(zoom_level)*,

- **VIZ→GUI:** aktualizace GUI podle momentálního stavu vizualizačního modulu. Touto cestou lze přímo odečíst veškeré parametry VIZ a využít je k dalšímu zpracování nebo jejich hodnoty prezentovat uživateli. Příkazy typu *ParametryObjektu(objekt)*, *ParametryKamery(kamera)*,
- **GUI→VIZ:** interakce s uživatelem, přímé řízení vizualizačního modulu. Určeno zejména pro snadnou změnu často užívaných parametrů a nastavení, která z pohledu VIZ nejsou interaktivní, ale uživatel s nimi chce takto zacházet. Příkazy typu *ZměňTexturu(objekt)*, *NastavOkno(okno)* apod.

Potenciálně úzkým místem celého výpočetního řetězce je tok dat **CMP→VIZ**, protože tudy procházejí veškerá zobrazovaná data. Pro efektivní práci výpočetního řetězce je nutné vyřešit tyto základní problémy:

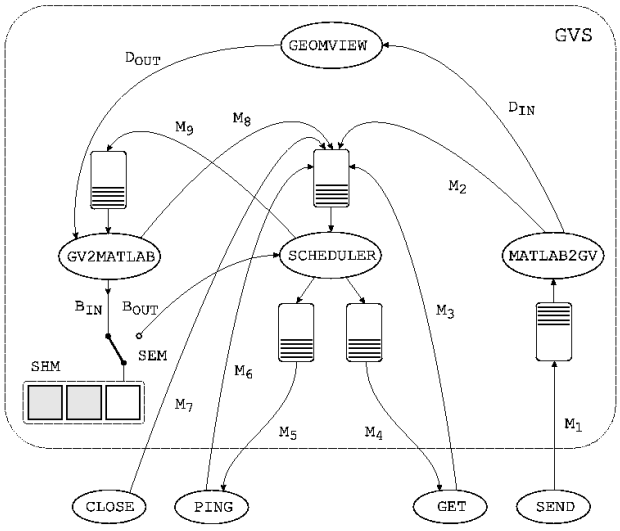
- přenos dat Matlab↔Geomview,
  - synchronní/asynchronní zpracování požadavku na přenos,
  - paralelní přenos Matlab→Geomview a Matlab←Geomview,
- transformace geometrických dat→Geomview OOGL data,
- syntaktická analýza a rozpoznání OOGL a GCL dat z Geomview.

Geomview Server

Jádrem toolboxu je program **Geomview Server** – **GVS** a rozsáhlá knihovna podpůrných programů. GVS je navržen jako samostatný program skládající se z tří nezávislých procesů a knihovny podpůrných programů.

- Jména a funkce procesů:
  1 **SCHEDULER**, rozvrhovací proces,
  2 **GV2MATLAB**, přenos Geomview→Matlab,
  3 **MATLAB2GV**, přenos Matlab→Geomview.
- Jména a funkce podpůrných programů:
  4 **Ping**, ověření činnosti GVS,
  5 **Send**, konverze dat do GCL/OOGL syntaxe, alokace sdílené paměti,
  6 **Get**, konverze bloků sdílené paměti do datových struktur programu Matlab.
  7 **Close**, vyslání žádosti o ukončení činnosti GVS.

Úplné schéma Geomview Serveru, komunikace mezi jednotlivými procesy a vzájemná synchronizace je na obr. 5



Obr. 5: Geomview Server

Zkratky použité ve schématu mají následující význam:

- $\mathbf{M_N}$ *messages*, komunikační kanál, přes který prochází jedna nebo více zpráv,
- $\mathbf{D_{IN/OUT}}$ *data*, označuje přenos dat do nebo z Geomview,
- $\mathbf{B_{IN/OUT}}$ *buffer*, zápis nebo čtení dat ve vyrovnávací paměti,

- **SHM** *SHared Memory*, alokované bloky sdílené paměti,
- **SEM** *semafor* pro výlučný přístup ke sdílené paměti.

Spolupráce Matlabu a Geomview

Pro komunikaci mezi Matlabem a Geomview je využito všech prostředků meziprocesní komunikace, v UNIXu známých jako IPC, tj.:

- **sdílená paměť** (shared memory),
- **fronty zpráv** (message queue),
- **semafor** (semaphore),
- **roura/pojmenovaná roura** (pipe/named pipe).

Přenos dat Matlab→Geomview

Formát dat Geomview pracuje s odlišnými datovými strukturami než je *surface*, *mesh* apod. Nejpoužívanějším formátem je množina trojúhelníků zadaná seznamem vrcholů a hran. Konverzi dat zajišťuje podpůrný program Send, který je napsaný jako *mex*-soubor. V praxi tedy uživatel připraví v prostředí Matlabu např. seznam vrcholů a hran a zavolá program Send, který přenese data do Geomview. V současné době je implementována konverze 10 různých formátů dat. Do Geomview se však nepředávají pouze data k zobrazení, Geomview je třeba programovat vlastním, objektově orientovaným jazykem GCL.

Rychlost přenosu Komunikace s Geomview probíhá nejčastěji přes pojmenovanou rouru. Při předávaní dat je teoreticky třeba 3 přenosů: Matlab→Send→GVS(MATLAB2GV)→Geomview. Při objemu dat řádově stovky kB až jednotky MB jde o vysoké časové nároky. Podstatného zrychlení jsem dosáhl využitím sdílené paměti a paralelizací konverze a přenosu již zkonvertovaných dat. Skutečný postup při přenosu je tedy Matlab→{Send & GVS(MATLAB2GV) & Geomview}

Přenos dat Matlab←Geomview

Rychlost přenosu Přenos dat probíhá, stejně jako v případě přenosu Matlab→Geomview, přes sdílenou paměť. Navíc je k dispozici synchronní a asynchronní mód přenosu. V případě, že přijde asynchronní požadavek a data jsou ve vyrovnávací paměti GV2MATLAB, SCHEDULER je ihned odešle. Pokud data k dispozici nejsou, odešle rozvrhovací proces do fronty zpráv programu Get zprávu, že žádná data nejsou k dispozici. U synchronního požadavku nemůže dojít k situaci, že SCHEDULER neodešle žádná data. Jsou-li ve vyrovnávací paměti data, reaguje rozvrhovací proces stejně jako v synchronním případě. Pokud data nejsou, čeká SCHEDULER s odpovědí tak dlouho, dokud Geomview nějaká data nevygeneruje.

Interpretace dat Geomview poskytuje data na svůj výstup výhradně ve formě textového řetězce. Součástí toolboxu je několik převodních funkcí.

Rozvrhovací proces

Alokace paměti Protože proces zajišťující komunikaci s Geomview nezná předem objem dat, která je třeba uložit do vyrovnávací paměti, ani za jak dlouho přijde požadavek na jejich předání Matlabu, je nutné alokovat vyrovnávací paměť po blocích. Rozvrhovací modul disponuje dynamickou frontou, která obsahuje informaci o všech alokovaných blocích. Alokace sdílené paměti je tedy centrální a děje se tak na základě žádostí zaslaných jednotlivými procesy.

Koordinace procesů SCHEDULER zajišťuje vytvoření front zpráv, iniciuje vznik a zánik ostatních procesů. Koordinace procesů se děje prostřednictvím speciálních zpráv a pomocí semaforu pro zajištění výlučného přístupu ke sdílené paměti.

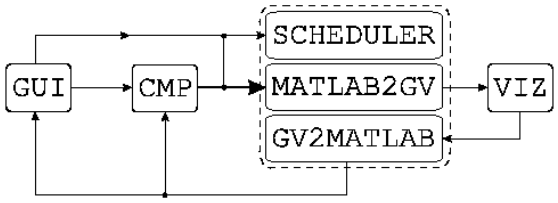
Ověření činnosti GVS

Program Ping slouží k ověření, že GVS pracuje správně. Využívá k tomu frontu zpráv rozvrhovacího procesu, kam zašle žádost o odpověď. Pokud získá odpověď do 1 sekundy, GVS pracuje správně, jinak je stav interpretován jako chybový.

Ukončení činnosti GVS a Geomview

Koordinovaný zánik procesů Ukončení činnosti GVS (a tedy také Geomview) je třeba provést koordinovaně. Jediným úkolem programu Close je zaslat GVS zprávu s žádostí o ukončení činnosti, další iniciativu přebírá proces SCHEDULER v GVS. Nejprve je ukončena činnost Geomview, následuje rozeslání zpráv všem procesům s žádostí o ukončení činnosti. Každý proces uvolní své dynamické struktury, sdílenou paměť a komunikační struktury. Protože IPC datové struktury neuvolňuje operační systém automaticky, je ukončení činnosti každého procesu několikanásobně jištěno a před skutečným koncem pošle rozvrhovacímu procesu potvrzení, že skutečně končí. Jako poslední končí činnost proces SCHEDULER.

Detailní schéma zapojení Geomview Serveru do typického výpočetního řetězce je na obr. 6.



Obr. 6: Detailní schéma výpočetního řetězce

Geomview Manager

Geomview Manager - GVM je knihovna m/mex-funkcí, které usnadňují práci s GVS a přinášejí některé kvalitativně nové funkce. Funkce lze rozdělit do 3 skupin:

1. správa více instancí GVS najednou,
2. komunikace s GVS,
3. syntaktická analýza výstupu Geomview

Správa GVS

Geomview Server je navržen tak, aby jej bylo možné spustit v libovolně mnoha instancích najednou. Každá instance je identifikována unikátní sadou 4 komunikačních klíčů (celých čísel). Pokud jsou komunikační klíče ztraceny, neexistuje žádný způsob jak zjistit jejich hodnoty.

Po spuštění vypisuje GVS klíče na standardní výstup, odkud je lze prostředky Matlabu přečíst. Protože získané hodnoty klíčů by bylo možné v prostředí Matlabu snadno ztratit jediným příkazem *clear*, spravuje GVM speciální databázi uloženou v souboru. Databáze obsahuje pořadové číslo GVS a zmiňované 4 klíče. Každá instance GVS je proto místo 4 libovolných čísel identifikována jen 1 malým celým pořadovým číslem.

Komunikace s GVS

Z uživatelského hlediska je důležitá zejména skutečnost, že funkce zavádějí tzv. „zabezpečenou komunikaci“. Komunikační funkce GVS nabízejí pouze synchronní a asynchronní přenos. Pro efektivní spolupráci s Geomview to však nestačí. Pokud uživatel požádá Geomview o větší objem dat, může sice využít synchronní komunikaci, ale tím získá buď data, která byla vygenerována již dříve nebo jen tolik dat, kolik uběhne času od vyslání žádosti o data ke skutečnému příjmu dat. Aby nebylo nutné neustále testovat, zda byla přenesena všechna data, která jsou výsledkem příslušného příkazu, umožňují funkce GVM vložit do proudu dat synchronizační značku. Pro nalezení značky ve výstupních datech není třeba prohledávat celá výstupní data, ale jen tolik posledních bytů, jaká je délka synchronizační značky. Aby tento mechanismus fungoval, je nutné, aby uživatel mezi vysláním příkazu se synchronizační značkou a příjmem dat, neposílal do Geomview žádné další příkazy, které mají za následek vygenerování nějakého výstupu.

Syntaktická analýza výstupu Geomview

Veškerá data, která Geomview poskytuje jsou ve formě textového řetězce. Závisí na charakteru dat, je-li tato forma vyhovující. GVM obsahuje několik syntaktických analyzátorů, které převádějí textové řetězce na strukturované datové typy Matlabu (např. nastavení kamery, nastavení grafického okna, nastavení vzhledu objektu).

Experimenty

Pro ověření činnosti všech článků navrženého výpočetního řetězce jsem sestavil aplikaci Virtual View Synthesis, která řeší problém diskutovaný v motivačním příkladu. Měření jsem prováděl na dvou různých počítačích za těchto podmínek:

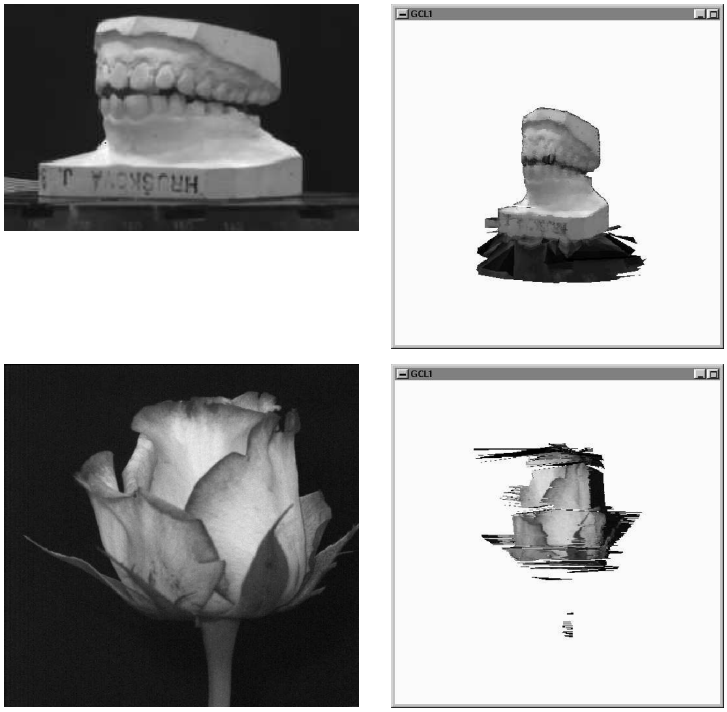
1.PC-Linux

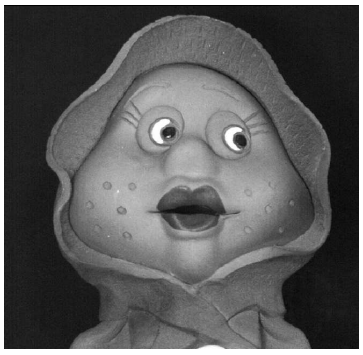
- počítač PC, procesor Pentium 100MHz, 32MB fyzické paměti RAM,
- OS Linux verze 2.0.30
- programová podpora OpenGL knihovnou Mesa verze 2.6,
- Matlab verze 5.1.0.421,
- Geomview verze 1.1.1.

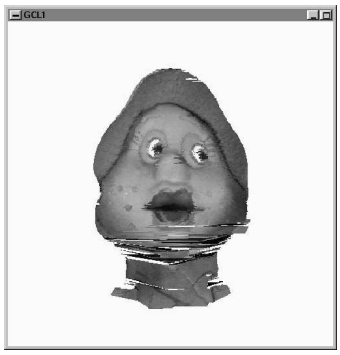
2.SGI-IRIX

- SGI Iris Indigo, procesor MIPS R10000 300MHz, 256MB fyzické paměti RAM,
- OS IRIX verze 6.2,
- akcelerovaný grafický subsystém GL bez hardware podpory mapování textur,
- Matlab verze 5.1.0.421,
- Geomview verze 1.6.1.

Objekty tvoří reprezentativní vzorek, protože se navzájem dostatečně liší velikostí a komplexitou. Výsledky syntézy jsou na následujícím obrázku.







Syntetizované modely jsou jen tak kvalitní, jak kvalitní byla vstupní informace o korespondenci bodů. Zejména na snímcích květiny je vidět omezení algoritmu, který byl využit pro nalezení korespondence. Referenční snímky byly získány pomocí počítačem ovládané točny snímané kamerou. Úzké trojúhelníky bez textury zasahující do objektů představují místa, kde nebyla korespondence nalezena.

Tabulka 1 a 2 uvádí časy naměřené při testování aplikace na platformě PC a SGI. Každá tabulka obsahuje 3 řádky s údaji pro model květiny, zubů a hlavu keramické panenky.

N [1]	M [1]	x [1]	y [1]	t_1 [s]	t_2 [s]	t_3 [s]	t_4 [s]	t_5 [s]	t_6 [s]	t_7 [s]
1580	2354	768	576	0.71	2.91	0.01	0.05	0.05	1.0	9.0
2258	4204	250	160	0.41	4.07	0.01	0.07	0.07	1.5	2.0
5216	9318	700	576	1.33	9.17	0.02	0.08	1.50	3.0	8.0

Tab. 1: PC-Linux

N [1]	M [1]	x [1]	y [1]	t_1 [s]	t_2 [s]	t_3 [s]	t_4 [s]	t_5 [s]	t_6 [s]	t_7 [s]
1580	2354	768	576	0.40	1.13	0.00	0.13	0.13	0.13	3.0
2258	4204	250	160	0.27	1.52	0.00	0.14	0.14	0.14	1.0
5216	9318	700	576	0.74	3.33	0.00	0.14	0.14	0.14	2.5

Tab. 2: SGI-IRIX

Parametr	**Význam**
N	počet uzlů = počet korespondujících bodů
M	počet trojúhelníků vzniklých triangulací
x	šířka referenčního obrázku
y	výška referenčního obrázku
t_1	doba generování omezujících podmínek pro triangulaci
t_2	doba triangulace, včetně vstupu a výstupu dat
t_3	doba syntézy 3D modelu
t_4	doba mezi zahájením přenosu dat (včetně konverze) a návratem řízení zpět do Matlabu
t_5	doba zobrazení drátového modelu
t_6	doba zobrazení „solid“ modelu
t_7	doba zobrazení „solid“ modelu včetně textur

Tab. 3: Význam parametrů v tab. 1, 2

Parametry t_1 až t_4 byly změřeny v Matlabu pomocí příkazů `tic` a `toc`. Dobu nutnou k zobrazení (t_5, t_6, t_7) jsem změřil ručně pomocí stopek, protože Geomview neposkytuje žádnou zpětnou vazbu s informací, kdy byla scéna skutečně zobrazena.

Z naměřených údajů lze vyvodit zajímavé poznatky. Podle očekávání je všeobecná rychlost zpracování na SGI vyšší než na PC. Výjimkou je čistá přenosová doba t_4, která je na SGI přibližně dvojnásobná.

Zajímavé je srovnání doby zobrazení s počtem uzlů, trojúhelníků a velikostí textury. Údaje bezpečně prokázaly, že Geomview skutečně využívá akcelerovaný grafický subsystém, je-li k dispozici. Doba zobrazení objektu bez textury je

na SGI prakticky nezávislá na počtu uzlů a trojúhelníků. Při zobrazení textur rychlost zobrazení strmě poklesla, což odpovídá skutečnosti, že grafický subsystém SGI neakceleruje práci s texturou. Hodnoty naměřené na PC mají stejný charakter, opět největší zátěž tvoří velikost textury.

Ze srovnání t_4 s t_5, t_6 a t_7 vyplývá, že Geomview Server a podpůrné programy (zejména Send) pracují velmi efektivně a režie spojená s přenosem dat a jejich konverzí ani na nepříliš rychlém počítači PC nepřesáhla 0.15 sekundy. Ve srovnání s dobou, kterou potřebuje program Geomview na interpretaci a zobrazení dat, se jedná o naprosto zanedbatelné časové nároky.

Jakmile jsou data přenesena do Geomview, závisí rychlost zpracování pouze na tomto programu. Při vypnuté textuře je na SGI odezva okamžitá bez ohledu na velikost grafického objektu. Po zapnutí textury rychlost poklesne, ale při práci s objekty, s velikostí srovnatelnou se 3 příklady v tabulce, je stále velmi přijatelná a umožňuje plně interaktivní ovládání.

Aplikace Virtual View Synthesis zdaleka nevyužívá všech možností, které navržený výpočetní řetězec a zejména modul VIZ nabízí, ale je dostatečně ilustrativní, aby ukázala jeho schopnosti.

Závěr

Sestavený toolbox není univerzální náhradou grafických knihoven programu Matlab. Jeho použití je vhodné zejména při zobrazování rozsáhlých 3D modelů, kde dosahuje velmi dobrých výsledků z hlediska rychlosti i možností interakce modelu s uživatelem. Unikátní je také skutečnost, že zobrazované scény jsou tvořeny objekty se všemi klasickými atributy, tj. encapsulace (zapouzdření), datová abstrakce, modularita a inheritance (dědičnost).

Z pohledu uživatele Matlabu je zajímavé, že pro komunikaci s externími programy lze využít všech standardních prostředků meziprocesní komunikace a dosáhnout tak podstatně větší rychlosti zpracování, než při obvyklém přenosu dat s využitím dočasných souborů.

Literatura

[1] J. Černík: Interpolace obrazů pro virtuální kameru, diplomová práce, ČVUT FEL, 1998

[2] T. Werner, T. Pajdla, V. Hlaváč: Visualizing 3-D Real-World Scenes Using View Extrapolation, Technical Report Nr. K335-CMP-1997-137, ČVUT FEL 1997

[3] http://www.geom.umn.edu/software/download/geomview_toc.html, Geomview Manual, 1998

[4] http://www.geom.umn.edu/software/download/geomview.html, Geomview Home Page, 1998

[5] http://www.cs.mnu.edu/~quake/triangle.html, Triangle Home Page, 1998

[6] M. Jelen: UNIX V - programování v systému, Grada 1993