Univerzita Palackého v Olomouci
Přírodovědecká fakulta

BAKALÁŘSKÁ PRÁCE

2009 Jan Hatlman

Přírodovědecká fakulta Univerzity Palackého v Olomouci
Katedra experimentální fyziky

Automatizace projekce mřížky v optických 3D měřeních
Grating projection automation in optical 3D methods

Jan Hatlman

Bakalářská práce

Vedoucí bakalářské práce:
RNDr. Tomáš Rössler, Ph.D.
Rok odevzdání: 2009

Vypracoval:
Jan Hatlman
AF, 3. ročník

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně pod vedením pana RNDr. Tomáše Rösslera, Ph.D. a výhradně s použitím uvedené literatury.

V Olomouci dne 25. dubna 2009

Bibliografická identifikace

Autor:	Jan Hatlman
Typ práce:	bakalářská práce
Název práce:	Automatizace projekce mřížky v optických 3D měřeních
Pracoviště	katedra experimentální fyziky
Vedoucí práce:	RNDr. Tomáš Rössler, Ph.D.
Rok obhajoby práce:	2009
Počet stran:	22
Počet příloh:	0
Jazyk:	čeština

Abstrakt: Bakalářská práce se zabývá vytvořením uživatelského grafického rozhraní (GUI - Grafical User Interface), jehož účelem je zvýšit produktivitu práce vědeckého týmu kolem RNDr. Tomáše Rösslera Ph.D. Samotné rozhraní umožňuje uživateli přímo ovládat promítání sinusové mřížky na zkoumaný objekt, což dovoluje snížit nutné vybavení laboratoře o jeden počítač.

V první části práce jsou stručně popsány základní rysy profilometrické metody (Fringed Pattern Profilometry), v které bude grafické rozhraní využíváno. Dále se věnuje prostředí programu MATLAB, v němž je rozhraní napsáno. Závěrečná část se věnuje samotnému grafickému rozhraní, obsahuje jeho kompletní zdrojový kód a jeho grafickou podobu.

Klíčová slova: Fringed Pattern Profilometry, Grafical User Interface, optické 3D měření, MATLAB

Bibliographical identification

Author:	Jan Hatlman
Type of thesis:	bachelor
Title:	Grating projection automation in optical 3D methods
Department:	department of experimental physics
Supervisor:	RNDr. Tomáš Rössler, Ph.D.
The year of presentation:	2009
Number of pages:	22
Number of appendices::	0
Language:	czech

Abstract: The bachelor thesis deals with a creating of Grafical User Interface (GUI). Its aim is to increase the work productivity of research team around RNDr. Tomáše Rösslera Ph.D. The Interface itself allows to its user to control directly projecting of sinus set of small bars on the researched object which allows to bring down the necessary laboratory equipment by one computer.

The first part of this work briefly describes the basic features of fringed pattern profilometry which takes advantage of this interface. The work continues by daeling with a medium of MATLAB programme in which the interface is written. The final part is dedicated to the grafical interface itself. It includes its complete source code and its grafical form.

Keywords: Fringed Pattern Profilometry, Grafical User Interface, optical 3D methods, MATLAB

Obsah

1 Úvod

Ve své bakalářské práci se zaměřím na profilometrickou metodu FPP (z anglického Fringed Pattern Profilometry), jejíž reálné uplatnění v posledních letech značně narůstá. Cílem mé bakalářské práce je vytvoření GUI (z anglického Grafical User Interface), které zvýší produktivitu práce vědeckého týmu seskupeného kolem RNDr. Tomáše Rösslera Ph.D. při promítání sinusové mřížky na zkoumaný objekt. Díky tomuto sníží nutné vybavení laboratoře o jeden počítač, jehož úkolem je doposud pouze vytváření promítaného obrazu promítaného dataprojektorem.

V první kapitole ve stručnosti popíši základní rysy FPP. To převážně z toho důvodu, že materiálů pro hlubší studium této metody je zájemcům k dispozici dostatek. Pro ty, kteří se však touto metodou zabývat hlouběji nehodlají, poskytne toto resumé potřebné základní informace.

V následné kapitole představím vybrané prostředí k vytvoření GUI, kterým je MATLAB. Všeobecně pojednám o možném uplatnění tohoto programu a konkrétně se zaměřím na vestavěné funkce MATLABu, které budou použity v následném GUI.

Třetí kapitola se zaměří na samotné GUI. Představím kompletní zdrojový kód, který cíl mé bakalářské práce generuje, příslušné části řádně okomentuji a vysvětlím hlouběji tam, kde se to projeví nezbytné. Nevynechám ani grafickou prezentaci své práce, aby měl čtenář jasnou představu jak vypadá mnou navržené uživatelské rozhraní.

2 Fringed pattern profilometry

FPP (fringed pattern profilometry) se v posledních letech stává nejpoužívanější nekontaktní 3–D optickou metodou. Při FPP je použita buďto Ronchiho nebo sinusová mřížka, která je promítaná na povrch třírozměrného objektu, který deformuje projekci zvolené mřížky. Deformovaný obraz spolu s jeho originálem promítaným na referenční plochu je snímán pomocí CCD (charge–coupled devices) čipu. Pro rekonstrukci 3–D obrazu zkoumaného objektu bylo vytvořeno mnoho variant této základní metody, například FTP (Fourier transform profilometry), PSP (phase shifting profilometry), SPD (spatial phase detection), PLLP (phase–locked loop profilometry) a další. Existují různé možnosti, jak danou mřížku na objekt promítat. Dnes se nejčastěji používá dataprojektor, protože se jedná o levné a v dnešní době i standardní vybavení, díky čemuž se velmi hodí pro průmyslové aplikace. Na druhé straně je však velmi obtížné obdržet dokonalou sinusovou mřížku za použití dataprojektoru z důvodů geometrického a barevného zkreslení. Pro eliminaci této chyby je možné použít digitální filtr pro vybrání základní frekvence a odříznout tak vyšší harmonické, které ji do měření zanášejí [1].

2.1 Phase-shifting profilometrie

Phase-shifting profilometrie je nekoherentní nekontaktní optická metoda pro měření profilu prostorových objektů. Měření se sestává z postupného promítání čtyř různě posunutých sinusových mřížek na zkoumaný předmět. Sinusová mřížka je dvourozměrná světelná stopa, jejíž intenzitu lze v jednom směru vyjádřit pomocí trigonometrické funkce následovně

$$I(x, y) = I_{max}(1 + sin(kx)),$$

kde I_{max} je maximální hodnota intenzity a k je vlnové číslo sinusové mřížky. Každému bodu obrazové matice lze přiřadit fázy Φ z intervalu $< 0, 2\pi >$. Intenzita obrazu je nyní popsána rovnicí

$$I(x, y) = I_{max}(1 + sin(\Phi(x, y))).$$

Inverzní funkce, která vyjadřuje fázi pomocí intenzity není jednoznačná, což činí analýzu obrazu značně komplikovanou a bez moderní výpočetní techniky zcela neproveditelnou. Nejednoznačnost je možno řešit dvěma způsoby. Při každém z nich je třeba zvýšit počet dat tak, aby byla inverzní funkce již jednoznačná. První způsob tyto data získává z okolí měřeného bodu, což je u předmětů bez skokových změn v topologii zcela postačující. Je-li však povrch předmětu poškozen například poškrábáním, profil předmětu bude vyhodnocen nepřesně, protože dojde k zprůměrování s body v okolí. K zamezení takto vzniklé chyby měření se používá druhá metoda, pří které se na předmět postupně promítá více sinusových mřížek, které jsou vzájemně posunuty o přesně definovanou fázi. Takto se získá několik údajů o velikosti intenzity pouze pro konkrétní bod a výsledek

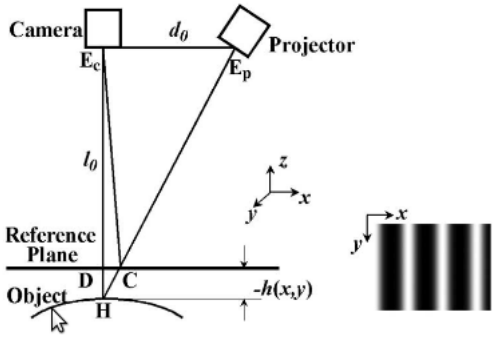
není náchylný na chyby způsobené skokovou změnou tvaru předmětu. Právě automatické promítání sinusových mřížek v různou fází je mým cílem. Lze dokázat, že tři sinusové mřížky, vzájemně posunuté o 120° stačí pro určení profilu předmětu. Standardně se však používají čtyři mřížky s posunutím 90°. Intenzita jednotlivých obrazů je vyjádřena následovně:

$$I_r(x,y) = I_0(x,y) + I_C(x,y)sin(\Phi_r(x,y))$$

kde $r = 1, \ldots, 4$, $I_0(x,y)$ je intenzita pozadí a $I_C(x,y)$ je kontrast mřížky. Fázi lze vyjádřit jako funkci všech čtyř intenzit takto:

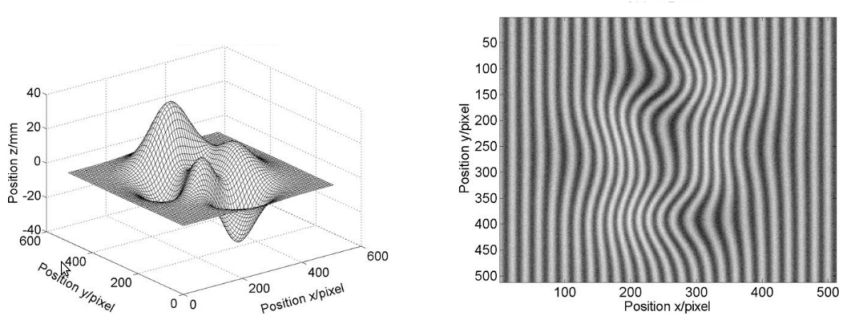
$$\Phi_1(x,y) = arctan\frac{I_1(x,y) - I_3(x,y)}{I_2(x,y) - I_4(x,y)}.$$

Základní experimentální sestava je zobrazena na obrázku (1).



Obrázek 1: Převzato z [1]. Schématické zobrazení jednoduchého FPP systému.

Na obrázku (2) jsou výstupy metody.



Obrázek 2: Převzato z [2]. Nalevo je zkoumaný objekt a napravo je záznam objektu, jenž byl pořízen CCD čipem.

3 MATLAB

MATLAB[1] je akronym z anglického MATrix LABoratory a na tomto místě je vhodné krátce popsat toto zvolené prostředí, ve kterém bude GUI vytvořeno.

3.1 Základní rysy MATLABu

MATLAB je vysokoúrovňový jazyk pro matematické a technické výpočty, vizualizaci a programování v uživatelsky přívětivém prostředí, které je doplněné o rozsáhlou dokumentaci v anglickém jazyce. Běžně se používá k řešení následujících problémů:

- Matematika a výpočetní technika
- Vývoj algoritmů
- Manipulace s daty
- Modelování a simulace
- Analýza dat, jejich interpretace a vizualizace
- Vědecká a inženýrská grafika
- Aplikační prostředí, zahrnující GUI

MATLAB je interaktivní systém, jehož klíčovým datovým prvkem je pole (anglicky array), které implicitně nevyžaduje zadání dimenze. To umožňuje řešení mnoha technických problémů, především v jejich maticové či vektorové formulaci. V porovnání s psaním kódu po jednotlivých skalárech, jak vyžaduje například jazyk C nebo Fortran, se jedná o úsporu času programátora.

MATLAB je vyvíjen firmou MathWorks již mnoho let a má značnou uživatelskou komunitu. Na mnohých univerzitách technického směru je to první, ne-li jediné prostředí, se kterým jsou studenti blíže seznámí. Jeho využití v průmyslu je také značné, neboť vývoj aplikací nevyžaduje takové odborné znalosti a zkušenosti jako například již výše zmíněný jazyk C. Jistou nevýhodou, v porovnání s nízkoúrovňovými jazyky je jistá těžkopádnost většiny programů, která ve projeví ve větší náročnosti na strojový čas. To se však u převážné části standardních problémů neprojevuje tak negativně, neboť při dnešním technickém pokroku tvoří cenu aplikace především práce programátora, která je při volbě MATLABu v drtivé většině případů mnohonásobně menší.

[1] Všechny informace související s MATLABem jsem čerpal z nápovědy poskytované spolu s programem.

Pro jednotlivé specifické problémy jsou dostupné tzv. toolboxy. Toolbox je obsáhlý soubor funkcí (m–souborů), které doplňují prostředí MATLABu o řešení individuální třídy problémů. Některé jsou v základní nabídce a jiné za úplatu rozšiřují tento základní set. Některé namátkou vybrané toolboxy:

- Optimalizační návrhy
- Statistické výpočty
- Výpočty diferenciálních rovnic
- Tvorba neuronových sítí
- Práce s grafikou
- Uživatelem vytvořené

3.2 Systém MATLAB

Systém MATLABu je založen na následujících částech: desktopové a vývojové prostředí, knihovna matematických funkcí, grafika a externí rozhraní.

Desktopové a vývojové prostředí

Tato část zahrnuje nástroje a podporu, která napomáhá uživateli s pochopením a manipulací vestavěných funkcí. Mnohé z těchto nástrojů jsou grafická rozhraní, což zvyšuje uživatelský komfort při jejich využití.

Knihovna matematických funkcí

Jedná se o nepřeberné množství vestavěných funkcí, které dovolují výpočty zcela triviální jako je sčítání, výpočet stopy matice či hodnoty sinu daného úhlu až po složité výpočetní algoritmy pro řešení obyčejných diferenciálních rovnic Rungeovou–Kuttovou metodou čtvrtého řádu, výpočet inverzní n–dimenzionální matice nebo FFT.

Grafika

MATLAB má rozsáhlé možnosti pro vizualizaci a tisk vektorů i matic. Disponuje propracovaným souborem funkcí pro vizualizaci dvou a tří dimenzionálních dat, image processingu, animace a grafické prezentace. Také poskytuje nízko úrovňové alternativy, které ocení především zkušenější uživatelé, jenž musí vyhovět přesným požadavkům a univerzální řešení nesplňují vlastnosti na ně kladené.

Externí rozhraní

Externí rozhraní dovolují psát kódy v jazyku C nebo Fortranu, které běží pod MATLABem. To zahrnuje utility pro dynamické linkování (volání funkcí MATLABu), volání výpočetního jádra MATLABu a pro čtení či zápis m–souborů.

3.3 Funkce imread

V této části nasleduje popis vestavěné funkce `imread.m` , která je základní součástí prezentovaného GUI.

Výčet syntaxe

- `A = imread (jméno souboru, fmt)`
- `[X, map] = imread (...)`
- `[...] = imread (jméno souboru)`
- `[...] = imread (URL, ...)`
- `[...] = imread (..., idx)` – CUR nebo ICO
- `[A, map, alpha] = imread (...)` – CUR nebo ICO
- `[...] = imread (..., idx)` – GIF
- `[...] = imread (..., frames,idx)` – GIF
- `[...] = imread (..., ref)` – HDF4
- `[...] = imread (..., 'BackgroundColor' ,BG)` – PNG
- `[A, map, alpha] = imread (...)` – PNG
- `[...] = imread (..., idx)` – TIFF
- `[...] = imread (...,  'PixelRegion', {ROWS, COLS})` – TIFF

Popis syntaxe

`A = imread(jméno souboru, fmt)` načte barevný obrázek či obrázek v barvě šedi, který je specifikovaný svým jménem. Není–li soubor v aktivním nebo kmenovém adresáři MATLABu je třeba zadat cestu k tomuto souboru. `fmt` specifikuje formát, ve kterém je soubor uložen.

Návratová hodnota A je pole obsahující obrazová data. Pokud soubor obsahuje obrázek v odstínech šedi, A je $M*N$ dimenzionální matice. Je–li obrazek barevný, A je $M*N*3$ pole. TIFF soubor používá pole o velikosti $M*N*4$. To z důvodu, že TIFF je založen na barevném modelu CMYK jenž míchá barvy subtraktvně. CMYK má tyto základní barvy:

- azurovou (**C**yan)
- purpurovou (**M**agenta)
- žlutou (**Y**ellow)
- černou (blac**K**) označovanou také jako klíčovou (**K**ey).

Třída odpovědi A záleží na bitech grafického souboru. Například imread vrací 24–bitové pole ve formátu uint8 (rozsah je tedy od 0 do 255 neboť $2^8 = 256$), protože vzorek pro každou základní barvu má 8 bitu.

Výčet formatů

MATLAB implicitně podporuje nasledující formáty:

- BMP – Windows Bitmap
- CUR – Cursor File
- GIF – Graphics Interchange Format
- HDF4 – Hierarchical Data Format
- ICO – Icon File
- JPEG – Joint Photographic Experts Group
- PBM – Portable Bitmap
- PCX – Windows Paintbrush
- PGM – Portable Graymap
- PNG – Portable Network Graphics
- PPM – Portable Pixmap
- RAS – Sun Raster
- TIFF – Tagged Image File Format
- XWD – X Window Dump

3.4 Funkce image

V této části nasleduje popis vestavěné funkce `image.m` , která je základní součástí prezentovaného GUI.

Výčet syntaxe

- `image (C)`
- `image (x, y, C)`
- `image (x, y, C, 'Vlastnost', HodnotaVlastnosti, ...)`
- `image ('Vlastnost', HodnotaVlastnosti, ...)`
- `handle = image (...)`

Popis syntaxe

Tato funkce vytváří obraz interpretací každé hodnoty jednotlivého elementu matice a to v definované barevné paletě nebo přímo v RGB, což záleží na specifikaci matice. Funkce pracuje ve dvou krocích. Nejprve se zavolá vysokoúrovňová funkce `newplot`, která vykreslí obraz a nastaví některé vlastnosti (rozsah x-ové a y-ové osy, odsazení obrazu od levého a spodního okraje monitoru, nastavení úhlu otočení je–li nějaký) a poté vyvolá nízkoúrovňovou funkci, která pak již jen přidá obraz do kostry vytvořené v předchozím kroku.

Je dovoleno nastavit různé vlastnosti, jenž dále upravují tuto funkci.
`image(C)` zobrazuje matici C jako obrázek. Každý element matice C specifikuje barvu v obrázku, který se zobrazuje jako pravoúhlá síť.
`image(x,y,C)`, kde x a y jsou dvouprvkové vektory, které specifikují rozsah osy x a y, přičemž vytváří stejný obraz jako `image(C)`. Tohoto se dá využít, když je třeba zobrazit pouze fyzikálně relevantní dimenze v odpovídajícím obrazu.
`image(x,y,C, 'Vlastnost', HodnotaVlastnosti, ...)` při této volbě se volá funkce `newplot`.
`image('Vlastnost', HodnotaVlastnosti, ...)` je nízkoúrovňový ekvivalent, který nepotřebuje pro svou činnost `newplot` a proto je její použití méně náročné na strojový čas.
`handle = image(...)` vrací ukazatel na obraz, který zároveň vytvoří. Je možné funkci `handle` použít u všech funkcí MATLABu, které pracující s grafikou.

4 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní, zkráceně GUI z anglického Graphical User Interface, je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků. Na monitoru počítače jsou zobrazena okna, ve kterých programy zobrazují své výstupy. Uživatel používá klávesnici, myš a grafické vstupní prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře a podobně [3].

Kromě grafických existují i jiná uživatelská rozhraní:

- textové uživatelské rozhraní (s menu, tlačítky a myší)
- příkazový řádek (příkazy se zadávají jejich zapsáním pomocí klávesnice)
- hlasová rozhraní a další

MATLAB umožňuje vytvořit GUI dvěma různými způsoby. Prvním z nich je pomocí průvodce (anglicky GUIDE), kde se vše vytváří pomocí myši. Toto je na první pohled jednodušší, ale pokud je potřeba cokoliv změnit, je nutné nakreslit GUI znovu, což je velmi nepraktické. Druhou možností je naprogramování GUI a uložení v klasické podobě m–souboru, který lze kdykoliv upravit podle aktuálních potřeb. Z důvodu větší pružnosti byla vybrána právě tato metoda tvorby grafického rozhraní. V následující části se bude popsán zdrojový kód.

4.1 Zdrojový kód

Celé GUI splňující výše zmíněné požadavky vypadá následovně.

```
mOutputArgs = {};

hMainFigure = figure(...
                    'NumberTitle','off',...
                    'Name', 'Verze alfa 0.8.2',...
                    'MenuBar','none', ...
                    'Toolbar','none', ...
                    'HandleVisibility','callback',...
                    'Color', get(0,...
                       'defaultuicontrolbackgroundcolor'));
```

```
hPlotAxes = axes(...
                   'Parent', hMainFigure, ...
                   'Units', 'normalized', ...
                   'HandleVisibility','callback', ...
                   'Position',[0.5 0.15 0.45 0.8]);

hUpdateButton1 = uicontrol(...
                   'Parent', hMainFigure, ...
                   'Units','normalized',...
                   'HandleVisibility','callback', ...
                   'Position',[0.1 0.85 0.3 0.1],...
                   'String','Vzor 1',...
                   'Callback', @vzor1);

hUpdateButton2 = uicontrol(...
                   'Parent', hMainFigure, ...
                   'Units','normalized',...
                   'HandleVisibility','callback', ...
                   'Position',[0.1 0.70 0.3 0.1],...
                   'String','Vzor 2',...
                   'Callback', @vzor2);

hUpdateButton3 = uicontrol(...
                   'Parent', hMainFigure, ...
                   'Units','normalized',...
                   'HandleVisibility','callback', ...
                   'Position',[0.1 0.55 0.3 0.1],...
                   'String','Vzor 3',...
                   'Callback', @vzor3);

hUpdateButton4 = uicontrol(...
                   'Parent', hMainFigure, ...
                   'Units','normalized',...
                   'HandleVisibility','callback', ...
                   'Position',[0.1 0.40 0.3 0.1],...
                   'String','Vzor 4',...
                   'Callback', @vzor4);
```

```
hUpdateButton5 = uicontrol(...
                  'Parent', hMainFigure, ...
                  'Units','normalized',...
                  'HandleVisibility','callback', ...
                  'Position',[0.1 0.25 0.3 0.1],...
                  'String','Zavrit',...
                  'Callback', @zavri);

function pushbutton1_Callback(hObject,eventdata)
                  a= imread ('r1.bmp');
                  image(a)
                  colormap(gray)
                  v = get(0,'MonitorPosition');
                  w=size(a);
                  figure('MenuBar','none','Name','Vzor 1',...
                     'Position',[v(2,1) 1 w(2) w(1)]);
                  set(gca,'position',[0 0 1 1])
                  colormap(gray)
                  image(a)

function pushbutton1_Callback(hObject,eventdata)
                  a= imread ('r2.bmp');
                  image(a)
                  colormap(gray)
                  v = get(0,'MonitorPosition');
                  w=size(a);
                  figure('MenuBar','none','Name','Vzor 1',...
                     'Position',[v(2,1) 1 w(2) w(1)]);
                  set(gca,'position',[0 0 1 1])
                  colormap(gray)
                  image(a)
```

```
function pushbutton1_Callback(hObject,eventdata)
                    a= imread ('r3.bmp');
                    image(a)
                    colormap(gray)
                    v = get(0,'MonitorPosition');
                    w=size(a);
                    figure('MenuBar','none','Name','Vzor 1',...
                       'Position',[v(2,1) 1 w(2) w(1)]);
                    set(gca,'position',[0 0 1 1])
                    colormap(gray)
                    image(a)

function pushbutton1_Callback(hObject,eventdata)
                    a= imread ('r4.bmp');
                    image(a)
                    colormap(gray)
                    v = get(0,'MonitorPosition');
                    w=size(a);
                    figure('MenuBar','none','Name','Vzor 1',...
                       'Position',[v(2,1) 1 w(2) w(1)]);
                    set(gca,'position',[0 0 1 1])
                    colormap(gray)
                    image(a)

function pushbutton1_Callback(hObject,eventdata)
                    display 'Copyright Jan Hatman Coop.'
                    close(gcbf)
```

4.2 Popis jednotlivých částí programu

V MATLABu je GUI figura. Předtím, než jsou přidány jeho komponenty, vytvoří se figura, do které se jednotlivé části přidají. Mnou vytvořena figura se jmenuje hMainFigure. V kulaté závorce za příkazem figure jsou specifikovány její parametry. Na konci každého řádku je uveden komentář, který vysvětlí jednotlivé nastavení. Tato konvence bude držena v popisu celého GUI. Pro větší přehlednost začíná nastavení jednotlivých parametru na novem řádku. Aby byl kód čitelný pro překladač MATLABu, je nutné přidat na řádkovém zlomu tři tečky, čímž předkladač pochopí, že příkaz pokračuje na dalším řádku a neukončí svou práci předčasně.

```
mOutputArgs = {}; % proměnná pro uložení návratových hodnot

hMainFigure = figure(... % vytváří hlavní okno - figuru
                     'NumberTitle','off',...
                     % neuvádí číslo okna v záhlaví
                     'Name', 'Verze alfa 0.8.2',...
                     % uvádí zvolené jméno
                     'MenuBar','none', ...
                     % neuvádí hlavní nabídku
                     'Toolbar','none', ...
                     % neuvádí nabídkový řádek
                     'HandleVisibility','callback',...
                     'Color', get(0,...
                              'defaultuicontrolbackgroundcolor'));
                     % standardní nastavení barev
```

Nyní je možno přistoupit k přidání jednotlivých částí do již vytvořené figury. Pro přidání komponentů je nutné jednoznačné určit jejich umístění. Lze postupovat dvěma způsoby. Zadat rozměry v podobě jednotlivých pixelu, přičemž se pak vždy jedná o přirozené číslo. Nebo zadat velikost jako poměr vůči délce a šířce vygenerovaného obrázku. Poté se jedna o kladné desetinné číslo o velikosti maximálně 1, které je sice MATLABem přípustné, ale nedoporučuje se používat čísla větší než 0.90 z vizuálních důvodu. V obou případech se jedna o zadáni čtveřice čísel v hranatých závorkách. První označuje vzdálenost od levého okraje figury, druhé od spodního okraje, třetí je šířka vkládaného prvku a poslední je jeho výška. Velikost byla zadávána poměrem, protože se při změně velikosti figury odpovídajícím způsobem automaticky změní velikost komponentu, což při zadáni natvrdo není možné.

GUI obsahuje dvě skupiny různých komponentů, kterými jsou tlačítka a vizualizační prostor.

Příkaz axes přidává pole, kde se vykresluji jednotlivé grafy či obrázky.

```
hPlotAxes = axes(...
                    'Parent', hMainFigure, ...
                     % uvádí kam se má pole vytvořit
                    'Units', 'normalized', ...
                     % nastavuje měření poměrem
                    'HandleVisibility','callback', ...
                    'Position',[0.5 0.15 0.45 0.8]);
                     % zadání vzdálenosti
```

Příkaz uicontrol vytváři tlačítko, které po kliknutí na něj provede předepsaný úkol.

```
hUpdateButton1 = uicontrol(... % vytvoří prvního tlačítko
                    'Parent', hMainFigure, ...
                    'Units','normalized',...
                    'HandleVisibility','callback', ...
                    'Position',[0.1 0.85 0.3 0.1],...
                    'String','Vzor 1',...
                     % vypíše název na lačítko
                    'Callback', @vzor1);
                     % nejdůležitější část, volá funkci
```

Následující čtyři tlačítka jsou vytvořena stejným způsobem.

V tomto okamžiku je vytvořená grafická podoba GUI, která není funkční. Je třeba ještě předepsat co se má stát po zmáčknutí příslušného tlačítka. To je provedeno v samostatných funkcích, které mohou být v samostatném m–souboru, který se pak musí jmenovat podle příslušné funkce (tedy poslední tlačítko s názvem Zavři volá @zavri, tudíž soubor se musí jmenovat zavri.m) a nebo může být vše uloženo v jednom souboru. Na funkci GUI nemá žádná ze zvolených variant vliv, ale většinou se volí vytváření samostatných souborů, protože se lepe editují a nebo se poté celé použijí v jiném projektu.

Nyní je vhodné popsat myšlenku, na které je zobrazení požadované sinusové mřížky na dataprojektoru vystaveno. MATLAB disponuje vestavěnu funkcí pro zjišťování počtu připojených externích grafických výstupu a jejich rozlišení udávané v pixelech. Touto funkci je:

```
get(0,'MonitorPosition')
```

Počet připojených zařízení odpovídá počtu řádků, které funkce vypíše, přičemž na jednom řádku je vždy jedno čtyřčíslí. Jsou–li připojeny k počítači dva monitory, přičemž primární má rozlišení 2048*1152 pixelů a sekundární 1024*768 pixelů MATLAB vypíše následující:

```
get(0,'MonitorPosition')
ans =
     1    1    2048 1152 % Primární monitor
     2049 1153 3072 1920 % Sekundární monitor
```

První číslo na prvním řádku značí pixel od kterého se počítají pixely monitoru v horizontálním směru a druhé číslo ve vertikálním směru. Třetí číslo je pořadí horizontálního pixelu, na kterém končí první monitor a poslední, čtvrté číslo je

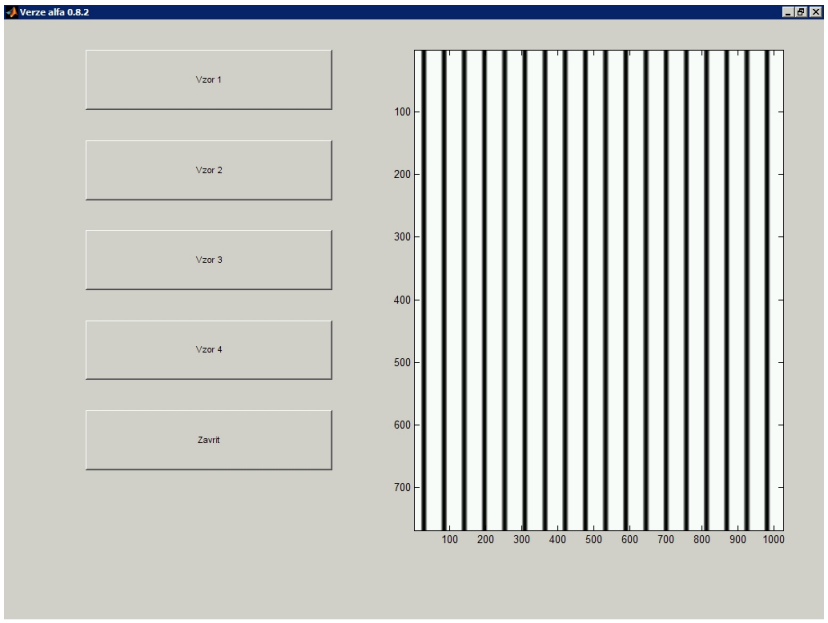
pixel, na kterém končí monitor v horizontálním směru. Druhý řádek odpovídá druhému monitoru a čtyřčíslí má stejnou logiku. Jeho první číslo (2049) je pixel v horizontálním směru, na kterém začíná druhý monitor, což je 2048 (z prvního monitoru) + 1. Druhé číslo na stejném řádku (1153) je pixel ve vertikálním směru, na kterém začíná druhý monitor, což je 1152 (z prvního monitoru) + 1. Třetí resp. čtvrtá číslice označuje pixel druhého monitoru ve vertikálním resp. horizontálním směru. Tedy v našem případě je odpověď (ans) matice 2*4.

V okamžiku, kdy je známa tato matice je třeba vytvořit novou figuru, která začíná na prvním horizontálním a prvním vertikálním pixelu dataprojektoru a jako konec se zvolily pixely, které se rovnají součtům těchto pixelů s rozměrem nahraného obrazu.

```
function pushbutton1_Callback(hObject,eventdata) % povinná část
                    a= imread ('r1.bmp');
                    % nahraje data ze souboru r1.bmp
                    image(a)
                    % zobrazí načtené data v GUI...
                    colormap(gray)
                    %... v černobílé podobě
                    v = get(0,'MonitorPosition');
                    % již vysvětleno
                    w=size(a);
                    % zjišťuje velikost nahraného obrazu
                    figure('MenuBar','none','Name','Vzor 1',...
                       'Position',[v(2,1) 1 w(2) w(1)]);
                    % vytváří nový obraz na dataprojektoru
                    set(gca,'position',[0 0 1 1])
                    colormap(gray)
                    image(a)
```

Následující tři tlačítka jsou stejná. Pouze nahrávají a zobrazí jiné obrazy. Jediným jiným tlačítkem je poslední, které zavře GUI.

```
function pushbutton1_Callback(hObject,eventdata)
                    display 'Copyright Jan Hatman Coop.'
                    % tuto firmu nenajdete v obchodním rejstříku
                    close(gcbf)
```



Obrázek 3: Takto vypadá výše popsané GUI. V levé části je možno vybrat jeden ze vzorů či GUI zavřít pouhým kliknutím na příslušné tlačítko. V pravé části je zobrazen vzor, který je současně promítán dataprojektorem.

5 Závěr

Bakalářská práce splnila cíle, které byly vytčeny v úvodu práce. Především se podařilo naprogramovat grafické rozhraní v programu MATLAB, které zvyšuje pohodlí při promítání čtyř sinusových mřížek s vzájemně posunutou fází o 90°. Světelná stopa se promítá dataprojektorem, který je ovládán počítačem, který zároveň provádí analýzu dat, čímž se snížil nárok na výpočetní vybavení optické laboratoře o jeden počítač, který doposud ovládal pouze dataprojektor. Mimo tohoto hlavního úkolu jsem stručně popsal základní princip optické metody zvané phase-shifting profilometrie, k jejíž dalšímu studiu na naší katedře se mnou vytvořené GUI používá.

S vědeckým týmem RNDr. Tomáše Rösslera Ph.D. bych rád pokračoval při studiu této metody v magisterském studiu, kdybych se chtěl zaměřit na ovládání CCD čipu a samotnou analýzu dat, čímž by se stalo měření zcela automatické.

Reference

[1] Y. Hu, J. Xi a kolektiv: *Study on Generalized Analysis Model for Fringe Pattern Profilometry*, IEEE transactions on instrumentation and measurement, vol. 57, no. 1, 2008

[2] Q. Zhang, W. Chen a kolektiv: *Method of choosing the adaptive level of discrete wavelet decomposition to eliminate zero componen*, Optics Communications, vol. 38, no. 4, 2008

[3] P. Marchand a O. T. Holland: *Graphics and GUIs with MATLAB*, Chapman, 2003

[4] G. S. Settles: *Schlieren and Shadowgraph Techniques*, Springer, 2001

[5] T. Yan, Z. Qiang a kolektiv: *BP neural network applied to 3D object measurement based on fringe pattern projection*, Optics, vol. 120, no. 10, 2007