FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Software pro zpracování dat získaných z laserového speckle interferometru |
| **Student:** | Jan Glaser |
| **Vedoucí:** | Ing. Zdeněk Otčenášek, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce zimního semestru 2019/20 |

## Pokyny pro vypracování

Cílem práce je návrh a implementace SW pro další zpracování dat získaných z laserového speckle interferometru. Jedná se o práci s daty vzniklými při testování hudebních nástrojů v laboratoři na HAMU (Hudební a taneční akademie múzických umění).

Funkční požadavky jsou následující:
1. Identifikace polohy bodu s nejmenším průhybem sledovaného objektu v sekvenci snímků z jedné periody jedné frekvence buzení objektu a umístění této polohy mezi okraje sledované oblasti pro další zpracování obrazců dané sekvence.
2. Interaktivní úpravy okrajů oblasti, ve které se nachází sledovaný objekt.
3. Porovnání jasu dvojice speckle obrazců a vyloučení nevhodných snímků.

Úkoly práce:
1. Nastudujte a popište data, která v měření vznikají a možnosti jejich úprav.
2. Analyzujte existující aplikaci, diskutujte možnost implementovat požadavky jako modul nebo jako samostatný SW. Volbu zdůvodněte.
3. Zvolte vhodnou implementační platformu a implementujte.
4. Řešení řádně zdokumentujte a otestuje.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 7. března 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Software pro zpracování dat získaných z laserového speckle interferometru

***Jan Glaser***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Zdeněk Otčenášek, Ph.D.

15. května 2018

# Poděkování

V první řadě děkuji vedoucímu této bakalářské práce Ing. Zdeňkovi Otčenáškovi, Ph.D. za konzultace a materiály, které mi během psaní práce poskytl. Dále děkuji Hudební a taneční fakultě Akademie múzických umění v Praze za přístup do laboratoře a možnost ladění aplikace. Děkuji též i mému otci, Ing. Vladimíru Glaserovi, CSc. za jeho podporu během celého studia.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2018 .....

### Odkaz na tuto práci

Glaser, Jan. *Software pro zpracování dat získaných z laserového speckle interferometru*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

# Abstrakt

Tato bakalářská práce se zabývá studií problematiky, návrhem a implementací aplikace pro zpracovávání dat získaných z testů kvality hudebních nástrojů v laboratoři na fakultě HAMU (Hudební a taneční fakulta Akademie múzických umění) v Praze. V práci se zabývám analýzou vstupních dat, návrhem a implementací software pro zpracovávání. Výsledná aplikace je napsána v jazyce C# a je schopna zpracovávat všechna validní vstupní data. Díky této aplikaci je možné urychlit a usnadnit zpracovávání výsledků.

**Klíčová slova** aplikace, hudba, hudební nástroje, akustika, analýza zvuku

# Abstract

This bachelor thesis deals with the study of problems, design and implementation of the application for processing of data obtained from tests of musical instruments done in the laboratory at the the Music and Dance Faculty of the Academy of Performing Arts in Prague. In the thesis I deal with analysis of input data, design and implementation of software for processing. The resulting application is written in C# and is capable of processing all valid input data. Thanks to this application, it is possible to speed up and facilitate the processing of test results at the faculty.

**Keywords** application, music, musical instrument, acoustic, sound analysis

# Obsah

# Seznam obrázků

# Úvod

Hudba a hudební nástroje provázejí lidstvo od nepaměti. Zkoumání historických a vývoj nových hudebních nástrojů novými metodami je i dnes velmi zajímavým a důležitým studijním oborem. U nástrojů s rezonanční deskou probíhá řada výzkumů, které dávají do souvislosti její kmitání a konstrukční provedení s posluchači vnímanou kvalitou zvuku. Pro potřeby analýzy je třeba opakovaně hudební nástroj rozeznít. Toho lze dosáhnout mechanicky rozkmitáním (buzením) např. kobylky u smyčcových nástrojů. Během buzení je objekt snímán laserovým speckle interferometrem a výsledná data se zpracovávají. Výsledkem je animace průběhu kmitání testovaného nástroje, která pomáhá při zásazích do konstrukce hudebního nástroje, umožňuje odhalit vady v ozvučnici atd.

Práce se zaměřuje na návrh a impementaci software, který dále zpracovává naměřená data a usnadňuje zpracovávání výsledků testů. Dlouhodobě se zajímám o výpočetní techniku a ve volném čase se zabývám hudbou, konkrétně hraji na klavír a housle. Tato práce spojuje obě oblasti mého zájmu a je prospěšná pracovníkům HAMU (Hudební a taneční fakulta Akademie múzických umění) v Praze.

V teoretické části práce se věnuji rozboru problematiky získání speckle obrazců prohýbajícího se objektu při laserové interferometrii a významu polohy nulové výchylky pro sledování průhybu. V praktické části se věnuji analýze uživatelských požadavků, vstupních dat, návrhu a implementaci vlastní aplikace. Cílem je umožnit uživateli upravovat mapu okrajů oblasti, ve které se nachází sledovaný objekt, porovnání jasu dvojice speckle obrazců, vyloučení jasově nepoužitelných snímků a identifikace polohy bodu s nejmenším průhybem sledovaného objektu v sekvenci snímků z jedné periody jedné frekvence buzení objektu a umístění této polohy mezi okraje sledované oblasti pro dalsí zpracování obrazců dané sekvence. Práce navazuje na existující aplikaci ISTRA, která provádí měření pomocí laserového interferometru a ukládá data. Tato data jsou pak dále zpracovávána mojí aplikací.

Kapitola 1

# Cíl práce

Cílem práce byl návrh a implementace software umožňující zpracování dat vzniklých při testování hudebních nástrojů v laboratoři HAMU.

Účelem teoretické části bylo studium a popis procesu měření, zjištění struktury měřených dat a diskutování možností jejich zpracování.

Cílem praktické části bylo navrhnout a implementovat aplikaci, která identifikuje polohu bodu s nejmenším průhybem ve sledovaném objektu v sekvenci snímků z jedné periody jedné frekvence buzení objektu a umístí polohu bodu mezi okraje sledované oblasti pro další zpracování obrazců dané sekvence. Aplikace dále umožní uživateli interaktivní úpravy okrajů oblasti, ve které se nachází sledovaný objekt a porovnáním jasu dvojice speckle obrazců vyhodnotí nevhodné snímky.

Kapitola 2

# Literární rešerše

## 2.1 Problematika

Cílem mé bakalářské práce je navrhnout a naimplementovat aplikaci, která usnadní proces testování hudebního nástroje.

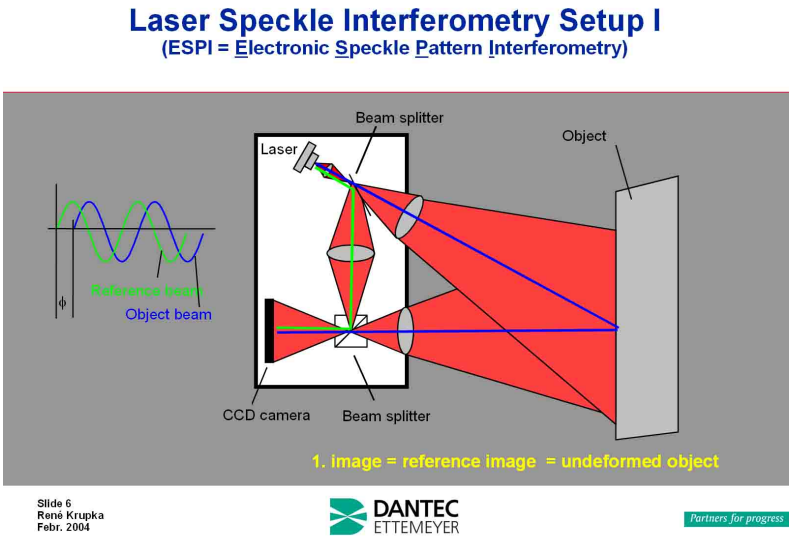
Hudební nástroj (např. housle, kytara či banjo) se upevní do k tomu určených úchytů v akustické laboratoři. Poté následuje série testů pomocí laserového speckle interferometru. Existující aplikace „ISTRA“ řídí průběh měření a zpracovává data. Výledkem měření je, mimo jiné, množina fázových map (obrázků), které v čase znázorňují rozdíl pohybu jednotlivých bodů měřeného objektu mezi dvojicí snímků získaných videokamerou. Snímky pořízené v extrémně krátkých časových intervalech zachycují stav průhybu objektu osvětleného záblesky laserového světla.

Ze série fázových map z takovýchto dvojic snímků posouvaných v čase vůči vhodně zvolené synchronizační události (stroboskopické snímkování při konstantním buzení) lze vytvořit animace, které umožní výrobcům hudebních nástrojů netradiční pohled na kmitající části nástroje a pomohou při úpravách konstrukce nástroje.

### 2.1.1 Podrobný popis měření

Cílem měření je zjistit, zda, a v jaké míře došlo k pohybu (prohnutí) objektu v bodech na jeho ploše v závislosti na čase. K měření se používá laserový speckle interferometr [1, s. 1]. Měřený objekt se umístí před interferometr do předem určené vzdálenosti. Interferometr se nabije světlem a na základě triger impulsu rovnoměrně osvítí celou plochu testovaného objektu dvěma záblesky laserového světla s předvolitelnou prodlevou $t$ mezi záblesky [2, s. 2]. Odražené laserové světlo ze záblesků se zachytí rychlou videokamerou. Rozdíl period monochromatického laserového světla odraženého z různých míst na osvětlené ploše s referenčním laserovým paprskem, přivedeným do kamery z laseru, vytváří přímo interference, které se na jednotlivých pixelech světlocitlivého

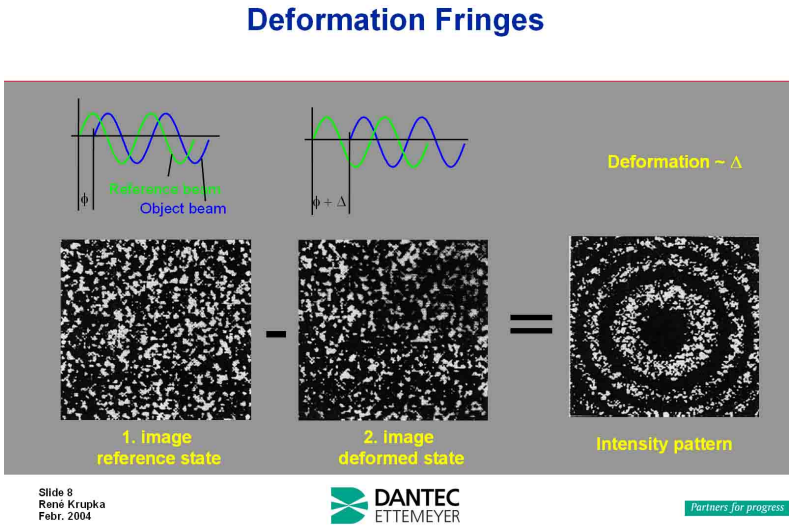
čipu kamery projeví jako tmavší a světlejší body (speckle) v ploše jednotlivých snímků [2, s. 3 - 7]. Proces zachycuje obrázek 2.1.



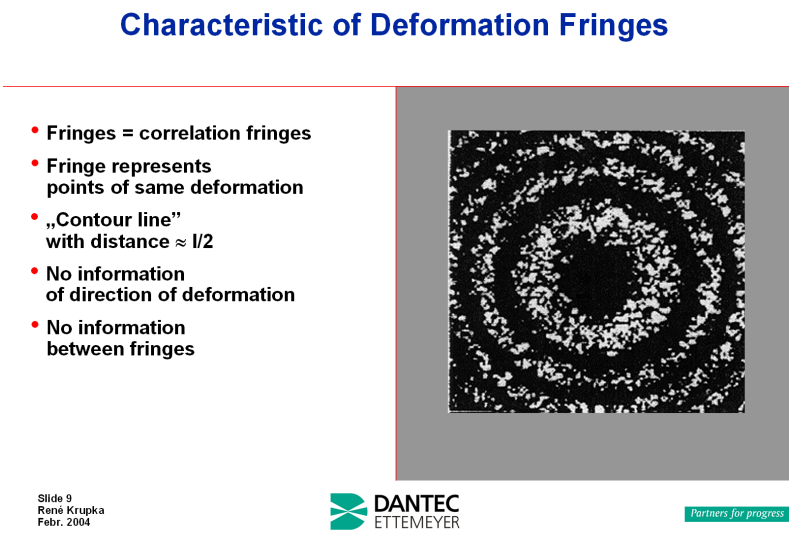

Obrázek 2.1: Schema laserové speckle interferometrie [3, s. 6]

Každý záblesk vytvoří jeden speckle obraz, který znázorňuje aktuální polohu všech bodů sledovaného objektu. Pokud mezi záblesky dojde k pohybu sledovaného objektu, tak se pozice a velikosti speckle na druhém snímku oproti prvnímu bude lišit. První speckle obraz se nazývá reference a druhý je recording [3, s. 8].

Výpočtem [3, s. 10] se pak z rozdílnosti speckle bodů mezi recording od reference, získá intenzitní obraz, který charakterizuje stav měřeného objektu pomocí míst se stejnou velikostí posunutí za dobu t mezi snímky (jde o hodnoty rychlosti, které za dobu mezi snímky představují „vrstevnice“ stejného posunutí). Oba speckle obrazy (recording a reference) a výsledné „vrstevnice“ jsou zobrazeny na obrázku 2.2.




Obrázek 2.2: Odečtení recording od reference [3, s. 8]

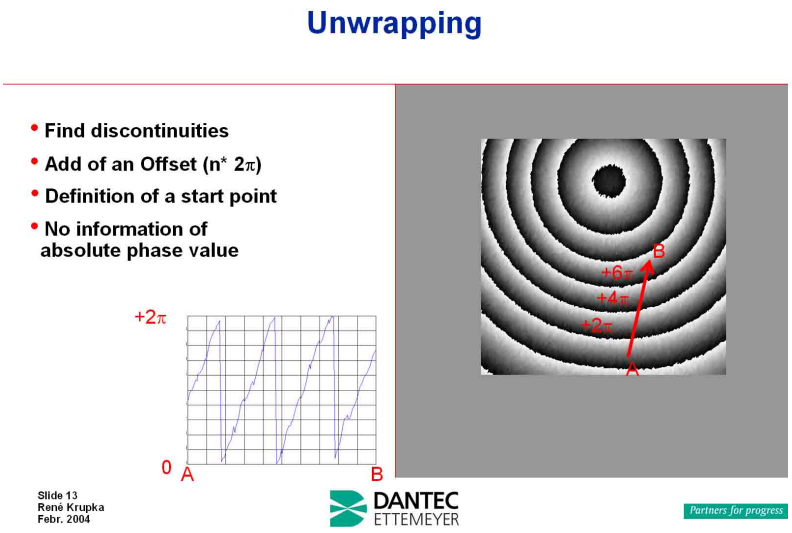



Obrázek 2.3: Výsledek odečtení recording od reference [3, s. 9]

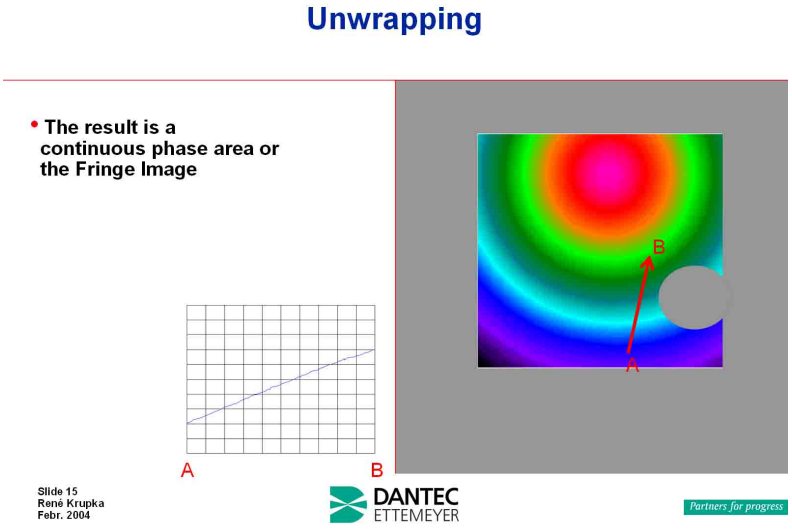
Pokud existují v intenzitním obrazu, který vznikl odečtením reference a recording v jednotlivých sousedících místech, stejné přechody z bílé do černé (určité shodné stupně šedi, viz např. bílé či černé kruhy na obrázku 2.3), pak všechny tyto body jsou součástí jedné vrstevnice. To tedy znamená, že body v rámci jedné vrstevnice jsou ve stejné výškové poloze. Takovýto vrstevnicový obraz se dále zpracuje a vyhladí. Výsledkem je obraz zvaný fázová mapa.

Na obrázku 2.4 jsou vyobrazeny vrstevnice fázové mapy. Počet vrstevnic mezi dvěma body určuje průhyb (změna vzdálenosti od kamery), který nastal mezi těmito body za dobu $t$ mezi jednotlivými snímky dvojsnímku. Rozdíl vzdálenosti obou bodů je dán počtem přechodů černé v bílou, každý jeden přechod představuje vlnovou délku použitého laserového světla (zde 660 nm). Tuto fázovou mapu je možno zobrazit i v barvách RGB, kde barvy zobrazují výšku jednotlivých vrstevnic. Na obrázcích 2.4 a 2.5 je znázorněn výpočet pro náhodně zvolené body A a B. Na obrázku 2.5 např. červená barva zobrazuje nejvyšší vrstevnici.

Na obrázku 2.6 je vidět fázová mapa resonanční desky houslí. Ta pak může být převedena na barevný obraz průhybů na ploše objektu za čas $t$ (obrázek 2.7).

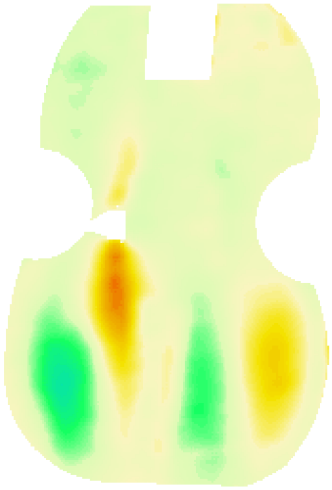



Obrázek 2.4: Vrstevnice ve fázové mapě [3, s. 13]




Obrázek 2.5: Vrstevnice ve fázové mapě, v barvách RGB [3, s. 15]

Obrázek 2.6: Fázová mapa resonanční desky houslí



Obrázek 2.7: Mapa průhybů resonanční desky houslí za čas t

### 2.1.2 Nulový bod

Každý relativní posun za dobu t mezi dvojicí snímků použitých k výpočtu fázové mapy, který mapují vypočtené vrstevnice, se musí vztahovat k nějakému bodu. Tento bod se nazývá nulový, nebo také referenční bod. Analogicky to lze přirovnat k nadmořské výšce, ta „je vyjádření výškového rozdílu konkrétního místa vzhledem k hladině moře“ [4].

Nulový bod měření lze tedy chápat jako „referenční hladinu“, ke které se ostatní body vztahují.

Pro dosažení exaktních výsledků měření je nezbytné, aby nulový bod, tedy reference, ze které se při měření vychází, byl skutečně nulovým bodem, tedy aby jeho rozkmit byl v průběhu měření nulový.

V současnosti je tento bod před zahájením měření do software ISTRA zadáván manuálně na základě zkušeností obsluhy. Pokud však nastane situace, že poloha nulového bodu není zadána správně, je nutné měření opakovat s novým zadáním. Ze série snímků zachycujících rovnoměrně kladné a záporné fáze jedné celé periody při sinusovém buzení kmitů je možné polohu nulového bodu zjistit. Takto získaný nulový bod se následně použije v dalším měření.

### 2.1.3 Buzení testovaného objektu

Pro účely měření vlastností hudebního nástroje, např. houslí, je třeba simulovat „hraní“ na tento nástroj. Tento proces se nazývá buzení objektu. Objekt je buzen sinusovým signálem pomocí zařízení na obr. 2.8.

Kovová část, zvýrazněná na obrázku 2.9 se v poli stacionárního magnetu při napájení střídavým proudem požadované frekvence rozkmitá vzájemným silovým působením střídavého a stacionárního magnetického pole. Tím dochází k mechanickému buzení objektu, u strunných smyčcových nástrojů např. kobylky. Touto metodou je však možné budit jakýkoli objekt.

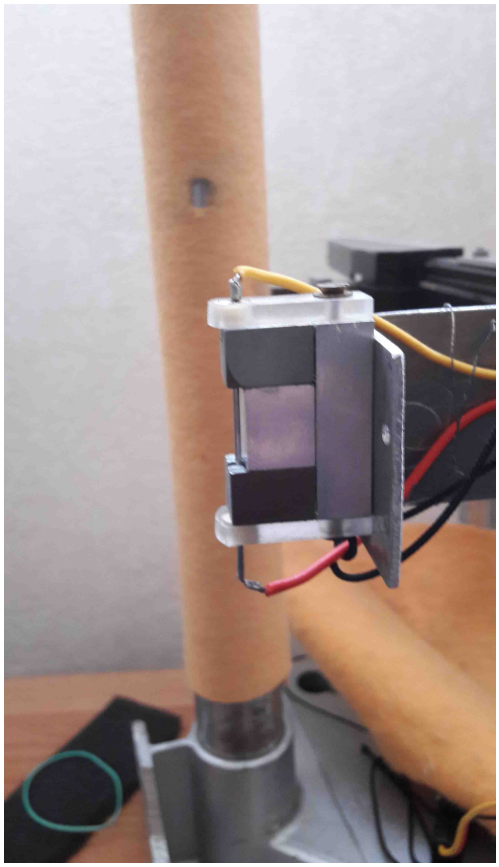
Za jednu periodu buzení objektu se naměří předem zvolené množství dvojsnímků. Dvojsnímkem se rozumí reference a recording snímek. Ve výsledku získáme předem zvolený počet fázových map, které znázorňují pohyb objektu v čase. Snímky reference, recording a fázová mapa jsou uloženy aplikací ISTRA do souboru s příponou .tfs a jedná se o vlastní formát dat, který není nikde dokumentován. Tento soubor lze otevřít a zobrazit v existující aplikaci ISTRA.

### 2.1.4 Mapa okrajů sledované oblasti

Součástí snímků je pozadí, které se nachází v laboratoři za měřeným objektem.

K odělení pozadí a měřené plochy slouží mapa okrajů sledované oblasti, (tzv. maska, či border), definující oblast, která má být v testech uvažována. Mapa okrajů oblasti se zadává v grafickém editoru aplikace ISTRA, je uložena na disk a pak se v testech při vyhodnocování ISTROU používá.

Maska se skládá z jednoduchých geometrických obrazců jako je např. přímka, kruh či kruhová výseč. Grafický editor ISTRY však neumožňuje ce-



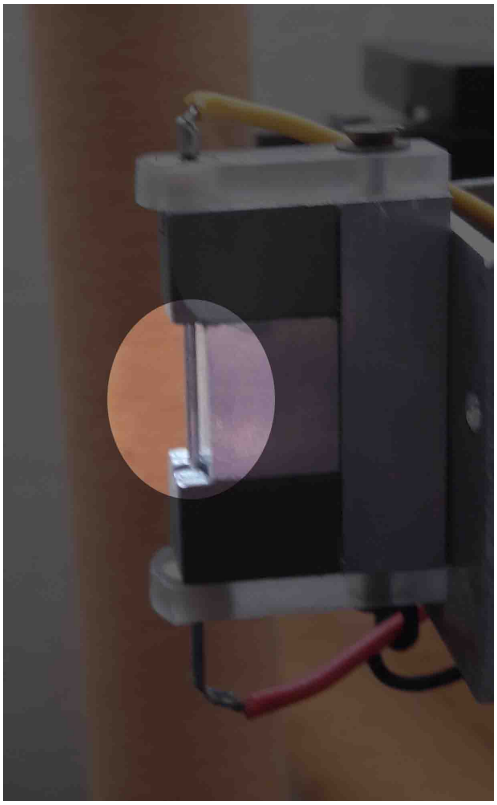
Obrázek 2.8: Budič testovaného objektu

lou definovanou masku posunout či otočit v určitém směru. Je možné pouze upravovat dílčí geometrické obrazce, ze kterých je složena. Pokud se tedy již vytvoří maska pro určitý hudební nástroj a pak se nástroj pootočí či posune, je nutné celou masku ručně v grafickém editoru ISTRY předělat, což zahrnuje manuální posun jednotlivých bodů, ze kterých se skládá, na novou pozici metodou bod po bodu.

Navíc, ISTRA barevně nerozlišuje např. polohu křížku, tudíž je někdy velmi těžké jej v editoru najít.

### 2.1.5 Jasově nepoužitelné snímky

Pro vytvoření dvou speckle obrazců se interferometr nabije. Ideálně první polovinu fotonů využije na první záblesk a zbytek fotonů na druhý záblesk. K nastavení slouží tzv. bias, což je hodnota udávající, na jak dlouho se otevře komora a tedy kolik fotonů se na záblesk použije. Tuto hodnotu lze ručně přenastavit v aplikaci ISTRA. Během měření se provádí i test na vypočtení ideální hodnoty biasu tak, aby byla distribuce fotonů na dva záblesky rov-



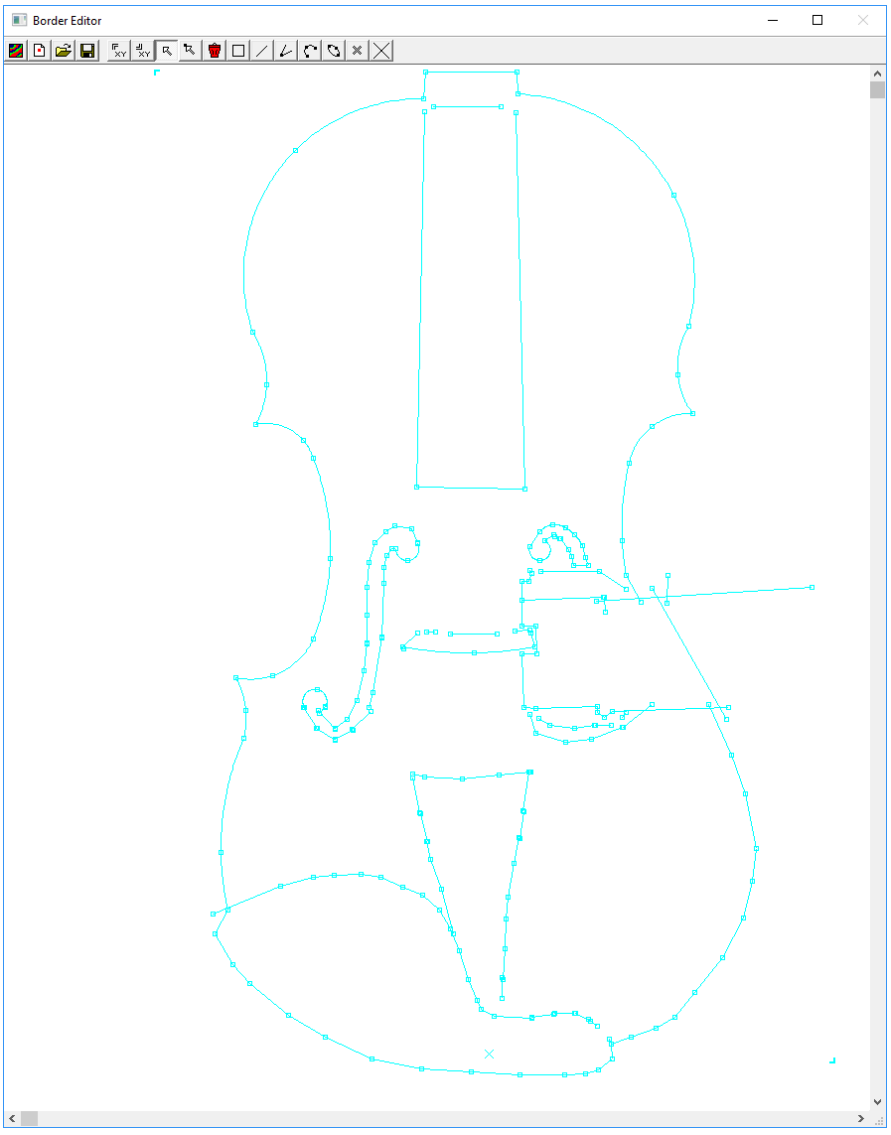
Obrázek 2.9: Kmitající část budiče

noměrná. Zpočátku měření je interferometr nezahřátý a s postupem času jeho teplota stoupá a kolísá. I navzdory výše uvedeným výpočtům hodnoty biasu se stane, že distribuce fotonů na snímky je nerovnoměrná. To má za následek, že jeden z dvojice obrazů (reference a recording) je přesvětlený a druhý je podsvětlený. To znemožňuje výpočet použitelné fázové mapy a měření se musí opakovat.

ISTRA rozpoznávání tohoto stavu neumožňuje a je nutno manuálně smímky otevřít v ISTRA editoru a vizuálně posoudit.

## 2.2 Analýza existujících řešení

V současné době neexistuje žádný modul či aplikace, umožňující nalezení nulového bodu či vyloučení nepoužitelných snímků na základě jasu. Tvorbu a úpravu mapy okrajů sledované oblasti umožňuje již existující aplikace ISTRA. Ukázka editoru mapy oblasti v aplikaci ISTRA je vidět na obrázku 2.10. Editor umožní pouze posun jednotlivých částí mapy, avšak nikoliv posun mapy jako celku. Z toho důvodu je jakýkoli posun mapy velmi pracný.




Obrázek 2.10: Editor mapy okrajů sledované oblasti v aplikaci ISTRA

Kapitola 3

# Praktická část

## 3.1 Analýza uživatelských požadavků (požadavků zadavatele)

V této kapitole shrnu informace o aplikaci z pohledu zákazníkem požadovaných funkcionalit.

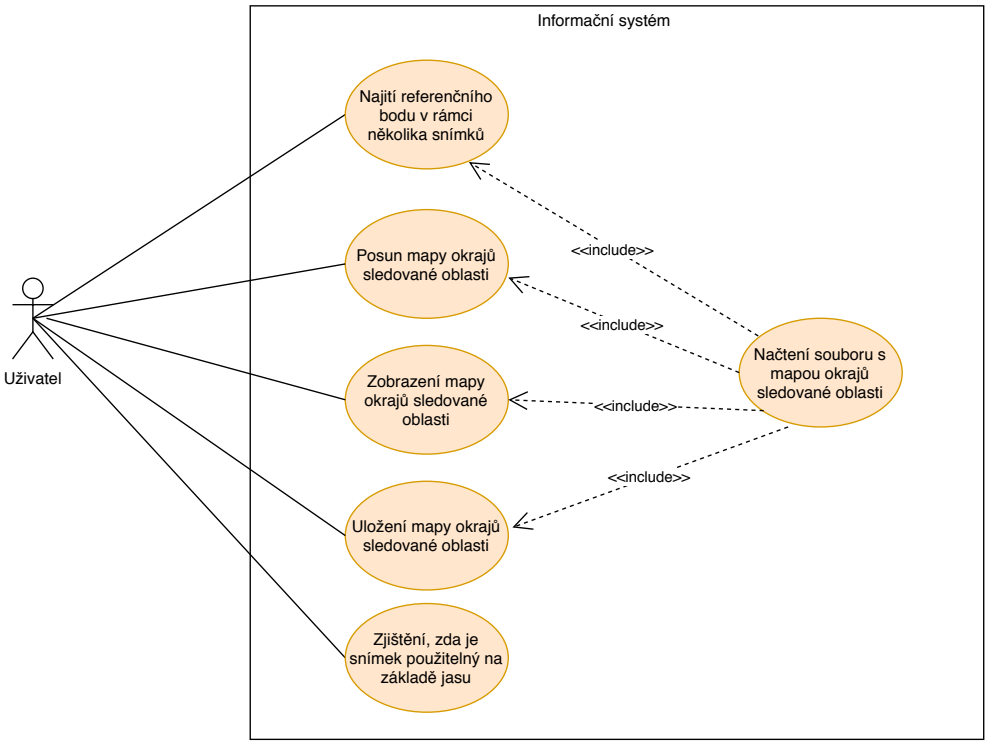
### 3.1.1 Funkční požadavky

Aplikace musí být schopna načíst soubor s mapou okrajů sledované oblasti (v diagramu 3.1 označeno jako BER soubor), umožnit interaktivní posun této mapy a uložit změny. Dále musí aplikace umožnit načtení dat z měření (v diagramu 3.1 označeno jako TFS soubor) a vyhodnotit, zda speckle obrazce načtené z tohoto souboru jsou dále použitelné na základě jejich jasu. Další požadovanou funkcionalitou aplikace je výpočet polohy referenčního bodu pro sérii speckle obrazců (tedy sérii souborů TFS).

Vnější pohled na modelovaný systém aplikace zachycuje UML[5] diagram užití aplikace na obrázku 3.1.

Diagram tříd na obrázku C.1 zobrazuje třídy a jejich vazby v celém projektu.

Pro tvorbu třídy s názvem LineEquation jsem se inspiroval online [6]. Kód je z velké části převzatý. Kód, který je převzatý, je v komentářích ve zdrojovém kódu označen. Ostatní metody a konstruktor jsem upravil a další metody jsem také přidal, jelikož jsem implementoval jiný způsob ukládání dat a například i průsečík přímek jsem počítal jinak. Třídy, které jsou převzaté, jsem nemodeloval a v UML diagramu jsou tedy pouze proměnné a metody, které jsou buď mnou modifikované, nebo celé mým dílem. V UML diagramu také nejsou zahrnuty metody ve třídě Form1, které jsou generované vývojovým prostředím.




Obrázek 3.1: Diagram užití aplikace

### 3.1.2 Nefunkční požadavky

Mezi nefunkční požadavky patří

#### Udržitelnost

- Je nutno zajistit čitelnost kódu a kvalitní dokumentaci pro další vývoj.

#### Spolehlivost

- Každá aplikace musí být spolehlivá, a proto i tato aplikace musí být ošetřena na chybné vstupy a všechny situace, které mohou nastat.

## 3.2 Analýza a návrh

V této kapitole se věnuji analýze struktury dat v souboru s fázovou mapou a analýze struktury souboru mapy okrajů sledované oblasti.

### 3.2.1 Analýza souboru s fázovou mapou

Fázová mapa ve stupních šedi, vztahující se k měřenému objektu, je ISTROU uložena do souboru s příponou .tfs (dále jen TFS soubor), formát dat není nikde dokumentován a abych mohl ze souboru data přečíst, musel jsem nejdříve zjistit, jak a v jakém formátu jsou data v souboru uložena.

Nejdříve jsem se pokusil soubor přečíst jako obrázek klasických formátů (.png, .tiff, .jpg). Tímto způsobem jsem však nebyl schopen získat smysluplná data. Dále jsem zkusil soubor číst po bajtech, kde jsem skupinu 4 bajtů uvažoval jako reprezentaci jednoho pixelu. Ačkoli jsem si byl vědom, že je možné, že soubor má vlastní hlavičku, předpokládal jsem, že od určité části se mi zobrazí odpovídající obrazová data. Dále jsem soubor zkusil číst odzadu, a konečně jsem opakoval všechny postupy s tím, že jsem jako jedno obrazové dato uvažoval jeden bajt. Pokud by na konci nebyla žádná „patička“, tak by se mi mohlo podařit obrazová data zobrazit. Výše uvedené postupy však nevedly k výsledku.

Zobrazil jsem si tedy každý bajt jako jeden ascii znak pomocí programu s názvem „od“ na linuxu.

Zjistil jsem, že v hlavičce je mimo jiné uložena i cesta k souboru a že soubor má také patičku, která oznamuje konec souboru, což je patrné z obrázku 3.2, kde je vidět text „Recording“.

Na obrázku 3.3 je zobrazen začátek souboru, kde jednotlivé bajty jsou převedeny na čitelné znaky. Zvýrazněná je část, kde je název a umístění souboru, pravděpodobně v době, kdy byl vytvořen.

Potřeboval jsem zjistit, na jaké pozici (offsetu) začíná první obrázek v souboru. Proto jsem vytvořil aplikaci, která přepsala hodnotou 0 určité množství dat v souboru TFS. Nejprve jsem se pokusil přepsat celou hlavičku a ideálně i

```
root@horse:~/Desktop# od -c 475\ 1200.TFS | tail
23770200 025 032 035 035 036   # 035 026 025 026 026 025 020 025 024 026
23770220 033 034 034 031 030 027 026 024 016 020 030 036 034 027 026 025
23770240 022 024 026 026 023 022 022 022 024 024 020 022 027 030 036 036
23770260 027 033   ! 032 020 022 025 025 022 022 025 032 032 027 017  \r
23770300 022 025 027 024 020 020 026 030 024 024 037 027  \r 021 017 023
23770320 030  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
23770340  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
24000320  \0   R   e   c   o   r   d   i   n   g  \0
24000333
```

Obrázek 3.2: Zobrazení konce .TFS souboru

```
root@horse:~/Desktop# od -c 475\ 1200.TFS | less
0000000   I   I   *  \0  \b  \0  \0  \0  \f  \0   ? 004 002  \0 037  \0
0000020  \0  \0 236  \0  \0  \0 320  \a  \t  \0 001  \0  \0  \0  \0 004
0000040  \0  \0 321  \a  \t  \0 001  \0  \0  \0  \0 004  \0  \0 322  \a
0000060  \v  \0 001  \0  \0  \0  \0  \0 200   ?   9  \b 001  \0  \0  \0
0000100 020  \0 275  \0  \0  \0   ?  \b 002  \0  \n  \0  \0  \0 275  \0
0000120 020  \0 235  \b 001  \0  \0  \0 020  \0 307  \0 020  \0 243  \b
0000140 002  \0  \n  \0  \0  \0 307  \0       \0 244  \b 001  \0  \0  \0
0000160 020  \0 321  \0       \0 245  \b 001  \0  \0  \0 020  \0 321  \0
0000200   0  \0 001  \t 001  \0  \0  \0 020  \0 321  \0   @  \0  \a  \t
0000220 002  \0  \n  \0  \0  \0 321  \0   P  \0  \0  \0  \0  \0   D   :
0000240   \   Z   d   e   n   e   k   \   s   c       b   a   s   2   \
0000260   4   7   5       1   2   0   0   .   T   F   S  \0 204   |   }
0000300 177 200 201 200 177   ~   ~ 177 206 217 221 216 216 220 220 217
0000320 217 220 225 234 242 247 251 252 252 253 252 246 235 222 211 210
0000340 215 223 231 240 253 270 241 202 207 216 221 223 224 225 230 234
0000360 241 245 243 234 223 211 201   | 201 220 230 232 234 236 241 241
0000400 237 233 227 222 212   ~   g     035   v 221 226 227 225 224 226
0000420 234 247 255 255 255 257 260 260 256 253 246 243 241 240 240 243
0000440 252 253 250 247 250 251 251 245 237 226 214   ~   z 233 234 233
0000460 232 232 233 236 241 245 250 253 255 256 250 206   u   s   u   x
0000500   {   ~ 203 205 206 204 204 205 210 214 222 227 233 234 222 216
0000520 216 221 226 233 240 245 251 254 257 261 261 260 262 272 270 270
0000540 270 272 273 273 272 271 271 270 264 257 250 242 237 235 235 237
0000560 244 253 261 232 205 206 216 233 244 242 227 214   }   u   | 223
0000600 267 313 324 331 333 335 340 343 343 351 367   #   N   O   S   \
0000620   h   s   } 205 215 223 227 232 235 241 246 256 255 237 234 233
0000640 231 230 232 253 267 273 273 266 256 243 232 226 225 226 232 242
0000660 247 255 262 264 257 242 223 205   ~   }   ~ 177 200 204 211 216
0000700 222 227 232 227 220 213 212 215 224 233 237 237 234 231 230 227
0000720 225 221 215 212 213 216 224 232 241 245 227 213 211 213 217 222
0000740 225 225 224 224 226 225 220 212 206 203 177 177 206 217 225 230
0000760 234 241 247 255 264 270 274 276 275 272 267 304 306 277 264 250
0001000 234 222 217 221 230 241 261 317 345 357 360 353 344 333 317 302
0001020 276 300 301 301 300 312 323 312 274 263 254 245 233 215   ~   q
0001040   m   q   y   z 220 316 333 333 316 307 310 307 304 275 260 263
0001060 026 034   c 231 241 247 252 250 242 225   g   +   # 032 017 002
0001100 347 317 310 307 310 313 320 326 333 341 350 356 364 367 370 371
0001120 370 366 363 361 355 361 337 323 306 261 275 300 274 266 262 262
0001140 263 264 264 263 262 257 252 244 237 235 237 246 252 253 252 250
0001160 246 245 250 254 255 256 260 264 266 266 261 247 231 212 207 221
0001200 232 240 243 246 260 335 360 355 342 331 321 313 306 305 306 306
:
```

Obrázek 3.3: Zobrazení začátku .TFS souboru

nějaká následující data s tím, že uvidím po otevření obrázku v ISTRA nějakou změnu. Ukázalo se, že aplikace ISTRA není ošetřena na výskyt nečekaných hodnot v souboru a přestala pracovat. Metodou pokus–omyl jsem v souboru našel pozici začátku dat, která jsem mohl přepsat, aniž by ISTRA spadla. Zjištěná velikost hlavičky souboru TFS je 190 bajtů (od pozice 0 do 189), všechny soubory mají velikost 5 121 kB.

Po otevření souboru dat ISTROU jsem následně v levé horní části obrázku zaznamenal černý proužek signalizující, že se mi podařilo přepsat začátek obrazových dat. Poté jsem již jen upravoval počet zapsaných bajtů a našel jsem tak i konec obrázku a dále zjistil, že každý bajt tedy určuje jeden pixel ve stupních šedi (0 - černá, 255 - bílá). Zjistil jsem tedy i dimenze obrázku, a podařilo se mi přečíst i druhý obrázek, který byl uložen v souboru hned za tím prvním. Identifikoval jsem, že první obrázek je fázová mapa a druhý obrázek je reference.

Každý bod fázové mapy je reprezentován hodnotou bajt, tedy nabývá hodnot od 0 do 255, což je 256 možných hodnot. Musí ale existovat hodnota, která reprezentuje nulový pohyb bodu. To by ideálně měla být prostřední hodnota. Pokud bychom měli například jen čísla 1, 2, 3, prostřední hodnotou by bylo číslo 2. Pro celkový počet čísel 256 ale takový reprezentant neexistuje. Potřeboval jsem tedy nějak odhadnout hodnotu, která určuje nulovou výchylku.

Hodnoty ve fázových mapách se pohybují od 0 do 255 a předpokládá se, že hodnota pro nulovou výchylku je 127. Tento předpoklad jsem ověřil na všech TFS souborech, které jsem měl k dispozici. Hledal jsem nejvyšší a nejnižší hodnoty, které se ve fázových mapách vyskytovaly a pro každou mapu jsem vypočetl hodnotu pro nulovou výchylku. Tyto hodnoty jsem zprůměroval a výsledkem bylo číslo 127.

### 3.2.2 Analýza souboru mapy okrajů sledované oblasti

Mapa okrajů sledované oblasti (dále jen BER soubor) je ISTROU uložena do souboru s příponou BER. Jedná se o textový soubor.

Soubor obsahuje na každém řádku čísla oddělená mezerou. V ISTŘE jsem si vždy vytvořil novou mapu okrajů, do které jsem umístil jen jeden obrazec, např. úsečku. Výsledný soubor jsem analyzoval. Tím se mi pro všechny obrazce, které lze v mapě použít, podařilo zjistit, jak jsou ukládány. Navrhl a implementoval jsem program, který soubor přečetl a vykreslil celou mapu, umožnil uživateli posouvat celou mapu v libovolném směru a změny uložil zpět do souboru. Zjistil jsem, že ani v tomto případě není ISTRA ošetřena proti výskytu neočekávaných hodnot a pokud se do BER souboru dostanou záporné hodnoty souřadnic geometrických útvarů, ISTRA po načtení takového souboru přestane pracovat. Implementoval jsem tedy i funkcionalitu znemožňující uživateli uložit pro ISTRU nekorektní hodnoty.

## 3.3 Vlastní řešení

V úvodu této kapitoly diskutuji možnost zakomponování software do existující aplikace ISTRA. Následuje popis zvolené technologie a popis struktury projektu.

### 3.3.1 Možnost zakomponování řešení do aplikace ISTRA

K aplikaci ISTRA, která řídí měření, mi byly poskytnuty zdrojové kódy, které však nejsou veřejně dostupné. Kódy jsou v jazyce Microsoft Visual C++. Součástí jsou také knihovny v assembleru. Soubory zdrojových kódů mají dohromady velikost kolem 2,50 GB. Na vývoji se zjevně postupně podílelo více programátorů, jelikož názvy funkcí jsou anglicky i německy, najdeme zde však i české komentáře. K celému projektu není k dispozici dokumentace kódu, kromě nahodilých komentářů v těle funkcí. Existují funkce, v jejichž těle je pouze volání další funkce. Takovýto řetězec kódu má někdy i 4 články. Zakomponování řešení do aplikace ISTRA značně převyšuje rozsah bakalářské práce. Řešením je tedy prozatím externí aplikace. Zvolené řešení a algoritmy se v budoucnu zakomponují do zdrojového kódu ISTRY.

### 3.3.2 Programovací jazyk

Jako programovací jazyk jsem zvolil jazyk C# [7]. Důvodem je, že po aplikaci se požadují interaktivní úpravy mapy oblasti, a celá grafická část se dá efektivně vyřešit pomocí knihoven .NET Frameworku. To značně urychlí práci a dovolí více se soustředit na implementaci funkcionality aplikace. Tento jazyk jsem použil i v procesu analýzy souborů protože v průběhu zkoumání měřených dat bylo třeba kód často upravovat.

### 3.3.3 Struktura projektu

Třídy v projektu jsou rozděleny do namespace. K problematice načtení a editace borderu se vztahuje namespace „Border“. Namespace „SpeckleBrightness“ se vztahuje k analýze snímků na základě jasu. Třídy vztahující se k výpočtu polohy referenčního bodu jsou v namespace „ReferencePointSolver“.

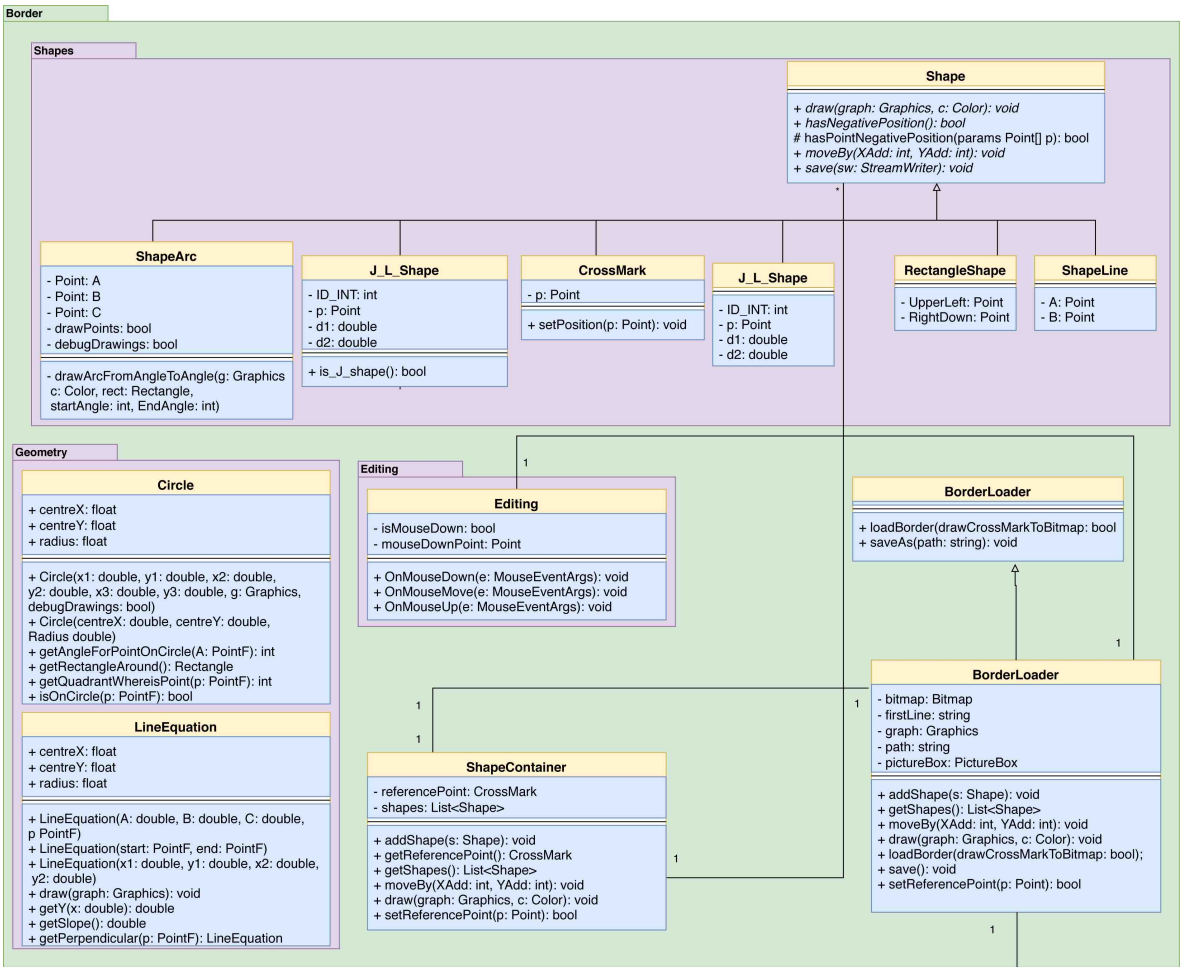
Popis funkcionalit všech tříd je v příloze, kde je třída uvedena i s namespace, ve kterém se nachází. Podrobnější popis tříd je pak v dokumentaci, vygenerované programem DOXYGEN, na přiloženém USB.

## 3.4 Realizace

V této kapitole je popsáno implementační řešení interaktivních úprav mapy okrajů sledované oblasti, identifikace nepoužitelných speckle obrazů a popis implementace algoritmu pro výpočet nulového bodu.

### 3.4.1 Interaktivní úpravy mapy okrajů sledované oblasti

Třída Border.BorderLoader se stará o načítání, management a ukládání mapy okrajů sledované oblasti. Každý geometrický útvar či objekt, který je v masce, reprezentuje samostatná třída. Všechny tyto třídy dědí ze třídy Shape. Struktura je patrná z části diagramu tříd na obrázku 3.4.

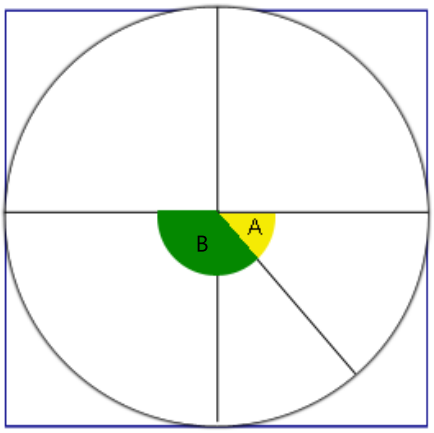



Obrázek 3.4: Část diagramu tříd v namespace „Border“

Ze souboru jsou načteny všechny řádky a vytvořeny instance odpovídajících tříd. Takto jsou realizovány všechny jednoduché geometrické útvary, tedy například instance třídy reprezentující úsečku, bod či kruhovou výseč. Je však nutné pracovat s daty, která jsou ve vstupním souboru. Úsečka je definována dvěma body, a tedy její vykreslení je triviální, jelikož existuje funkce DrawLine() v knihovnách .NET, která vykreslí úsečku, jako argument jsou požadovány dva body. Analogické bylo zobrazení obdélníku a křížku reprezentujícího nulový bod. Poloha referenčního bodu je pak v aplikaci vykres-

lena tak, aby bylo snadné ji najít. Nejsložitější bylo zobrazení kruhové výseče, definované třemi body, kterými prochází. V knihovnách .NET existuje funkce DrawArc(), která vykreslí kruhovou výseč, ale jako argumenty požaduje čtverec opsaný kruhu, jehož výseč vykreslujeme, dále pak počáteční úhel ve stupních, určující začátek výseče a konečně druhý úhel ve stupních.

Problematiku znázorňuje obrázek 3.5. Zde úhel A určuje, kde kruhová výseč začíná. Úhel B určuje velikost výsledné kruhové výseče. Modře je znázorněn čtverec opsaný uvažovanému kruhu. Pokud bychom do funkce DrawArc() předali jako argumenty čtverec (instance objektu Rectangle), úhel A a úhel B, pak by došlo k vykreslení části kružnice, definované úhlem B.




Obrázek 3.5: Vykreslení kruhové výseče

Bylo nutné vyjádřit rovnici kružnice ze znalosti tří bodů, opsat jí čtverec a vypočítat výše uvedené úhly ve stupních.

Všechny třídy reprezentující geometrické elementy dědí ze třídy Shape a jsou přidány do kontejneru reprezentovaného instancí třídy ShapeContainer.

Situaci znázorňuje obrázek 3.4.

Pokud uživatel chce posunout celý border, ve třídě ShapeContainer se zavolá funkce moveBy() a následně se pro každý „Shape“ změní pozice bodů pomocí metody Shape.moveBy(). Pak se znovu všechny objekty graficky vykreslí.

Uložení je pak funkce inverzní k načítání a uloží aktuální stav pozic geometrických útvarů.

### 3.4.2 Identifikace nevhodných dvojic speckle obrazů na základě jasu

Pro tuto operaci je potřeba vstupní soubor TFS, obsahující speckle obrazy. O načtení dat ze souboru TFS se stará třída TFSReader. Ta dědí ze třídy s názvem TFSReaderInterface. Záměrně se nejedná o interface, ale abstraktní třídu, jelikož v případě užití interface není dovoleno definovat metodu s modifikátorem přístupnosti private nebo protected. Aplikace je tímto v budoucnu snadno rozšiřitelná, změnil-li by se formát vstupních dat. Pak jen stačí vytvořit novou třídu implementující tento „interface“. Pro identifikaci nevhodných snímků je nutné pracovat buď s obrazem reference, nebo recording. Ve zvoleném řešení aplikace neprve načte do paměti obraz reference, poté je spočítána průměrná hodnota všech pixelů obrazu. Na základě vstupních dat jsem experimentálně spočítal dvě hodnoty, A, B, pro které platí, že $A < B$. Pokud je průměrná hodnota všech pixelů menší než hodnota A, pak je obraz velmi tmavý a nelze tedy speckle obrazy použít. Je-li průměrná hodnota všech pixelů větší než hodnota B, pak je obraz velmi světlý. Je-li tedy průměrná hodnota obrazových dat uvažovaného obrazu v intervalu $< A, B >$, pak je obraz validní.

### 3.4.3 Výpočet polohy nulového bodu

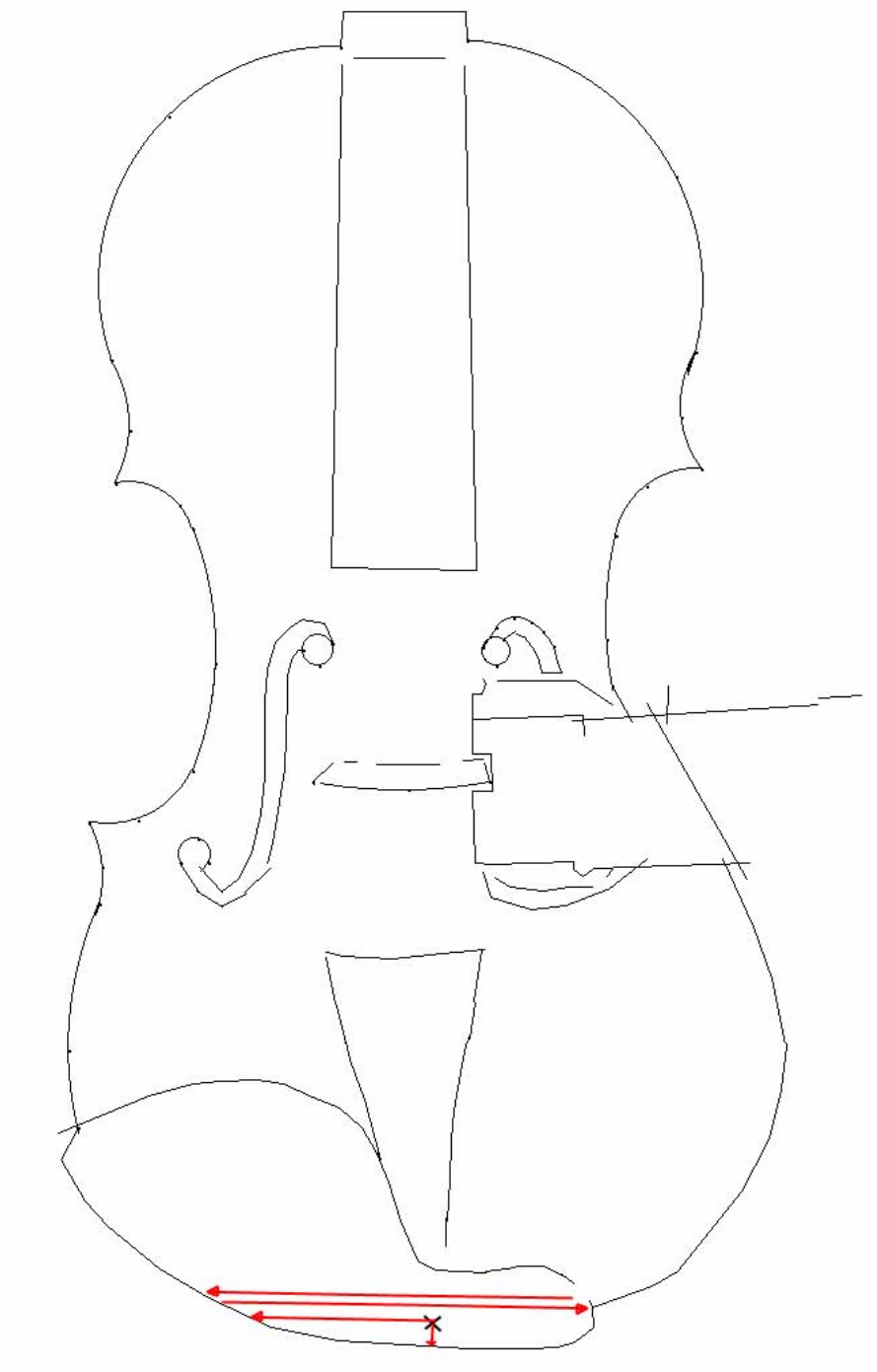
Pro výpočet polohy nulového bodu je nutné mít několik vstupních TFS souborů a odpovídající BER soubor, obsahující polohu křížku z měření. Poloha křížku může být zvolena i špatně, avšak očekává se, že je ve vnitřní části mapy okrajů oblasti.

Cílem je tedy najít bod, který v sekvenci snímků kmitá s nejmenší možnou amplitudou.

Program načte data ze souboru BER a vytvoří bitmapu, kde jsou vykresleny okraje. Pozadí této bitmapy tvoří průhledné pixely. V tomto případě se však zcela záměrně nevykreslí do bitmapy poloha křížku. Poloha křížku je známa a bude ve vnitřní části mapy okrajů. Užije-li se algoritmus BFS, lze najít všechny body ve vnitřní části mapy okrajů. Algoritmus BFS hledá nejkratší cestu v grafu. Obraz okrajů oblasti se převede na graf, kde každý pixel je jeden uzel v grafu. Dva uzly jsou spojené, pokud pixely, které reprezentují, jsou vedle sebe (nad, pod, vlevo či vpravo) a jsou průhledné.

Algoritmus BFS použitý na mapu okrajů znázorňuje obrázek 3.6. Na tomto obrázku jsou z důvodu demonstrace průhledné pixely nahrazeny bílou barvou. Původní referenční bod je pro názornost vyznačen černým křížkem. Průběh algoritmu je naznačen červenými šipkami.

Pixel je algoritmem dále zpracováván, pokud není průhledný. Z toho důvodu se do mapy záměrně nevykreslila poloha křížku, jelikož by algoritmus narazil na neprůhledné pixely kolem počátečního bodu a zastavil by se.



Obrázek 3.6: Znázornění algoritmu BFS v mapě okrajů sledované oblasti

Takto tedy bude algoritmus fungovat i při změně barvy pro vykreslování mapy okrajů oblasti. Pro každý nalezený bod, který je uvnitř mapy okrajů, se tento bod přidá do hašovací tabulky. Jelikož je cílem zjistit, které body jsou v uzavřené oblasti mapy okrajů sledované oblasti, algoritmus vždy projde všechny body. Užití jiného algoritmu (např. DFS) by algoritmickou složitost nezlepšilo. Takto je možné zjišťovat v konstantním čase, zda je bod ve vnitřní části masky. Alternativně lze přes tyto body iterovat.

Takto předvypočtené body budou pro všechny vstupní soubory TFS s obrazovými daty stejné.

Následně dojde k načtení dat z prvního snímku (fázové mapy) ze všech vstupních TFS souborů do paměti. Nulovou výchylku každého bodu fázové mapy reprezentuje číslo 127, jak bylo zjištěno v analýze. Každý bod se tedy přepočte tak, aby nulovou výchylku označovala hodnota 0, což usnadnilo proces budoucích výpočtů.

Na obrázku 3.7 je grafické znázornění dvou bodů, referenční bod je označen A, zkoumaný bod je označen B. Číselná hodnota pod body udává velikost a směr výchylky. Oba body se vychýlily ve stejném směru (hodnoty výchylek jsou kladné). Referenční bod se vychýlil o hodnotu 25 a druhý bod se vychýlil o hodnotu 100. Výchylka bodu B ale zahrnuje i pohyb referenčního bodu A. Reálná výchylka pouze bodu B tedy je

$$100 - 25 = 75$$

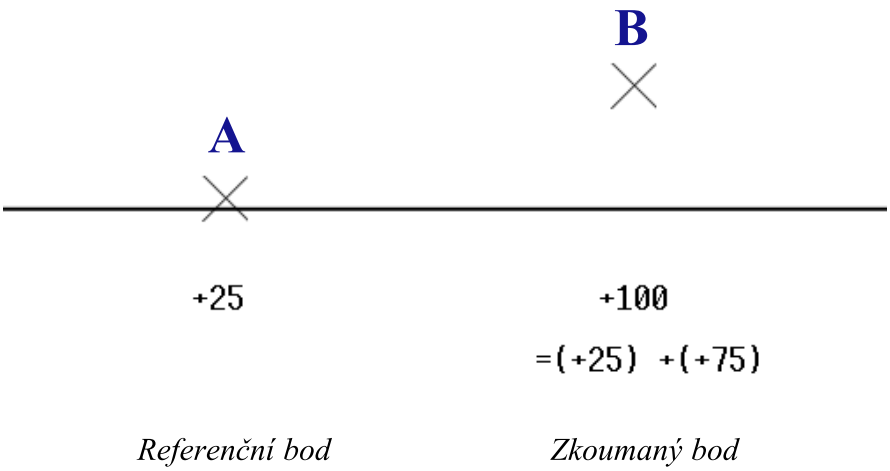
.

Pokud se odečte od výchylky každého bodu ve fázové mapě výchylka referenčního bodu, výsledkem jsou výchylky všech bodů nezávislé na pohybu referenčního bodu. Tyto všechny hodnoty se sečtou a výsledkem je celková hodnota, udávající směr a výchylku celého testovaného objektu, nezávislá na případném pohybu referenčního bodu.

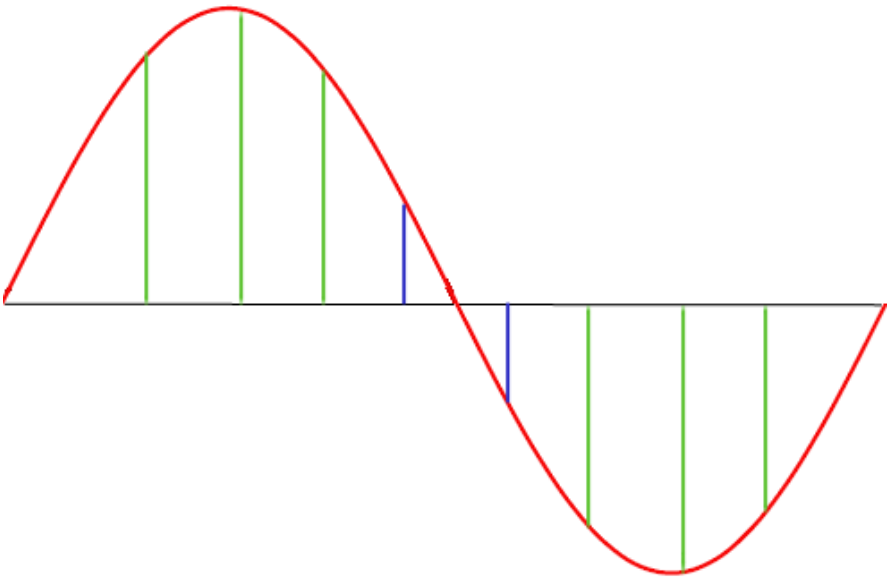
Z těchto součtů se pak vyberou dva obrazy se součtem nejblíže hodnotě 0.

Toto ilustruje obrázek 3.8. Na obrázku je zobrazena jedna perioda buzení objektu. Modře jsou v rámci periody sinu zobrazeny polohy dvou fázových map se součtovou hodnotou nejblíže nule. Tyto dva obrazy budou, vzhledem k povaze buzení objektu, v protifázi. V obrázku jsou zobrazeny modře.

Pokud se tyto dva obrazy odečtou, výsledkem je obraz, který obsahuje body s nejnižší amplitudou kmitu. V rámci tohoto obrazu pak lze najít bod s výchylkou nejblíže nule. Tento bod je pak nejlepším kandidátem na polohu nulového bodu.




Obrázek 3.7: Grafické znázornění referenčního a zkoumaného bodu



Obrázek 3.8: Grafické znázornění součtových hodnot jednotlivých fázových map

## 3.5 Testování

Testoval jsem požadované funkcionality a stabilitu aplikace. Ta je schopna načíst vstupní data a zpracovat je. Z fakulty mi bylo poskytnuto přibližně 200 dalších souborů TFS, má aplikace všechny úspěšně otevřela, načetla a zobrazila.

Obdobně jsem testoval správnou funkčnost a spolehlivost čtení souborů s mapou okrajů sledované oblasti. Pro všechy testovací vstupní soubory, kterých bylo přibližně 20, se mapu okrajů podařilo načíst, upravovat a upravenou šlo uložit.

Pro testování funkčnosti vyloučení snímků na základě jasu jsem měl jen jeden vstupní soubor. Nevalidní měření se na fakultě neskladují. Z povahy problematiky analýzy vstupního TFS souboru však není nutné mít mnoho testovacích dat, jelikož zkažené snímky budou velmi tmavé či velmi světlé.

Co se týče vypočtené polohy referenčního bodu, lišila se o přibližně 12 pixelů od polohy zadané před měřením, což svědčí o velkých zkušenostech obsluhy.

# Závěr

Úkolem práce byl návrh a implementace aplikace pro zpracování dat získaných z testů kvality hudebních nástrojů v laboratoři na fakultě HAMU v Praze. Na základě literární rešerše problematiky analýzy hudebních nástrojů jsem navrhl a implementoval software pro analýzu dat z měření. V experimentální části jsem zjistil strukturu měřených dat a navrhl možnosti jejich zpracování. Navrhl a implementoval jsem aplikaci, která pro sérii dat měření uložených v souborech identifikuje polohu bodu s nejmenším průhybem ve sledovaném objektu v sekvenci snímků z jedné periody jedné frekvence buzení objektu a umístí polohu vypočteného referenčního bodu mezi okraje sledované oblasti pro další zpracování obrazců dané sekvence.

Dále umožní uživateli interaktivní úpravy okrajů oblasti, ve které se během měření nachází sledovaný objekt a také je schopna porovnat jas dvojice speckle obrazců a vyloučit snímky nevhodné pro další zpracování.

Výsledná aplikace je napsaná v jazyce C# a je schopná zpracovávat všechna validní vstupní data. Díky této aplikaci je možné zjednodušit, urychlit a zpřesnit zpracovávání výsledků.

Nejdříve jsem se zabýval analýzou obsahem a formátem souborů obsahujících vstupní data. Jednalo se tedy o soubory s fázovou mapou a soubory obsahující mapu okrajů sledované oblasti v rámci měření. Obsah těchto souborů nebyl nikde zdokumentován, tudíž jsem musel manuálně zjistit, jak tato data načíst, abych s nimi pak dále mohl pracovat.

Zdrojový kód bude v budoucnu možno zakomponovat do aplikace ISTRA, což zvýší produktivitu měření.

Všechny požadavky zadavatele jsem splnil.

# Literatura


[1] KIM, K.-S.; CHANG, H.-S.; JUNG, S.-W.; aj.: Determination of Elastic modulus of thin materials by Speckle Interferometry: str. 6. [cit. 2018-03-4]. Dostupné z: `https://pdfs.semanticscholar.org/025f/06de4759e54531a6549f31da31dc46a64d52.pdf`

[2] Erne, O.: Electronic Speckle Pattern Interferometry: Temporal vs. Spatial Phase-Shifting. [online], [cit. 2018-03-4]. Dostupné z: `http://web.pdx.edu/~larosaa/Applied_Optics_464-564/Projects_Presented/Oliver_Speckle_Pattern_Interferometry_Report3.doc`

[3] DANTEC ETTREMEYER GmbH, Kässbohrerstr. 18: *11th International User Meeting*. 2004, doi:10.1.1/jpb001.

[4] Co to je Nadmořská výška? Význam slova. březen, [cit. 2018-03-4]. Dostupné z: `http://cojeto.superia.cz/veda/nadmorska_vyska.php`

[5] Pender, T.: *UML bible*. John Wiley & Sons, Inc., 2003.

[6] PasteBin - LineEquation. duben 2018, [cit. 2018-03-4]. Dostupné z: `https://pastebin.com/iQDhQTFN`

[7] Perry, S. C.: *Core C# and. NET*. Prentice Hall PTR, 2005.

Příloha **A**

# Seznam použitých zkratek

**BER** Označuje soubor s mapou okrajů oblasti sledovaného objektu, mající příponu .BER

**BFS** Anglicky Breadth First Search, označuje algoritmus pro vyhledávání nejkratší cesty v grafu

**HAMU** Hudební a taneční fakulta Akademie múzických umění

**RGB** Anglicky Red, Green, Blue, značí rozsah barev

**TFS** Označuje soubor obsahující tři specle obrazy, fázovou mapou, recording a reference obraz. Tento soubor má příponu .tfs

Příloha B

# Popis všech tříd v implementačním řešení

Zde je popis tříd v celé aplikaci. Pokud je třída součástí namespace, je to uvdeno před jejím názvem.

### Form1

Třída Form1 obsahuje metody, které se zavolají po vyvolání události (např. kliknutí na tlačítko či stisk myši). V těchto metodách je dále realizována funkčnost aplikace.

### Settings

Tato třída obsahuje různé nastavení programu tak, aby bylo jednoduché jej v budoucnu najít a případně změnit. Jedná se například o barvy použité při vykreslování grafiky.

### Border.BorderLoader

Třída BorderLoader má na starost načtení, zobrazení a manipulaci s daty ze souboru s mapou okrajů sledované oblasti. Součástí této třídy je mimo jiné metoda loadBorder(), která se pokusí načíst data ze souboru do struktur. Pokud se načtení nepovede, je zobrazen text popisující chybu. Metoda čte soubor po řádcích pomocí metody readLine(). Metoda přečtená data zpracuje. V závislosti na tom, jaký geometrický útvar je načítán, se vytvoří odpovídající objekt a ten je přidán do kontejneru. Třída dále obsahuje proměnnou container typu ShapeContainer, která obsahuje všechy načtené geometrické útvary.

Metoda redraw() se stará o vykreslení všech načtených objektů. Pokud chceme všechna data uložit zpět do souboru, užijeme metodu save().

### Border.Geometry.Circle

Tato třída reprezentuje kruh a umožňuje vytvoření kruhu definovaného třemi body. V kontruktoru se vypočítá rovnice kruhu, pomocí které pak můžeme kruh vykreslit (z ladících důvodů). Hlavní přínos této třídy je při počítání křivky (reprezentované třídou ShapeArc).

### Border.Geometry.LineEquation

Tato třída reprezentuje přímku. Umožňuje vykreslení přímky, její konstrukci, konstrukci přímky kolmé na tuto přímku, a další základní operace. Přínos je například tehdy, pokud je počítán průsečík přímek. Tato třída je využívána např. při výpočtu parametrů potřebných k zobrazení křivky (reprezentované třídou ShapeArc).

### Math

Tato třída obsahuje pomocné matematické a geometrické funkce. Metoda getMiddlePoint() bere jako argument dva body, bod A a bod B. Vrací bod, který je středem úsečky, kterou bychom zkonstruovali z bodu A a B. Metoda RadianToDegree() přepočítá hodnotu v obloukové míře do míry stupňové.

### Border.ShapeContainer

Obsahuje spojový seznam všech objektů typu Shape, reprezentující geometrické útvary načtené ze souboru s mapou okrajů sledované oblasti. Zde můžeme objekty přidávat pomocí metody addShape(), ale hlavně je všechny i posouvat, k čemuž slouží metoda moveBy(int XAdd, int YAdd). Ta jako argumenty bere dvě čísla, udávající posun na ose X a ose Y. Metoda draw() vykreslí všechny objekty ve spojovém seznamu.

### Border.Shapes.Shape

Jedná se o abstraktní třídu, ze které dědí každý objekt reprezentující geometrický útvar, který chceme zobrazovat a pracovat s ním. Při rozšíření třídy Shape je vyžadována implementace metod draw(), moveBy(), save() a hasNegativePosition().

Dále zde najdeme metodu hasPointNegativePosition(params Point[] p), která abstraktní není. Jako argument bere pole bodů, a vrací true, pokud existuje bod v předané množině, který má nějakou souřadnici zápornou. Abstraktní metoda hasNegativePosition() slouží k určení, zda geometrický útvar má nějakou souřadnici zápornou.

### Border.Shapes.CrossMark

Reprezentuje polohu křížku. Je reprezentován jen jedním bodem, zde je definován přesný formát na uložení a vykreslení.

### Border.Shapes.J_L_Shape

Reprezentuje „skobičku“ (dále jen objekt) ve tvaru písmena L nebo písmena J. Každý objekt je reprezentován čtyřmi hodnotami. První dvě určují polohu objektu v rámci borderu. Třetí hodnota určuje vzdálenost na ose X a poslední hodnota vzdálenost na ose Y. Tyto dvě vzdálenosti udávají výšku a šířku hudebního nástroje. Pro implementační část práce však nejsou podstatné.

### Border.Shapes.RectangleShape

Reprezentuje obdélník, který je definován dvěma body, kde první je levý horní roh a druhý pravý dolní roh obdélníku.

### Border.Shapes.ShapeArc

Reprezentuje křivku. Ta je definována třemi body.

### Border.Shapes.ShapeLine

Reprezentuje úsečku definovanou dvěma body.

### Border.editing.Editing

Tato třída se stará o ukládání pozice kliknutí uživatele. Pokud například uživatel posune myš za účelem posunutí celé masky, zavolá se v této třídě příslušná metoda a dojde k posunutí všech objektů a následnému překreslení. Metoda OnMouseDown() je volána když uživatel stiskne tlačítko myši, informace o stisknutí tlačítka myši se uloží. V metodě onMouseMove(), která se zavolá, pokud uživatel pohne myší, se zjistí, zda je stisknuté tlačítko myši. Pokud ano, spočítá se vzdálenost, o jakou se kurzor posunul a adekvátně se změní pozice všech načtených dat z mapy okrajů. Metoda OnMouseUp() se pak volá, když uživatel pustí tlačítko myši a tlačítko již dále není stisknuté.

### TFS_Reader.TFSReaderInterface

Abstraktní třída TFSReaderInterface se chová jako interface. Důvod, proč se jedná o abstraktní třídu je, že v interface není možné definovat třídu s modifikátorem přístupnosti private nebo protected. Pokud by se v budoucnu změnil formát vstupních dat, je aplikace snadno rozšiřitelná, jelikož stačí vytvořit novou třídu dědící z TFSReaderInterface, která bude implementovat metodu readImage(), obsahující novou načítací logiku.

### TFS_Reader.TFSReader

Třída TFSReader se stará o načtení dat ze souboru TFS. Umožní specifikovat cestu k souboru, a poté, pokud se soubor podaří načíst, získáme libovolný ze dvou načtených obrázků, které můžeme vykreslit. Konstruktor bere jako argument cestu k souboru, který chceme načíst. Metoda readFirstImage() přečte ze souboru první obrázek. Vrací nám hodnotu udávající, zda se načtení povedlo. Výsledný obrázek lze pak získat z instance objektu této třídy, a to zavoláním metody getImage(). Metoda readSecondImage() přečte ze souboru druhý obrázek a jeho následné získání je analogické. Obě metody volají metodu readImage() pro vlastní načtení dat. Metoda readImage() pak jako argument bere číslo reprezentující obrázek, který chceme získat (1-první, 2-druhý). Tato metoda má modifikátor přístupnosti private, a provádí vlastní načtení dat. Čte data ze souboru a z těchto dat tvoří výsledný obrázek.

### ReferencePointSolver.BorderWithConsideredPoints

Třída BorderWithConsideredPoints slouží k vypočtení bodů, které se nachází uvnitř mapy okrajů. Používá BorderLoader k načtení borderu. Po úspěšném načtení má za úkol vypočítat body, které jsou uvnitř mapy okrajů. Metoda load() bere jako argument cestu k souboru BER. Po úspěšném načtení se do paměti uloží výsledný obrázek mapy okrajů oblasti. Metoda runBFS() spustí algoritmus BFS, pomocí kterého se body uvnitř mapy okrajů vypočítají.

### ReferencePointSolver.ImageRefPointCalculationData

Tato třída funguje jako pomocná třída k uchování informací o souboru TFS během výpočtu referenčního bodu. Uchovává se část binárních dat obrázku a celkový součet rozdílů referenčního bodu a všech ostatních bodů ve snímku fázové mapy.

### ReferencePointSolver.ReferencePointSolver

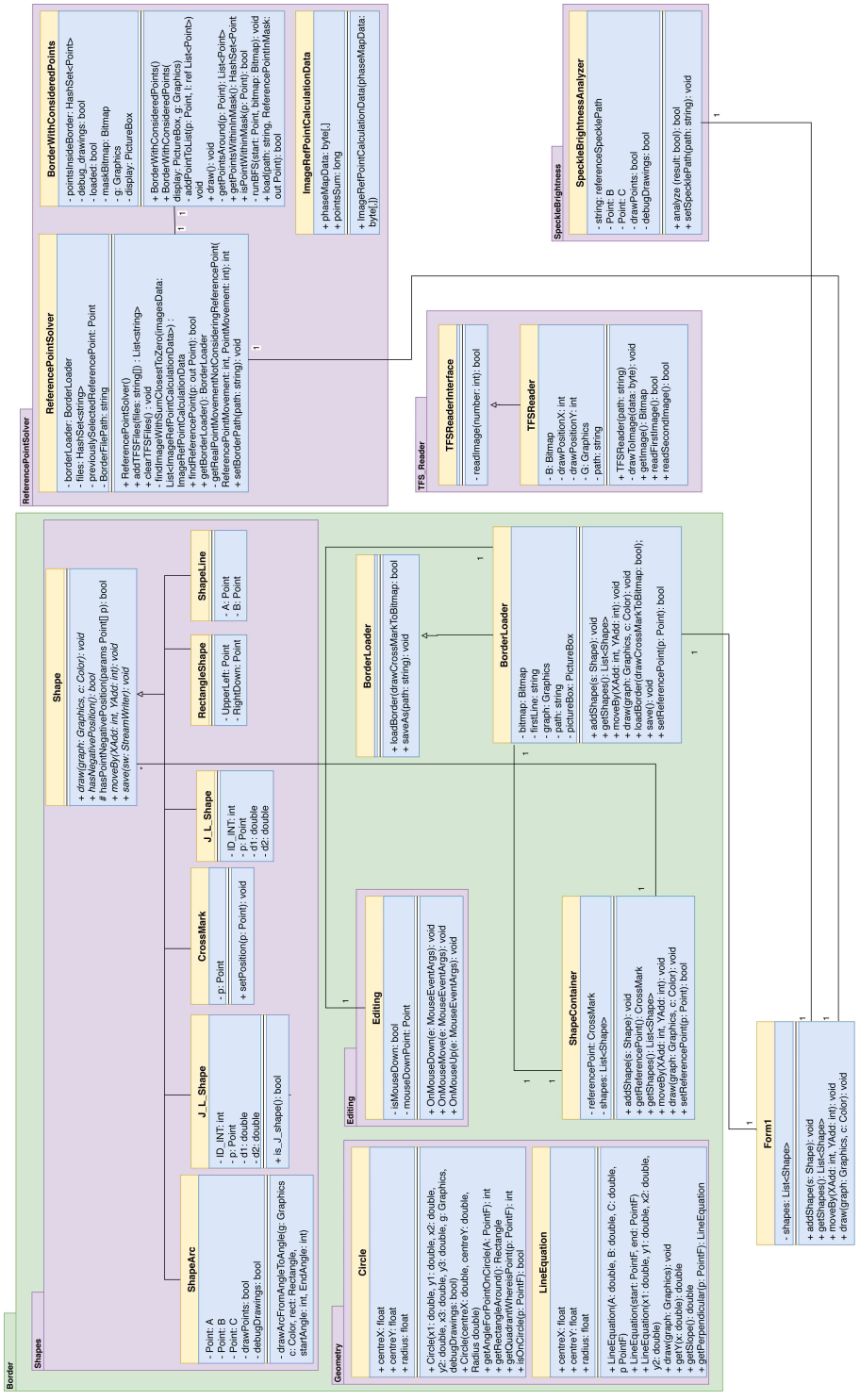
Tato třída se stará o vlastní výpočet optimálního referenčního bodu. Pomocí metody addTFSFiles() lze přidat soubory TFS, které budou v algoritmu používány. Metoda setBorderPath() slouží k specifikování cesty k souboru BER. Metoda findReferencePoint() provádí vlastní výpočet a ve výstupním parametru Metoda findImageWithSumClosestToZero() bere jako argument list struktur ImageRefPointCalculationData s daty pro jednotlivé TFS soubory a vrátí strukturu, která obsahuje nejmenší součet rozdílů referenčního bodu a všech ostatních bodů v snímku fázové mapy.

### SpeckleBrightness.SpeckleBrightnessAnalyzer

Tato třída se stará o výpočet, který určí, zda jsou snímky v souboru TFS použitelné na základě jasu. Cesta k souboru TFS se předává pomocí metody setSpecklePath(). Metoda analyze() pak implementuje vlastní algoritmus. Návratová hodnota určuje, zda se podařilo soubor načíst a algoritmus proběhl. Ve výstupním argumentu metody je pak vrácena hodnota testu, tedy zda je, či není soubor použitelný.

Příloha C

# Obrazová příloha




Obrázek C.1: Diagram tříd

Příloha D

# Instalační a uživatelská příručka

## D.1 Minimální požadavky

Aplikace je psaná pro operační systém Windows. Abychom ji spustili, potřebujeme .NET Framework. Ne vždy je ale potřeba knihovny instalovat. Na čerstvě nainstalovaném operačním systému Windows 7 byly knihovny již součástí celé instalace systému, tudíž aplikace šla rovnou spustit. Aktuální verzi lze stáhnout zde: `https://www.microsoft.com/net/download/dotnet-framework-runtime`.

## D.2 Práce s mapou okrajů sledované oblasti

Zde následuje popis možných operací a práce s mapou okrajů sledované oblasti.

### D.2.1 Načtení mapy okrajů sledované oblasti

Soubor s mapou okrajů oblasti (dále jen BER soubor) lze načíst dvěma způsoby. Buď lze soubor přetáhnout na okno aplikace, čímž se načte, a nebo užitím nabídky Open → Open Border. Pro přetáhnutí na okno aplikace je potřeba vybrat nahoře záložku „Border Editor“.

V této záložce se také data zobrazí.

V praxi je pro posun potřeba mít nějaký snímek pozadí. Pro tyto účely lze načíst i TFS soubor obsahující fotografii pozadí laboratoře (či vyfoceného nástroje), vůči kterému se může uživatel při posunování orientovat. Načtení tohoto souboru probíhá stejně jako načtení souboru s fázovou mapou (viz dále). Výsledné pozadí bude zobrazeno pod načtenou maskou.

### D.2.2 Posun mapy okrajů sledované oblasti

Uživatel vybere nahoře záložku „Border Editor“. Předpokládá se, že je již úspěšně načtený soubor s mapou okrajů oblasti. Mapu lze posunout, pokud

uživatel klikne levým tlačítem myši a podrží ho stisknuté. Následně posune kurzor na cílové místo a tlačítko myši pustí. Pro detailnější posuny lze užít čtyř tlačítek v dolní části aplikace. Každé tlačítko znázorňuje směr posunu (nahoru, dolu, doleva, doprava), a posun se po kliknutí na tlačítko děje vždy o jeden pixel v požadovaném směru.

### D.2.3 Uložení mapy okrajů sledované oblasti

Uživatel vybere nahoře záložku „Border Editor“. Předpokládá se, že je již úspěšně načtený soubor s mapou okrajů oblasti. Mapu lze uložit vybráním příslušné volby z horního menu, tedy Save → Save Border. Pokud má některý z bodů mapy zápornou hodnotu, aplikace zobrazí hlášku, že border nelze uložit a k uložení nedojde. Uživatel musí adekvátně pozměnit polohu borderu, aby byla validní.

## D.3 Práce se souborem s fázovou mapou

Zde následuje popis možných operací a práce se souborem s fázovou mapou.

### D.3.1 Načtení souboru s fázovou mapou

Soubor s fázovou mapou okrajů (dále jen TFS soubor) lze načíst analogicky jako BER soubor. Také lze soubor přetáhnout na okno aplikace, čímž se načte, a nebo užitím nabídky Open → Open TFS. Pro přetáhnutí na okno aplikace je potřeba vybrat nahoře záložku „Border Editor“, kde se pak zobrazí první obrázek v TFS souboru, tedy fázová mapa.

### D.3.2 Vyloučení nepoužitelných snímků na základě jasu

Uživatel vybere nahoře záložku „Speckle Brighness“.

V pravé dolní části klikne na tlačítko s třemi tečkami, označující výběr souboru. Pak vybere soubor, který má být analyzován. Po načtění je graficky zobrazen první snímek obsažený v souboru (tedy fázová mapa). Následně uživatel klikne na tlačítko „Analyze“. Tím se spustí výpočet, po jehož ukončení se zobrazí uživateli oznámení, zda jsou snímky použitelné, či nikoliv.

### D.3.3 Výpočet optimální polohy referenčního bodu

Uživatel vybere nahoře záložku „Reference Point“. Klikne na tlačítko „Add Files“, a vybere několik TFS souborů. Pak je potřeba vybrat soubor s borderem. Po kliknutí na tlačítko s třemi tečkami se zobrazí dialog umožňující výběr BER souboru. Následně může zvolit jméno pro výsledný BER soubor, kam se uloží výsledná poloha referenčního bodu. Po stisknutí tlačítka „Find point“ se spustí výpočet. Nakonec se zobrazí hláška oznamující, že výsledný

bod byl vypočten a dále se uloží mapa okrajů oblasti do zvoleného souboru. V tomto souboru je uložena i vypočtená poloha referenčního bodu. Poloha bodu je navíc také zobrazena v okně. Aplikace očekává validní vstupní data, která se vztahují k testům prováděným na fakultě HAMU.

Příloha E

# Zobrazení dokumentace vygenerované DOXYGENEM

Pro zobrazení dokumentace stačí otevřít soubor „index.html“, který je na USB mediu ve složce src\doc, ve webovém prohlížeči.

Příloha F

# Obsah přiloženého usb

- `readme.txt` ..... stručný popis obsahu usb
- `exe` ..... adresář se spustitelnou formou implementace
- `data` ..... adresář obsahující vstupní data
  - `border editing` ..... adresář vstupní data pro editaci borderu
    - `pozadí.TFS` .... soubor s pozadím, na kterém se dá posouvat mapa okrajů sledované oblasti
    - `spodlen prodej 20 12 16.BER` . soubor s mapou okrajů sledované oblasti
    - `spodlen suk 10 1 17.BER` ..... soubor s mapou okrajů sledované oblasti
  - `reference point data` .. adresář se vstupními soubory pro vypočtení polohy referenčního bodu
    - `*.TFS` ..... vstupní TFS soubory
    - `spodlen prodej 20 12 16.BER` ..... vstupní BER soubor
  - `speckle brightness` ..... adresář se vstupními soubory pro vyloučení jasově nepoužitelných snímků
    - `not valid` ..... adresář obsahující nevalidní snímek
      - `_0049_00208.TFS` ..... nevalidní TFS snímek
    - `valid` ..... adresář obsahující validní snímek
      - `usable.TFS` ..... validní TFS snímek
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `diagrams` ..... .....složka s UML diagramy vytvořených pomocí www.draw.io
  - `doc` ..... dokumentace zdrojových kódů vygenerovaná pomocí aplikace DOXYGEN
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF