České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra kybernetiky

Bakalářská práce

Aplikace pro vyhodnocování experimentu prováděného v aerodynamickém tunelu

Štěpán Riss

Květen 2015

Vedoucí práce: Ing. Pavel Strnad, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Štěpán R i s s

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Aplikace pro vyhodnocování experimentu prováděného v aerodynamickém tunelu

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci sloužící k vyhodnocování dat z fyzikálního experimentu prováděného v aerodynamickém tunelu. Aplikace bude poskytovat poloautomatické vyhodnocování snímků. Celkově se jedná cca o stovky snímků, které jsou vyhodnocovány. Aplikaci navrhněte tak, aby byla co nejvíce přátelská pro uživatele a co nejvíce usnadňovala práci s nasbíranými snímky. Aplikace je vytvářena pro účely pracoviště Institutu termomechaniky AV ČR. Představa zadavatele je taková, že poloautomatické vyhodnocování bude probíhat pomocí průvodce, který bude obsahovat následující kroky:

- kalibrace,
- definice profilu,
- parametry měření,
- vyhodnocování.

Výstupem celého procesu budou výsledná data (tlak, rychlost, hustota, náklon, stoupání) ve vhodném formátu pro další zpracování. Veškeré implementované funkce v práci důkladně otestujte a demonstrujte na vhodných příkladech.

Seznam odborné literatury:

[1] Arlow J., a Neustadt I.: UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky, Brno: Computer Press, 2007, 567 s. ISBN 9788025115039.
[2] Vlček V., Kozánek J.: Preliminary interferometry measurements of flow field around a fluttering NACA0015 profile, Svratka (CZ) 2011, Engineering Mechanics. ISBN 978-80-87012-26-0

Vedoucí bakalářské práce: Ing. Pavel Strnad, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 2. 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. 5. 2015

Poděkování

Mé díky patří Ing. Pavlu Strnadovi Ph.D. za vedení této práce a užitečné podněty a rady při konzultacích.
Déle je třeba říci, že tato práce vznikla za podpory grantu GAČR 13-10527S Analýza subsonického flutteru elasticky uložených profilu s využitím interferometrie a CFD. V neposlední řadě bych rád poděkoval Ing. Petru Šidlofovi Ph.D. za efektivní a příjemnou spolupráci při vývoji aplikace.
Nerad bych zapomněl i na celou rodinu, které bych také vyjádřil svůj vděk za podporu při tvorbě této práce.

Abstrakt

Cílem této bakalářské práce je popis celého vývoje aplikace, která má sloužit k vyhodnocování fyzikálního pokusu. Tento fyzikální experiment probíhá na Akademii věd ČR pod názvem Analýza subsonického flutteru elasticky uložených profilů s využitím interferometrie a CFD. Při tvorbě aplikace je třeba projít celým vývojovým procesem počínaje analýzou požadavků, přes návrh datové struktury a implementace, až k finálnímu testování.

Abstract

The goal of this bachelor thesis is to describe complete development process of application, which shall evaluate physical experiment. This experiment is run on The Czech Academy of Sciences under name of Subsonic flutter analysis of elastically supported airfoils using interferometry and CFD. Complete development of software includes analysis of requirements, design of data structures, implementation and testing.

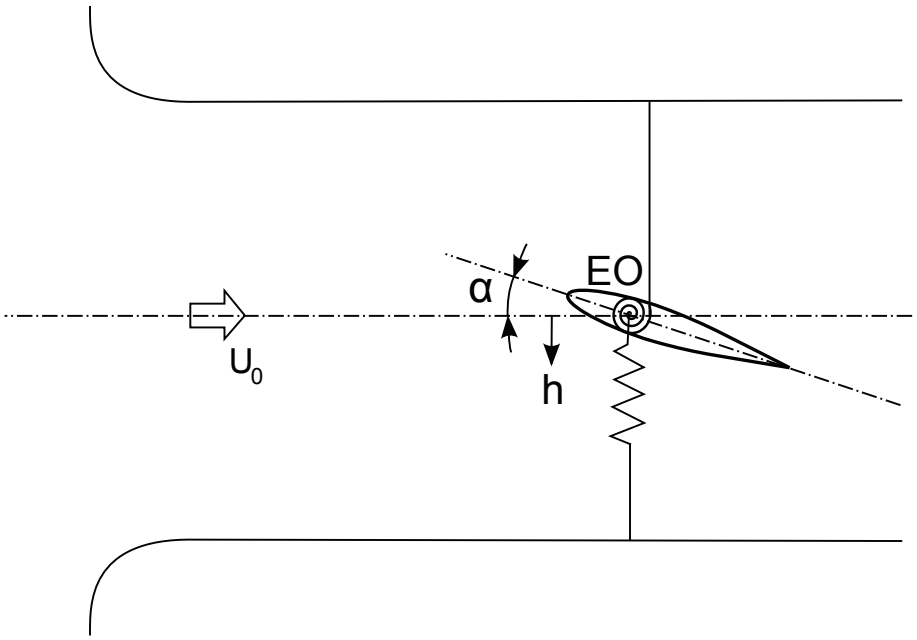
Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1
Úvod

Začátkem roku 2013 na Ústavu termomechaniky AV ČR začal výzkumný projekt ve spolupráci s Technickou univerzitou Liberec. Jméno projektu je Analýza subsonického flutteru elasticky uložených profilů s využitím interferometrie a CFD (Computational fluid dynamics). Tento projekt zkoumá základní nestabilitu typu flutter. Dále se zabývá měřením rychlostí a tlakových polí v blízkosti samo-buzeného modelu křídla. Jeho cílem je vývoj nové metodiky vyhodnocení proudových polí z interferogramů a následné porovnání naměřených dat s výsledky numerických simulací. Experimenty se provádějí v aerodynamickém tunelu, který je znázorněn na obr. 1. Profil NACA0015[1] má v tunelu dva stupně volnosti. První stupeň volnosti je kolem horizontální osy rámu a druhý

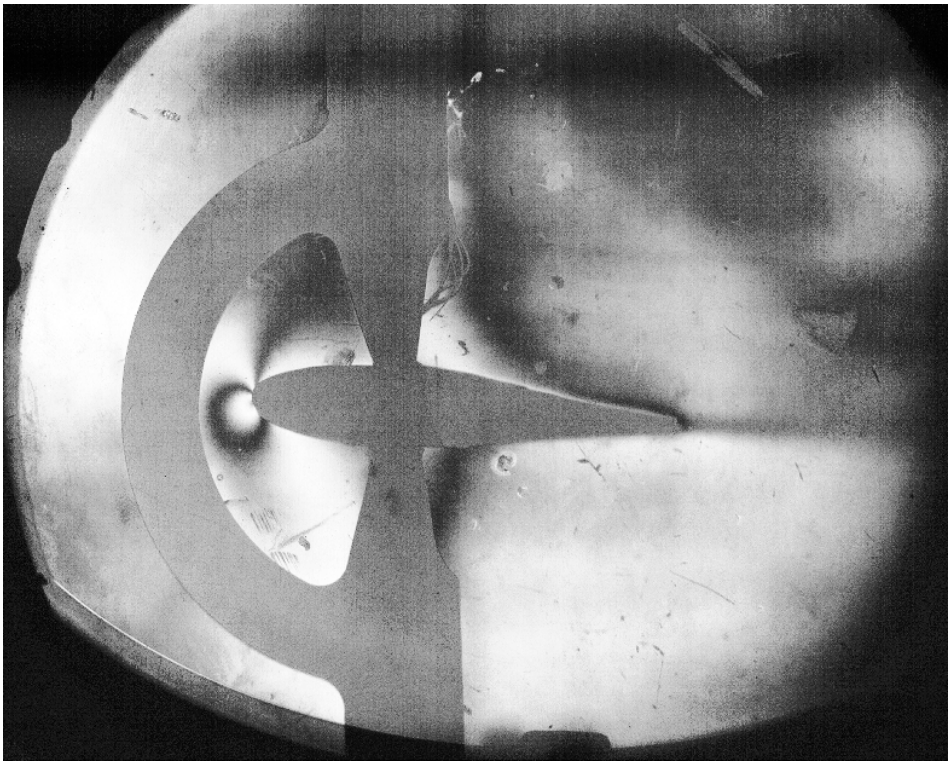


Obrázek 1: Aerodynamický tunel

stupeň je posuv ve vertikálním směru. Tuhost úhlového pohybu α závisí na výběru torzní pružiny a tuhost vertikálního pohybu h na plochých ocelových pružinách. Ve směru U_0 proudí vzduch jehož rychlost se pohybuje kolem 0.4 Ma. Měření trvá v řádech vteřin a pomocí rychlostní kamery a interferometru se pořizují série snímků. Na těchto snímcích je pak možné pozorovat změny hustot vzduchu kolem profilu. Příklad snímku je na obr. 2. Ze snímků se dají pomocí analýzy vypočítat údaje jako jsou rychlost, hus-

[1]Přesně definovaný tvar křídla vyvinutý úřadem National Advisory Committee for Aeronautics

tota a tlak vzduchu v určitých bodech na profilu. Následně i moment kolem elastické osy profilu a celková síla, která působí na profil v horizontálním a vertikálním směru.



Obrázek 2: Interferogram

K tomuto vyhodnocování by měl sloužit software, k jehož vývoji jsem měl tu příležitost se dostat. V následujících kapitolách se budu zabývat celým postupem při vývoji této aplikace. Od sepsání a analýzy požadavků, přes návrh datové struktury až po samotnou implementaci a testování.

Kapitola 2
Seznam požadavků

Po několika schůzkách se zadavateli programu, jsem byl schopen zformulovat konkrétní požadavky. Tyto požadavky odpovídají stanovenému zadaní a jsou rozděleny na funkční a obecné[2]. Požadavek chápeme tak, že definovat funkci, kterou musí systém provést, nebo podmínky výkonu, který musí systém dosáhnout. Dále musí být proveditelný a měřitelný.

2.1 Funkční požadavky

Funkční požadavky identifikují nutné akce a funkce, které musí být aplikací splněny.

2.1.1 Příprava projektu

FP1 Založení nového projektu
Při zakládání nového projektu nejdříve proběhne výběr složky, ve které se snímky na vyhodnocování nacházení. Následně uživatel musí projekt pojmenovat a program hlídá, aby nedošlo k přejmenování již existujícího projektu.

FP2 Kalibrace snímku
Na vybraném snímku uživatel označí dva body, u kterých zná vzájemnou vzdálenost a tím definuje přepočet pixelů na milimetry.

FP3 Zadání ideální a reálné délky profilu
Ideální délku profilu zadává uživatel v milimetrech a reálnou délku udává v procentech ideální délky.

FP4 Nastavení equalizace histogramu a gamma korekce
Pro případ špatně kontrastních snímku možnost vyvážení histogramu a nastavení gamma korekce.

FP5 Nastavení parametrů měření
Prostor pro zadávání všech důležitých parametrů pro měření (fyzikální konstanty pro počítání výsledků, základní umístění elastické osy, čas prvního snímku a frame rate kamery, . . .).

FP6 Přidání fyzických bodů fixovaných na objekt nebo snímek do měření
Zvýraznění některých důležitých bodů ve snímku. Tyto body jsou buď pevně

[2]Z anglického non-functional requirements často v literatuře překládané jako nefunkční požadavky. Jedná se ovšem o velmi nešťastný překlad, proto budu v této práci používat označení obecné požadavky.

přichycené na snímek, nebo si budou vždy držet konstantní vzdálenost od profilu. Elastická osa profilu je jediný fyzický bod, který musí uživatel povinně definovat.

FP7 Každému snímku označit proudy vzduchu fringe indexem
Před samotným vyhodnocováním měření můžou všechny snímky projít hrubou přípravou od kvalifikovaného odborníka. Ten projde všechny snímky a k proudům vzduchu přiřadí fringe indexy. Uživatel, který bude posléze měření vyhodnocovat, bude tato označení používat jako pomůcku.

2.1.2 Práce s projektem

FP8 Prohlížení snímků měření
Jedná se o přepínání mezi vyhodnocovanými snímky s možností sekvenčního přehrávání.

FP9 Ukládat a načítat rozpracovaný projekt
Při uložení se soubor, který uchovává rozdělanou práci, uloží automaticky do adresáře s vyhodnocovanými snímky. Uložení projektu je vždy vázáno právě k těmto snímkům, proto uživatel nemá možnost projekt uložit jinam.

FP10 Vyhodnocování snímků kvalifikovanou obsluhou
Lokace profilu probíhá zadáním předního a zadního bodu profilu (poté se profil automaticky dokreslí). Při identifikaci proužků musí kvalifikovaný uživatel nalézt ve snímku dané místo na profilu, které označí tzv. proužkem. Proužek označí indexem, podle kterého se vypočítají požadované fyzikální hodnoty v tomto bodě.

FP11 Exportování vyhodnocených dat v definovaném formátu Po vyhodnocení snímků má uživatel možnost výsledná data vyexportovat. Místo, kam se data mají exportovat uživatel zadá před každým exportem zvlášť. Formát exportovaných dat bude definován níže.

2.2 Obecné požadavky

Obecný požadavek definujeme jako omezující způsob, jímž bude systém implementován.

OP1 Funkční na platformě Windows
Aplikace bude funkční na operačních systémech Windows XP a vyšší verze.

OP2 Zadané uživatelské prostředí
Uživatelské rozhraní má být inspirované programem *i-Speed Control Software*.

OP3 Nastavování prostředí programu
Možnost nastavení vlastností grafických objektu (viditelnost, barva, tloušťka čar). Dále počet desetinných míst, na které se má při exportu zaokrouhlovat.

OP4 Aplikace ve formátu průvodce bez možnosti zpětné modifikace parametrů
Program se chová jako průvodce, tedy uživatel je veden a postupně musí vyplňovat požadované parametry. V momentě, kdy uživatel potvrdí zadané parametry, ztrácí možnost jejich úpravy.

OP5 Program se má jmenovat *IFGPro*
IFG je zkratka pro interferometrické snímky a *Pro*, protože je program nástupcem předchozí jednodušší verze.

Kapitola 3
Analýza požadavků

V této část práce se budu věnovat návrhu a analýze případu užití. Nejdříve popíši případ užití pro celou aplikaci a pak se kvůli **OP4** budu věnovat případu užití pro jednotlivé kroky.

Všechny obrázky případů užití jsem kreslil v programu *Visual Studio*[3].

3.1 Aktéři

V této aplikaci se vykytuje pouze jeden aktér. Jedné se o uživatele, který bude celý program obsluhovat. Podle **FP10** se musí jednat o uživatele, který je pro tuto práci kvalifikován.

3.2 Případy užití

PU1 Celá aplikace

1. Uživatel otevře projekt (viz **PU2**)
2. Dále proběhne kalibrace snímku (viz **PU3**)
3. Nastaví vhodný typ profilu a polohy fyzických bodů (viz **PU4**)
4. Vyplní parametry měření (viz **PU5**)
5. Proběhne vyhodnocování potřebných snímků (viz **PU6**)

PU2 Otevření projektu

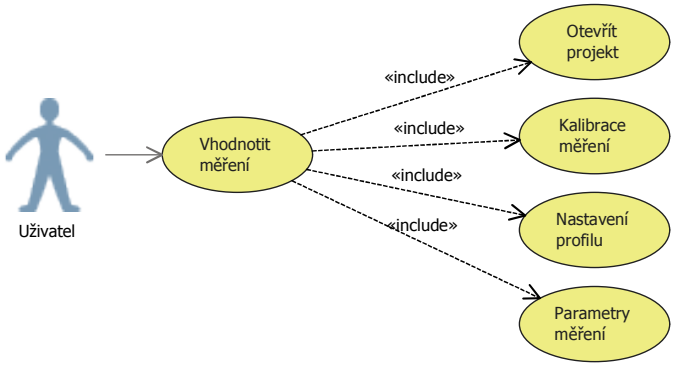
1. Uživatel vybere složku měření
2. Pojmenuje projekt

Tento případ užití má alternativní scénář a to, že uživatel otevře již existující projekt. Od chvíle, kdy je otevřen projekt může uživatel také ve snímcích přiřazovat finge indexy (**FP7**), projekt ukládat (**FP9**) a procházet snímky projektu (**FP8**).

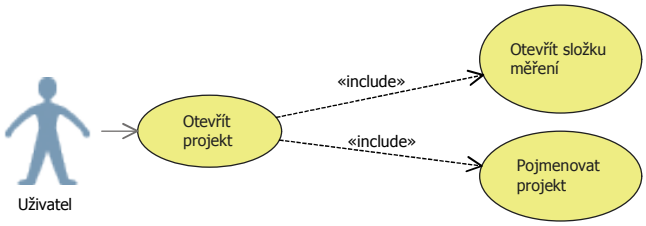
PU3 Kalibrace projektu

1. Uživatel na vybraném snímku definuje dva body
2. Udá reálnou vzdálenost mezi těmito body v milimetrech
3. Nastaví equalizaci histogramu a hodnotu gamma korekce

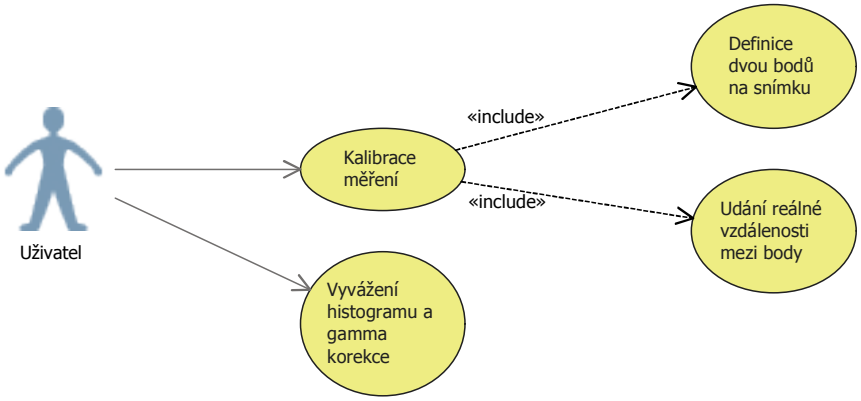
[3] `https://www.visualstudio.com/`



Obrázek 3: Případ užití celé aplikace



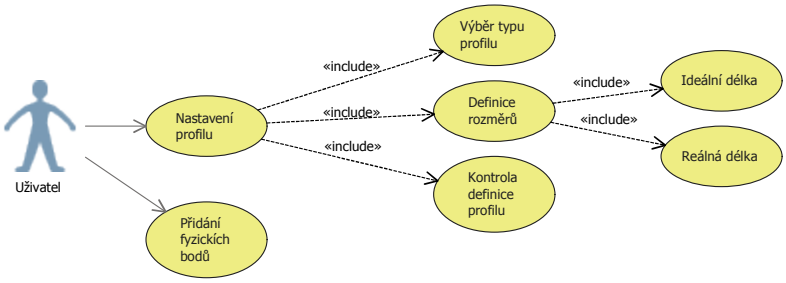
Obrázek 4: Případ užití pro otevření projektu



Obrázek 5: Případ užití pro kalibraci

PU4 Nastavení profilu

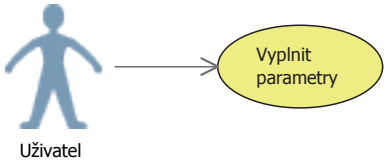
1. Výběr typu profilu, se kterým se v měření pracuje
2. Zadání ideální délky profilu v milimetrech
3. Nastavení reálné délky profilu v procentech ideální délky
4. Porovnání uživatelem definovaného profilu s profilem na snímku
5. Přidání fyzických bodů do měření



Obrázek 6: Případ užití pro nastavení profilu

PU5 Parametry měření

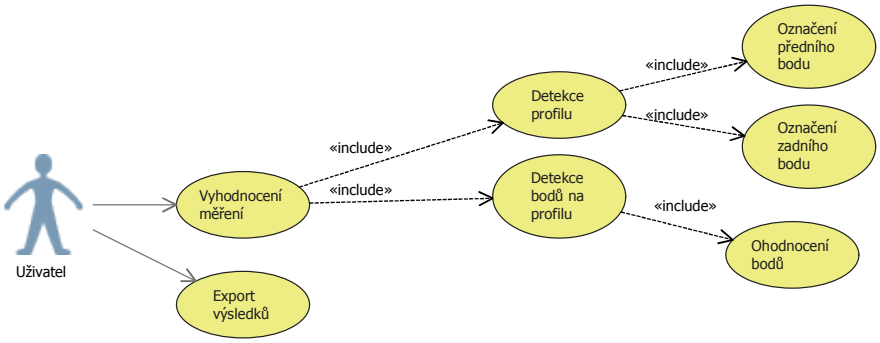
1. Uživatel vyplní požadované parametry měření



Obrázek 7: Případ užití pro nastavení parametrů měření

PU6 Vyhodnocování snímků měření

1. Uživatel na vyhodnocovaném snímku identifikuje profil
2. Označí na profilu důležité body
3. Těmto bodům přiřadí hodnoty pro následný výpočet
4. Vyhodnocený projekt exportuje



Obrázek 8: Případ užití pro vyhodnocování měření

3.3 Sledování požadavků

Abychom si ověřili pokrytí všech funkčních požadavků, použijeme metodu matice sledovanosti požadavků. Tato matice je vidět na tabulce [1]. Je v ní znázorněno, jaký případ užití pokrývá jaký funkční požadavek. Seznam funkčních požadavků je na vodorovné ose a na svislé ose jsou případy užití. Relace mezi požadavkem a případem užití je znázorněna na příslušných souřadnicích.

	Případy užití					
Funkční požadavky		**PU2**	**PU3**	**PU4**	**PU5**	**PU6**
	FP1	X				
	FP2		X			
	FP3			X		
	FP4		X			
	FP5				X	
	FP6			X		
	FP7	X				
	FP8	X				
	FP9	X				
	FP10					X
	FP11					X

Tabulka 1: Matice sledování požadavků

Kapitola 4
Návrh architektury

V této kapitole se budu věnovat popisu navržené architektury programu. K návrhu aplikace použijeme návrhový vzor zvaný MVC (Model-view-controler) a budu tedy popisovat jeho jednotlivé komponenty. Stejně jako pro případy užití jsem pro kreslení diagramů použil program *Visual Studio*.

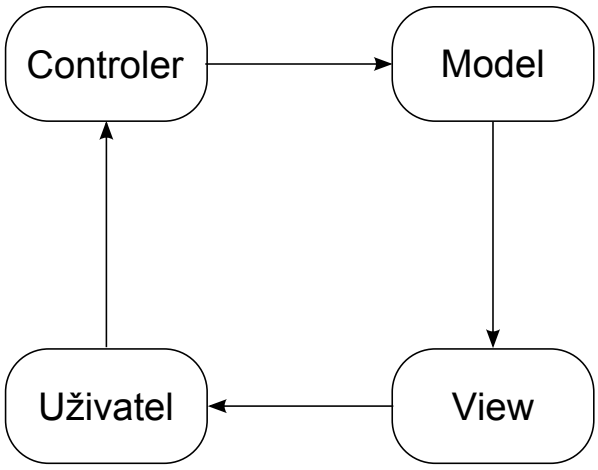
4.1 Model-view-controller

MVC je velmi populární návrhový model, který se používá při implementaci uživatelského rozhraní. Je rozdělen do tří vrstev, které jsou vnitřně vzájemně propojené (viz obr. 9). Velký výhoda tohoto vzoru je přehlednost zápisu, ze kterého vyplývá vyšší rychlost při manipulaci s kódem. Návrhový vzor se sestává z následujících:

Model
: Do této části patří kompletní datové schéma celé aplikace.

Controller
: Controller reaguje na akce uživatele a na základě těchto akcí modifikuje model. Dále se stará o to, aby view zobrazoval správná data.

View
: View slouží k zobrazování dat uživateli.



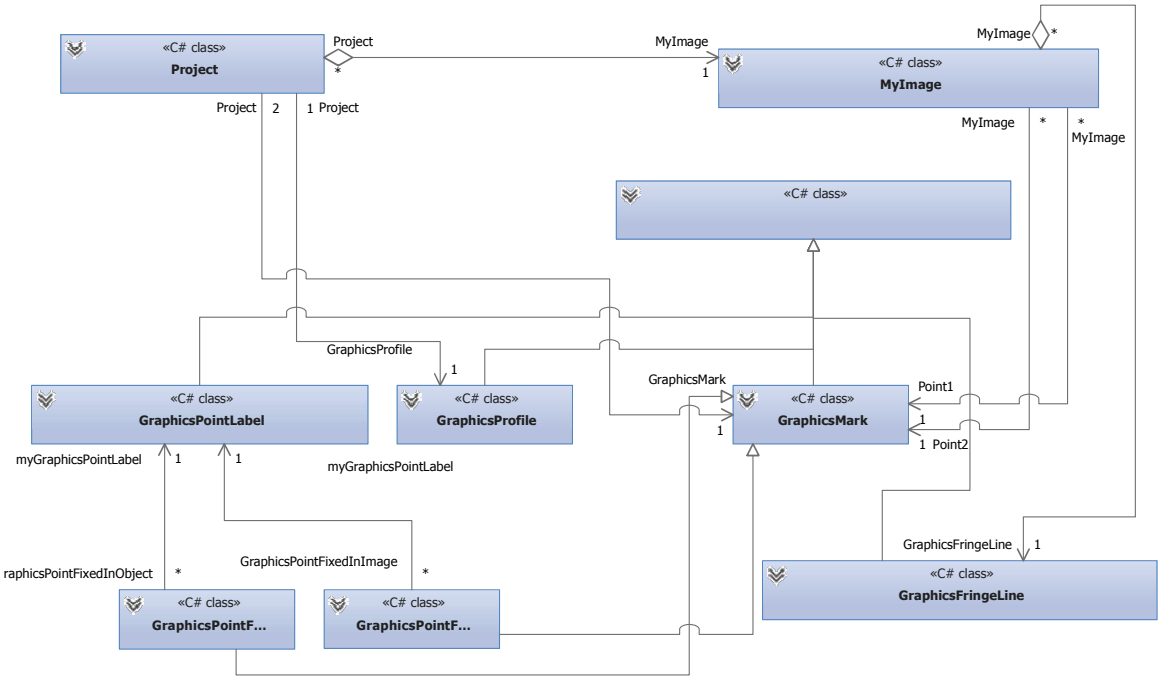
Obrázek 9: MVC model

4.2 Model

Modelem jsou v našem případě všechna data, se kterými pracujeme a která budeme chtít ukládat a načítat. V našem případě bude mít aplikace dva hlavní modely. První je model projektu, který drží všechny informace o daném měření. A druhý je model nastavení aplikace.

4.2.1 Model projektu

V tomto modelu je potřeba vhodně skladovat všechny informace. Z toho důvodu byla navržena struktura tříd, která by měla všem požadavkům vyhovět (viz obr. 10). V textu níže budu všechny navržené třídy a vztahy mezi nimi popisovat.



Obrázek 10: Návrh datového modelu

Projekt
: Tato třída obsahuje základní informace o projektu jako jeho jméno, cestu do složky měření a objekty, které jsou vázané na měření. Dále drží list, který se skládá ze všech snímků měření. Dále má tato třída vlastnosti pro pohyb v tomto listu.

MyImage
: Třída MyImage je v podstatě reprezentace jednoho snímku se všemi jeho vlastnostmi. Mezi jeho základní atributy patří jméno a cesta k obrázku, pozice profilu definovaná dvěma body, list bodů na profilu a vypočtené fyzikální hodnoty pro daný snímek.

GraphicsObjetct

Kvůli tomu, že v aplikace dochází k vykreslování a práci s různými grafickými objekty je vytvořena tato abstraktní třída. U všech grafických objektů totiž vyžadujeme stejné atributy

bool IsSelected Definuje, zda je objekt aktivní. To ovlivňuje například barvu, kterou se má vykreslit a chování při další aktivitou uživatele.

a stejné metody

GraphicsObject IsHit(Point p)
Tato metoda nám vrací objekt, který je zasažen bodem p. Není-li zasažen vrací *null*.

void DrawToGraphics(Graphics e)
Metoda, která zajišťuje vykreslení grafického objektu do grafiky obrázku.

void SetLocation(Point p)
Pro nastavení nové lokace grafického objektu.

Seznam potomků této třídy je následující:

GraphicsMark
Jednoduchá třída pro grafické znázornění bodu pomocí symbolu X. Tato třída je předkem

GraphicsPointFixedInImage
Třída, která zajišťuje data bodu fixovaným na snímek. Má k sobě fixovaný ještě textové označení.

GraphicsPointFixedInObject
Data bodu fixovaný na objekt. Stejně jako 4.2.1 má k sobě ještě textové značení.

GraphicsProfile
Grafický objekt profilu. Je definován polem bodů.

GraphicsPointLabel
Třída, která slouží k vykreslování textu do grafického pole. Může být fixována na 4.2.1.

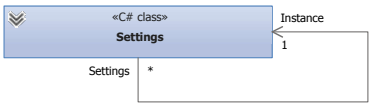
GraphicsFingeLines
Grafické označení bodu na povrchu profilu úsečkou kolmou na profil právě v tomto bodě. Označení tohoto bodu je na druhé straně této úsečky.

Parametry měření

Vzhledem k tomu, že z této třídy budou téměř všechny ostatní třídy číst, je tato třída typu singleton. To znamená, že existuje pouze jediná instance této třídy, která je přístupná ve všech místech aplikace. Tato třída obsahuje všechny parametry projektu (délka profilu, fyzikální konstanty,. . .).

4.2.2 Model nastavení programu

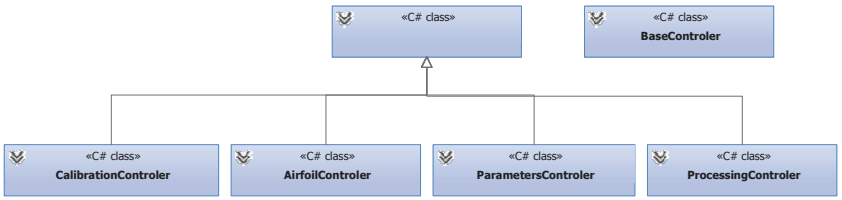
V tomto modelu je pouze jedna třída (viz obr. 11), která obsahuje všechna nastavení aplikace. Tato třída je typu singleton, protože její atributy je potřeba číst téměř ze všech částí kódů.



Obrázek 11: Třída settings

4.3 Controller

Tato část MVC se tedy stará o veškeré řízení a logiku aplikace. Zpracovává vstupy od uživatele a reaguje na ně modifikací modelu. Je tedy jasné, že jeden controller bude potřeba na obecnou práci s programem a druhý, který bude zajišťovat veškeré funkce programu, u kterých se nepracuje s projektem. Kvůli požadavku na program, podle kterého má být práce s programem ve formátu průvodce (viz **OP4**), je tento controller předkem controllerů dalších, které se starají o logiku konkrétních kroků. Diagram tříd controllerů je na obr. 12.



Obrázek 12: Třídy controllers

Controller, který se stará o řízení programu se jmenuje *BaseController* a controller pro řízení projektu se jmenuje *ProjectController*.

4.4 View

Sem patří třídy, které budou zajišťovat uživatelské rozhraní. Vzhledem k tomu, že celý hlavní program se bude odehrávat v jednom okně, stačí jenom jedna hlavní view třída. Patří sem ještě mimo hlavní view *MainWindowView* dialog na otevření nového projektu *OpenNewProjectDialogView*, dialog na nastavování programu *SettingsView* a formuláře na nastavování fyzických bodů *FixedInImage* a *FixedInObject*.

Kapitola 5
Implementace

Tato kapitola má za úkol popsat postup implementace navržené datové struktury a jejich konkrétních funkcí. Tento postup zahrnuje výběr programovacího jazyka, vývojového prostředí a popis samotné implementace.

5.1 Použité technologie

Zde popíši vybrané technologie, pomocí kterých docházelo k realizaci aplikace.

5.1.1 Programovací jazyk

Díky požadavku **OP1** nemusí být naše aplikace multiplatformní. Proto jsem zvolil programovací jazyk C# na .NET frameworku[4] 4.0. C# je čistě objektově orientovaný programovací jazyk vyvinutý společností Microsoft. C# je založen na jazycích C++ a Java. Tento jazyk disponuje vlastnostmi jako jsou například polymorfismus, dědění a automatický garbage collector.

5.1.2 Vývojové prostředí

Při vývoji aplikace v jazyce C# nemá programátor mnoho možností výběru vývojového prostředí. Software s největší podporou pro tento vývoj je Visual Studio. Visual Studio je ze stejné dílny jako samotný jazyk C#, tedy od společnosti Microsoft. U tohoto vývojového prostředí především využíváme možnosti toho, že můžeme všechna okna aplikace navrhnout pomocí grafického rozhraní, které následně generuje samotný kód. Tato funkce šetří velké množství času.

5.1.3 Uživatelské rozhraní

Uživatelské rozhraní je vždy ve formě okna nebo dialogu, to zajišťuje třída *Form*. Tato třída je z knihovny *System.Windows.Forms* a je to grafická reprezentace okna nebo dialogového boxu. Tato třída je použita jako základ všech view. Skládá se ze dvou částečných tříd (v originále *partial class*). V jedné je automaticky generovaný kód, který definuje grafické rozhraní. Tedy všechny komponenty daného okna jako jsou tlačítka, textboxy apod.. Druhá třída obsahuje logiku okna a zároveň jsou zde zachycovány události všech komponent okna. Dalo by se říci, že model MVC má tento přístup návrhu již implementovaný v sobě. Toho my ale nebudeme využívat nebudeme, a logickou část

[4] Jako framework se označuje ucelená sada použitelných knihoven, tříd atd.. Využití frameworku zpravidla usnadňuje a zrychluje vývoj aplikací.

třídy form využijeme k přesměrování zachycených událostí příslušnému controlleru. Další důležitou komponentou v realizaci grafického rozhraní je prvek, který bude zobrazovat obrázek. V základní knihovně *System.Windows.Forms* je komponenta s názvem PictureBox, která umí pouze obrázek zobrazit. Vzhledem k tomu, že budeme s obrázkem pracovat a budeme chtít obrázek přibližovat a pohybovat se v něm co nejvíce intuitivním způsobem, musely by být všechny tyto funkce implementovány. Proto jsem vyhledal knihovnu, která má komponentu, která tyto funkce implementuje. Jedná se o knihovnu *Cyotek.Windows.Forms* ze studia Cyotek [2], která se zabývá tvorbou grafických komponent pro .NET. Komponenta na zobrazování obrázku se nazývá ImageBox. Mimo již výše uvedených funkcí implementuje tento prvek i metody na přepočet bodů vzhledem k úrovni přiblížení a pozici obrázku v ImageBoxu.

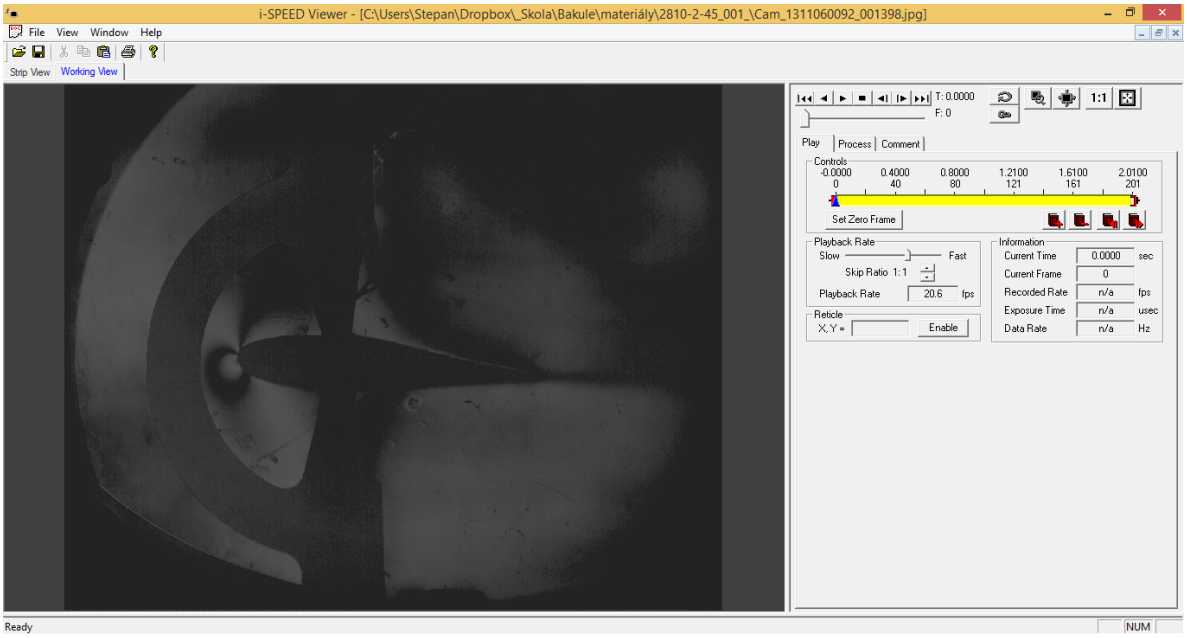
5.1.4 Práce s obrázky

Kvůli požadavku **FP4**, podle kterého musí program vyvažovat histogram a zvládat nastavení gamma korekce, jsem se rozhodl použít grafickou knihovnu Emgu CV [6]. Emgu CV je balíček, který umožňuje jazykům kompatibilním s .NET (např. C#, Visual Basic, IronPython) volat funkce knihovny OpenCV[5]. Mimo to, že díky této knihovně můžeme vyrovnávat histogram a nastavovat gamma korekci, umožňuje rychlé načítání a vykreslování obrázků, díky čemuž můžeme se splňuje i požadavek **FP8**, podle kterého má aplikace být schopná sekvenčně přehrávat snímky. Nevýhoda této knihovny je její velikost. Má cca 1 GB a jenom kvůli těmto pár funkcím by se mohlo zdát zbytečné tak velký nástroj implementovat. Počítá se ale ještě s dalším vývojem, který by mohl další funkce této knihovny využívat.

[5]Vysoce výkonná knihovna na zpracovávání obrázku od společnosti Intel.

5.2 Návrh grafického rozhraní

Z požadavku **OP2** máme zadaný základ vzhledu grafického rozhraní z programu *i-Speed Control Software*[6]. Na obr. 13 je vidět tento software. Na dalším obr. 14 je



Obrázek 13: *i-Speed Control Software*

vidět již výsledná navržená aplikace. Dále bude popis jednotlivých převzatých rysů ze zadaného uživatelského prostředí (očíslování výčtu odpovídá očíslováním jednotlivých částí uživatelského rozhraní na obr. 14):

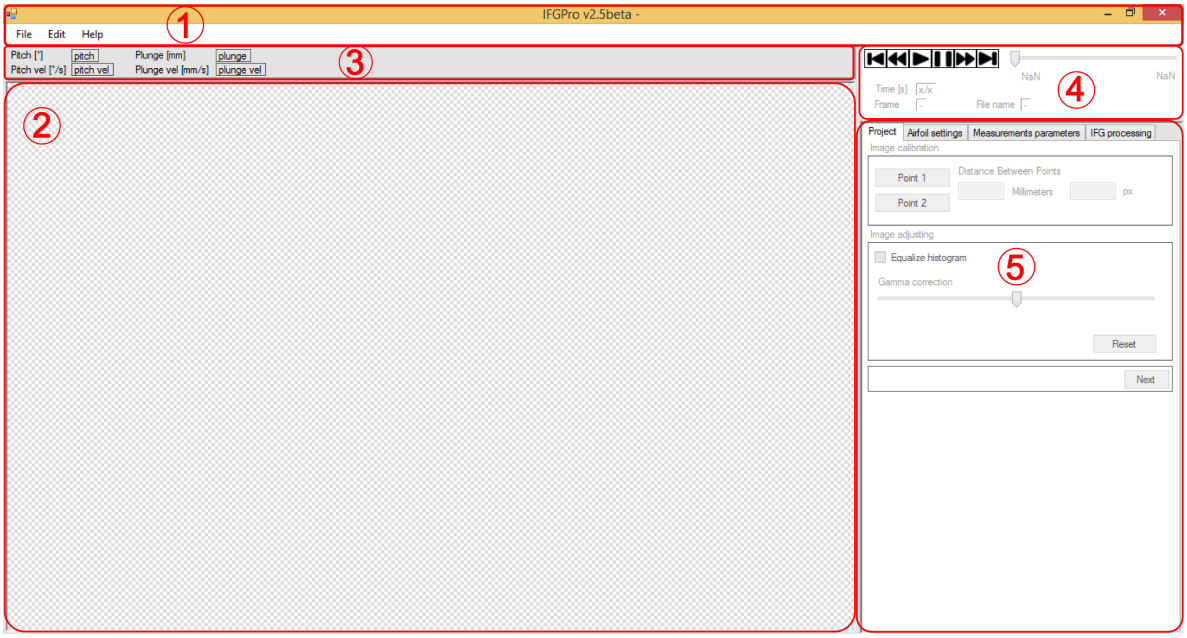
1. Hlavní lišta uživatelského rozhraní má obsahovat jméno programu a jeho verzi. V případě že je programem otevřený projekt, tak i jeho jméno. Lišta dále obsahuje standardní položky jako File, Edit a Help. Pod položkou File jsou možnosti otevření nového projektu, jeho načtení, uložení nebo exportu. Pod další položkou Edit je pouze možnost otevření dialogu pro nastavování aplikace. A pod položkou Help je pouze možnost otevření krátkého popisu aplikace.

2. Tento prostor aplikace slouží pouze k zobrazování snímku měření, se kterým se zrovna pracuje. Dále se do tohoto prostoru vykresluje veškerá grafika.

3. Prostor nad snímkem, ve kterém se zobrazují hodnoty pitch (úhel natočení profilu α viz obr. 1), její rychlost, plunge (hodnota posunu elastické osy oproti referenční poloze elastické osy) a její rychlost. Rychlosti obou hodnot se počítají pouze zpětnou diferencí.

4. V tomto místě se nachází komponenty pro přepínání snímků. Nachází se zde sada tlačítek, pomocí kterých může uživatel přepínat aktuální snímek na snímek

[6] `http://www.ix-cameras.com/ispeed-viewer/`



Obrázek 14: Navržená aplikace

nadcházející nebo předešlý. Dále jsou zde tlačítka, díky kterým může uživatel přeskočit na začátek nebo na konec série snímků, a tlačítka na spuštění a zastavení sekvenčního přehrávání série snímků. Pro přepínání mezi snímky tu je také trackbar. Pomocí jeho jezdce se můžeme pohybovat mezi snímky. Na začátku a na konci je označení času prvního a času posledního snímku v sérii. Také se v tomto místě nachází komponenty na zobrazování informací jako jsou čas daného snímku, kolikátý v pořadí je snímek a jméno snímku.

5. V tomto místě se odehrává největší část práce. Základem je komponenta, která se skládá ze záložek. Každá záložka reprezentuje jeden krok postupu při práci s programem.

5.3 Specifické části kódu

5.3.1 Vlákna aplikace

Celá aplikace běží v podstatě jenom v jednom vlákně. Jsou jen dva případy, kdy se pouští více vláken. První případ je pro otvírání a načítání obrázku do paměti. Při sekvenčním přehrávání snímků je třeba rychle načítat snímky do paměti. Zároveň nechceme aby nám to znemožňovalo práci s grafickým rozhraním. Druhý případ je pro počítání sil působících na profil. Při těchto výpočtech iterujeme celý vrchní i spodní povrch profilu tisíci kroky.

Pro spuštění nového vlákna využívám v aplikace speciální třídu s názvem *BackgroundWorker*. Tato třída slouží ke spuštění operací v novém vlákně a disponuje vlastnostmi

jako jsou možnost kontroly stavu operace vlákna, možnost zrušení operace a možnost spouštění události, v momentě kdy je operace vlákna dokončena.

5.3.2 Export výsledků

Při exportu uživatel vybere složku do které se mají výsledky exportovat. V této složce se poté vytvoří nová složka, které má název
export_jmého_projektu_rok-mesic-den_hodiny_minuty_vetřiny. Tato složka obsahuje:

Textový dokument pojmenovaný jménem projektu. Tento soubor obsahuj výpis všech snímků, na kterých uživatel detekoval profil a informace a vyhodnocená data tohoto snímku.

Textový dokument s názvem *parameters.txt*, který obsahuje výpis všech parametrů, se kterými se v tomto měření pracovalo.

Adresáře s názvem *Lower surface*, resp. *Upper surface*. V těchto adresářích jsou vytvořeny soubory za každý vyhodnocený snímek. Obsahují informace o všech bodech, které uživatel vyhodnotil na spodním, resp. horním povrchu profilu.

Všechny textové soubory obsahují hlavičku, ve které je napsáno o jaká data se jedná. V řádku jsou informace oddělovány tabulátorem.

5.3.3 Ukládání a načítání projektu

Největší otázkou bylo, v jakém formátu projekty ukládat. Požadavkem bylo, aby data byla čitelná, tj. aby po otevření souboru v textovém dokumentu dával zápis smysl. Tento požadavek splňují nejlépe formáty XML[7] a JSON[8]. Nakonec jsem rozhodl pro formát JSON, protože byl při načítání rychlejší než XML a velikostně úspornější.

5.4 Fyzikální výpočty

5.4.1 Hodnoty pro body na profilu

Jak je z výše uvedeného textu zřejmé, nejprve je potřeba identifikovat a ohodnotit finge indexem (tento index musí kvalifikovaná obsluha správně navrhnout) speciální body na profilu. Tyto body značí místa, na kterých proudy vzduchu narážejí do profilu. Podle [3] rychlost a hustota vzduchu v bodě s fringe indexem $i = 0$ odpovídá rychlosti a hustotě vtékajícího proudu vzduchu. Zvyšováním tohoto indexu rychlost a hustota stoupá a snižováním klesá. Celý způsob optického měření hustoty vzduchu je postaven na faktu, že index lomu světla v plynu je lineárně závislý na jeho hustotě. Tento vztah je dán Gladston-Dalovou rovnicí

$$n - 1 = K\,\rho\,, \tag{1}$$

[7] http://en.wikipedia.org/wiki/XML
[8] http://en.wikipedia.org/wiki/JSON

kde K je Gladston-Dalova konstanta, která je určena pro daný plyn. Když máme nastavený interferometr na tzv.*infinite fringe width* (tj. při nulové rychlosti proudu vzduchu je interferometrický snímek plně prosvícen), může být ze vztahu [1] vyvozen rozdíl mezi hustotami $\rho^{(0)}$ a $\rho^{(i)}$, kde bílá barva proudu vzduchu na snímku znázorňuje $\rho^{(0)}$ a černá $\rho^{(i)}$. Tento rozdíl odpovídá

$$\rho^{(0)} - \rho^{(i)} = i\frac{\lambda}{L\,K}\ , \tag{2}$$

kde L je délka testované části a λ je vlnová délka světelného zdroje. Více se tomuto odvozování věnuje odborná literatura [3]. Hodnoty které nás v bodě na profilu tedy zajímají jsou:

Tlak

$$p^{(i)} = p_0 \left(\frac{\rho^{(0)} - i\,\frac{\lambda}{L\,K}}{\rho_0}\right)^{\kappa} \tag{3}$$

K tomu potřebujeme dopočítat

$$\rho_0 = p_0/\left(R\,T_0\right), \tag{4}$$

kde p_0 je celkový tlak vzduchu, který působí na profilu (obvykle atmosferický tlak) a κ měrná tepelná kapacita ideálního plynu.

Machovo číslo

$$M^{(i)} = \sqrt{\frac{2}{\kappa - 1}\left(\left(\frac{p^{(i)}}{p_0}\right)^{\frac{1-\kappa}{\kappa}} - 1\right)}\ , \tag{5}$$

Rychlost vzduchu

$$U^{(i)} = M^{(i)}\sqrt{\kappa\,R\,T^{(i)}} \tag{6}$$

K tomu potřebujeme spočítat ještě lokální statickou teplotu podle vztahu

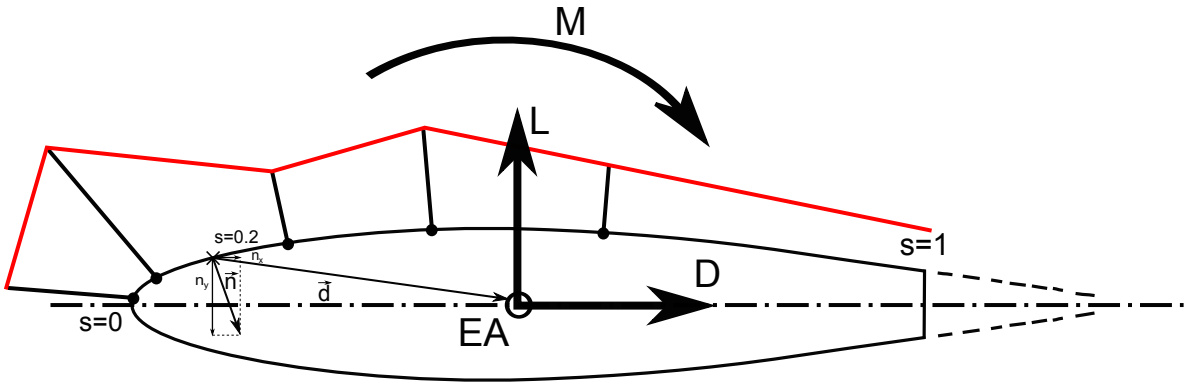
$$T^{(i)} = T_0\left(1 + \frac{\kappa - 1}{2}M^{(i)^2}\right)^{-1}\ , \tag{7}$$

kde T_0 je teplota vzduchu, který při měření proudí zařízením.

5.4.2 Síly působící na profil

V momentě, kdy máme detekovány a platně ohodnoceny alespoň dva body na obou stranách profilu, můžeme vypočítat síly, které působí na profil. Síly jsou znázorněny na obr. 15. Jde nám o lift (L) a drag (D) force a moment (M) kolem elastické osy. Při počítání celkové síly, která působí na profil musíme spočítat zvlášť síly na horním povrch (označován indexem u) a spodní povrch (označován indexem l) profilu. Výsledné síly jsou tedy

$$D = D_u + D_l, L = L_u + L_l, M = M_u + M_l. \tag{8}$$



Obrázek 15: Obrázek k výpočtu sil působící na profil

Síly působící na jednu stranu spočítáme následovně. Díky tomu, že máme spočítané minimálně dva body, ve kterých známe tlak, můžeme tento tlak lineárně interpolovat a vznikne mám funkce tlaku nad povrchem profilu. Interpolace je znázorněna červenou čarou na obr. 15. Z tlaku se již působící síly vyjádří snadno

$$D_{u/l} = l_{s,u/l} w \int_0^{s_{max}} [\tilde{p}(s) - p^{(0)}] \cdot n_{x,u/l}(s) ds, \tag{9}$$

$$L_{u/l} = l_{s,u/l} w \int_0^{s_{max}} [\tilde{p}(s) - p^{(0)}] \cdot n_{y,u/l}(s) ds, \tag{10}$$

$$M_{u/l} = |l_{s,u/l} w \int_0^{s_{max}} (\vec{n}_{u/l}(s) \times \vec{d}_{u/l}(s))(\tilde{p}(s) - p^{(0)}) ds|. \tag{11}$$

Pro náš případ je třeba ještě vzorce zdiskreditovat na následující

$$D_{u/l} \doteq l_{s,u/l} w \Delta s \sum_{i=1}^{N} [\tilde{p}((i - \frac{1}{2})\Delta s) - p^{(0)}] \cdot n_{x,u/l}((i - \frac{1}{2})\Delta s), \tag{12}$$

$$L_{u/l} \doteq l_{s,u/l} w \Delta s \sum_{i=1}^{N} [\tilde{p}((i - \frac{1}{2})\Delta s) - p^{(0)}] \cdot n_{y,u/l}((i - \frac{1}{2})\Delta s), \tag{13}$$

$$M_{u/l} \doteq l_{s,u/l} w \Delta s \sum_{i=1}^{N} |(\vec{n}_{u/l}((i - \frac{1}{2})\Delta s) \times \vec{d}_{u/l}((i - \frac{1}{2})\Delta s))(\tilde{p}((i - \frac{1}{2})\Delta s) - p^{(0)})|. \tag{14}$$

Kde:

$l_{s,u/l}$ - povrchová délka horní, resp. spodní strany profilu

s - povrchová souřadnice, $s \in < 0, 1 >$

s_{max} - povrchová souřadnice konce profilu (v našem případě je to 1)

$\vec{v}$ - jednotkový normálový vektor mířící směrem do profilu, skládá se z $[n_x, n_y]$

$\vec{d}$ - vektor od iterovaného bodu do bodu elastické osy

w - šířka profilu

$p^{(0)}$ - statický tlak bodu s nultým fringe indexem

$\tilde{p}(s)$ - lineárně interpolovaný tlak v bodě povrchové souřadnice s

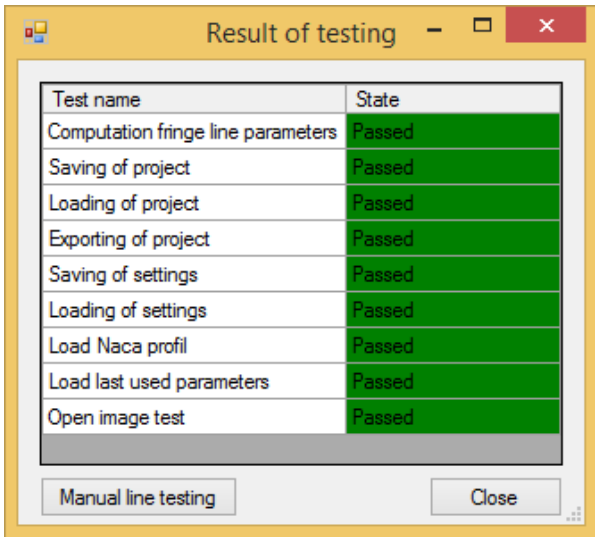
N - počet integračních kroků, v našem případě 1000

Δs - integrační interval, $\Delta s = \frac{s_{max}}{N}$

Kapitola 6
Testování

Testování je velmi důležitá část vývoje aplikace a slouží především k potvrzení správného chování všech funkcí. Testování může být prováděno automaticky nebo manuálně.

Automatické testování, nazývané Unit testing, je způsob testování dílčích částí systému. Spočívá v tom, že víme jaký výstup za daných podmínek má testovaná část systému vrátit a tím automaticky ověřit její funkčnost. V našem případě testujeme správnost počítání fyzikálních hodnot a funkce, které můžeme automaticky vyhodnotit. Test pro kontrolu výpočtů počítá několik hodnot a porovná je s hodnotami referenčními. Pro ostatní automaticky vyhodnocované funkce je vytvořena testovací složka měření (*TestFolder*), ve které je pár snímků měření. Výsledky testování jsou vidět na obr. 16.



Obrázek 16: Navržená aplikace

Ostatní funkce se musí testovat manuálně.

Manuální testování probíhalo postupně při vývoji. V momentě, kdy byli všechny požadované funkce implementovány, aplikace se distribuovala beta testerům, kteří ještě aplikaci testovali a byli použiti jako zpětná vazba. Díky této zpětné vazbě jsem mohl upravit uživatelské rozhraní a některé funkce tak, aby uživatelům program více vyhovoval.

Kapitola 7
Závěr

7.1 Vize do budoucna

S dalším vývojem programu se rozhodně počítá. Nejvyšší prioritu má především automatická detekce profilu ve snímku. Tato práce trvá na samotném vyhodnocování velmi dlouhou dobu a kdyby tuto práci mohl zastávat počítač velmi by to vyhodnocování urychlilo. Díky tomu, že je profil na snímcích z větší části viditelný, neměl by být problém v budoucnu implementovat algoritmus, který by profil detekoval. Počítá se i s potenciální propojeností s programem Matlab pro jeho výpočetní kapacitu. Další přidání funkcí záleží na zadavateli. Momentálně se již hovoří o nové funkci, díky které by si uživatel mohl nechat programem vykreslit grafy výsledků a dále s nimi pracovat.

7.2 Shrnutí

Během tvorby aplikace jsem prošel kompletním vývojem programu a musel jsem řešit problémy, díky kterým jsem nasbíral velké množství užitečných zkušeností. Jak v oboru jednání se zadavatelem, tak v porozumění fyzikálního problému, analýze požadavků, návrhu architektury, implementace, testování a ladění programu. Velká výhoda této práce je, že se s jejím výsledkem aktivně pracuje a neskončí takzvaně v šuplíku. V současnosti vývoj programu stále probíhá. Od uživatelů se sbírá zpětná vazba, na základě které se program upravuje k příjemnějšímu uživatelskému rozhraní.

Literatura

[1] N. I. Arlow J. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2007, ISBN: 978-80-251-1503-9.

[2] Cyotek. Cyotek - imagebox. `http://www.cyotek.com/blog/imagebox-update-version-1-1-0-0`. Naposledy navštíveno 29. 2. 2015.

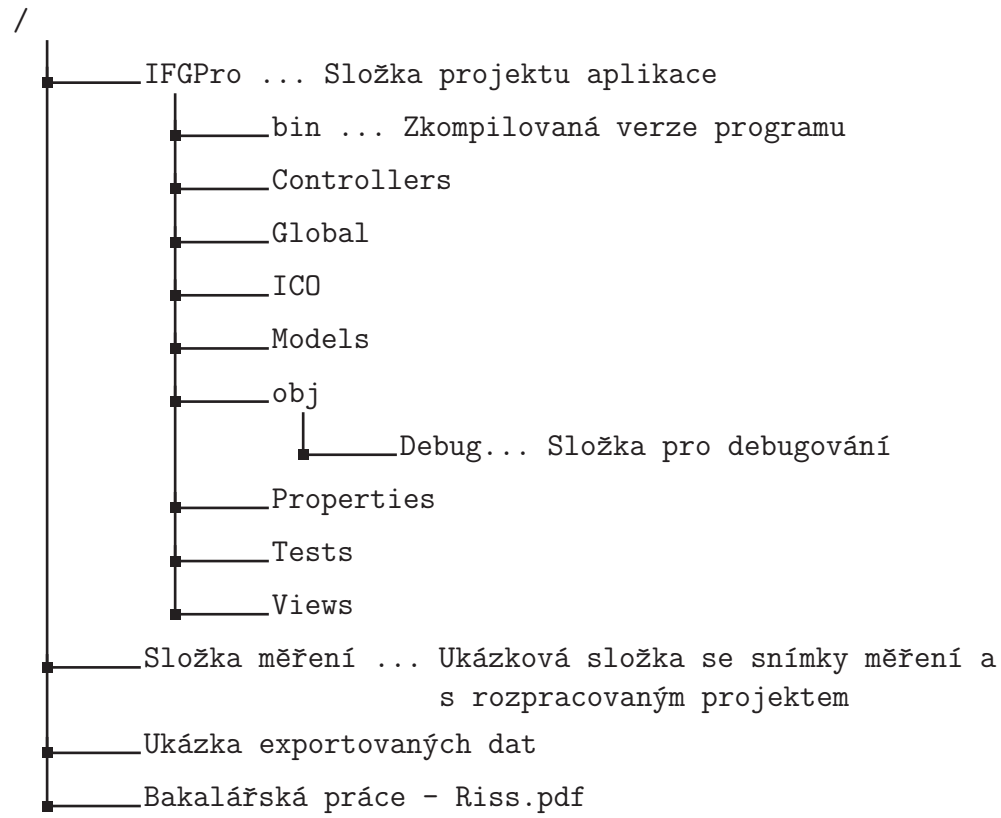
[3] Šidlof P. *Numerical modeling and experimental investigation of flow in domains with moving boundaries*. Habilitation thesis, Technical University of Liberec, 2014.

[4] Microsoft. C# documentation. `https://msdn.microsoft.com/en-us/library`. Naposledy navštíveno 3. 4. 2015.

[5] K. J. Vlček V. Preliminary interferometry measurements of flow field around a fluttering naca0015 profile. *Acta Technica ČSAV*, pages 379–387, 2011, ISSN: 0001-7043.

[6] G. Willow. Emgu cv. `http://www.emgu.com/`. Naposledy navštíveno 8. 3. 2015.

Obsah CD


/
IFGPro ... Složka projektu aplikace
bin ... Zkompilovaná verze programu
Controllers
Global
ICO
Models
obj
Debug... Složka pro debugování
Properties
Tests
Views
Složka měření ... Ukázková složka se snímky měření a
s rozpracovaným projektem
Ukázka exportovaných dat
Bakalářská práce - Riss.pdf