Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

BAKALÁŘSKÁ PRÁCE

Rozhraní pro zpracování záznamu z kamery

Plzeň, 2018

Marek Přitasil

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. 6. 2018 Marek Přitasil,

ABSTRACT

Title: Camera footage processing interface

The topic of this bachelor thesis is the design and development of a video processing user interface used for digital image correlation. The aim is to improve the output visualization of measurements and provide the user with high flexibility of data visualization settings.

The program can load measured data from digital image correlation software. The data can be then further modified and visualized. The program's output is a graphic representation of measured data, for instance in diagrams, modified according to the user's requirements and ready to be published.

ABSTRAKT

Tématem bakalářské práce je návrh a vývoj uživatelského rozhraní pro zpracování záznamu z kamery využívaného k metodě korelace digitálního obrazu. Cílem práce je zkvalitnit vizualizaci výstupu měření a poskytnout uživateli vysokou míru flexibility nastavení podoby vizualizovaných dat.

Vytvořený program umí načíst naměřená data, která jsou výstupem softwaru používaného pro korelaci digitálního obrazu. Data mohou být dále upravována a vizualizována. Výstupem programu je grafická reprezentace naměřených dat, například ve formě digramů, které jsou upraveny dle požadavků uživatele a připraveny k následné publikaci.

OBSAH

1 ÚVOD

Korelace digitálního obrazu (anglicky DIC – Digital Image Correlation) je optická metoda pro určování polohy, posuvů a deformace povrchu objektů způsobených vnějšími silami. Metoda je založena na korelaci referenčního a deformovaného digitálního obrazu sledovaného objektu. V dnešní době se jedná o velmi uznávanou a hojně používanou metodu.

Korelace digitálního obrazu je využívána také pracovníky ZČU na Katedře mechaniky Fakulty aplikovaných věd. Pro své experimenty využívají komerčně dostupný software. Tento software však v některých aspektech, jmenovitě při exportu diagramů, neposkytuje dostatečně široké možnosti nastavení. U Diagramů například nelze měnit popisky os či barvy datových řad.

Cílem této práce je připravit základ softwaru, který bude schopen načíst a vizualizovat data spočtená za pomoci softwaru používaného na Katedře mechaniky FAV (dále uváděn jako „software DIC“). Při vizualizaci dat bude kladen důraz na flexibilitu, zejména při generování diagramů, u kterých by měl mít uživatel k dispozici širokou škálu nastavení.

Obsahem práce je stručný úvod do problematiky korelace digitálního obrazu. Následuje formulace požadavků na software, srovnání a výběr vhodných nástrojů pro vývoj požadovaného softwaru. Druhá část se pak zaměřuje na detaily implementace a dosažené výsledky.

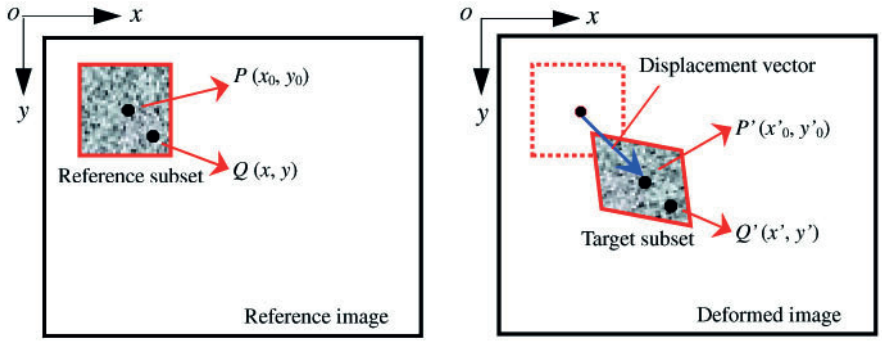
2 KORELACE DIGITÁLNÍHO OBRAZU

Korelace digitálního obrazu je optická bezkontaktní měřící metoda užívaná pro určování deformací a posuvů sledovaného objektu. Metoda je založena na měření změny polohy kontrastního skrvnitého vzoru, který je nanesen na sledovaný objekt. S využitím stereo-kamerových systémů lze určit posuv a deformace ve třech dimenzích. Tato práce se však bude zabývat pouze vizualizací dvoudimenzionální korelace digitálního obrazu.

2.1 PRINCIP VÝPOČTU

Na sledovaný objekt je nejprve nanesen kontrastní skrvnitý nátěr. Tím je na povrchu objektu vytvořena nahodilá struktura, která je nositelem sledované informace. Poté je pořízen snímek objektu v klidovém stavu, tento snímek je označován jako **referenční snímek**. Následně je sledovaný objekt deformován, průběh deformace je snímán vysokorychlostními digitálními kamerami, které pořizují tzv. **deformované snímky**.

Principem korelace je lokalizovat stejné body (pixely) na referenčním a deformovaném snímku. Snímky se pořizují černobílé, každý pixel tak nese informaci o stupni šedi. Protože pořízený snímek má vysoké rozlišení, obsahuje velké množství pixelů stejného stupně šedi, a proto nelze lokalizovat konkrétní pixely. Tento problém je řešen pomocí **subsetů**, což je čtvercové okolí bodu P, pro který je posunutí zjišťováno. Každý subset obsahuje část skrvnitého vzoru, protože je skrvnitý vzor nahodilý, lze subsety jednoznačně identifikovat - dva různé subsety nebudou obsahovat stejný vzor. Náročnost lokalizace jednotlivých subsetů se snižuje s velikostí subsetu, avšak se zvyšující se velikostí subsetu, také roste nepřesnost výpočtu. Z vektorů posunutí je software DIC schopen vypočítat poměrné deformace [1].



Obr. 2.1 – Příklad lokalizace subsertu na deformovaném snímku [21]

3 ANALÝZA

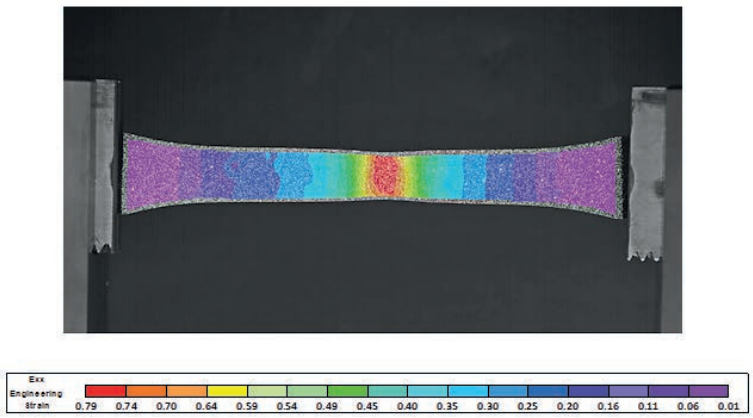
3.1 POŽADAVKY NA SOFTWARE

Na základě diskuze byly stanoveny tři základní požadavky na software, které je bezpodmínečně nutné splnit.

1. Software by měl být spustitelný na různých platformách (Windows, Unix). Software by neměl vyžadovat zakoupení licence dalšího jiného softwaru. Software by mělo být možné v budoucnosti jednoduše rozšířit o další funkce.
2. Software by měl umožňovat načtení zpracovaných dat a umožnit jejich editaci a následnou vizualizaci.
3. Způsob vizualizace bude poskytovat vysokou míru flexibility. Bude například možné měnit rozměry vygenerovaného diagramu, barvy a názvy datových řad v diagramu, apod. Přes sledované snímky půjde zobrazit teplotní mapa s libovolnou průhledností, a další.

3.2 TEPLOTNÍ MAPA

Teplotní mapa (z anglického heatmap) je druh grafické reprezentace dat, kdy je hodnota zastoupena barvou z předem určeného spojitého barevného spektra [20]. Díky převedení číselných dat o viziuální podoby (barvy), je při použití teplotní mapy mnohem jednodušší určit místa, kde sledované veličiny nabývají extrémních hodnot. V případě korelace digitálního obrazu pomáhá k nalezení nejvíce namáhaných míst sledovaného objektu. Tato místa poté mohou být podrobena podrobnějšímu zkoumání.



Obr. 3.1 – Příklad užití teplotní mapy [22]

3.3 ZÁKLADNÍ DRUHY DIAGRAMŮ

Diagram je grafická reprezentace určité informace. V případě vizualizačního programu budou diagramy používány ke znázornění vztahů mezi naměřenými veličinami. V následující části budou popsány typy diagramů, které by šlo využít při vizulizaci dat. Vzhledem k povaze sledovaných veličin se bude tato část věnovat pouze osovým diagramům. Osové diagramy požívají k vizualizaci dat dvou a více os. Typickým příkladem osového diagramu je graf matematické funkce.

Aby bylo možné diagram správně interpretovat, je třeba, aby se z diagramu dalo vyčíst, jaké veličiny zobrazuje, jaké jsou zobrazované hodnoty a v jakých jednotkách jsou zobrazované hodnoty uváděny. K tomuto účelu slouží popis os diagramu či legenda.

3.3.1 Sloupcový diagram

Data jsou vizualizována za pomoci obdélníků, jejichž výška odpovídá naměřené hodnotě sledované veličiny.

3.3.2 Histogram

Histogram je speciálním případem sloupcového diagramu. Data jsou nejprve rozdělena do tříd o stejné šířce. Pro každou třídu je v histogramu zobrazen jeden sloupec, výška sloupce znázorňuje četnost sledované veličiny v daném intervalu. Histogram se používá například pro znázornění distribuce jasu.

3.3.3 Korelační diagram

Korelační (bodový) diagram zobrazuje vztah mezi dvěma proměnnými v kartézské soustavě souřadnic. Jedné dvojici proměnných odpovídá bod v diagramu. Hodnota první proměnné odpovídá poloze na vodorovné ose, hodnota druhé proměnné pak poloze na svislé ose.

3.3.4 Spojnicový diagram

Spojnicový diagram je podobný, jako korelační diagram, jednotlivé body diagramu jsou ale navíc propojeny křivkami.

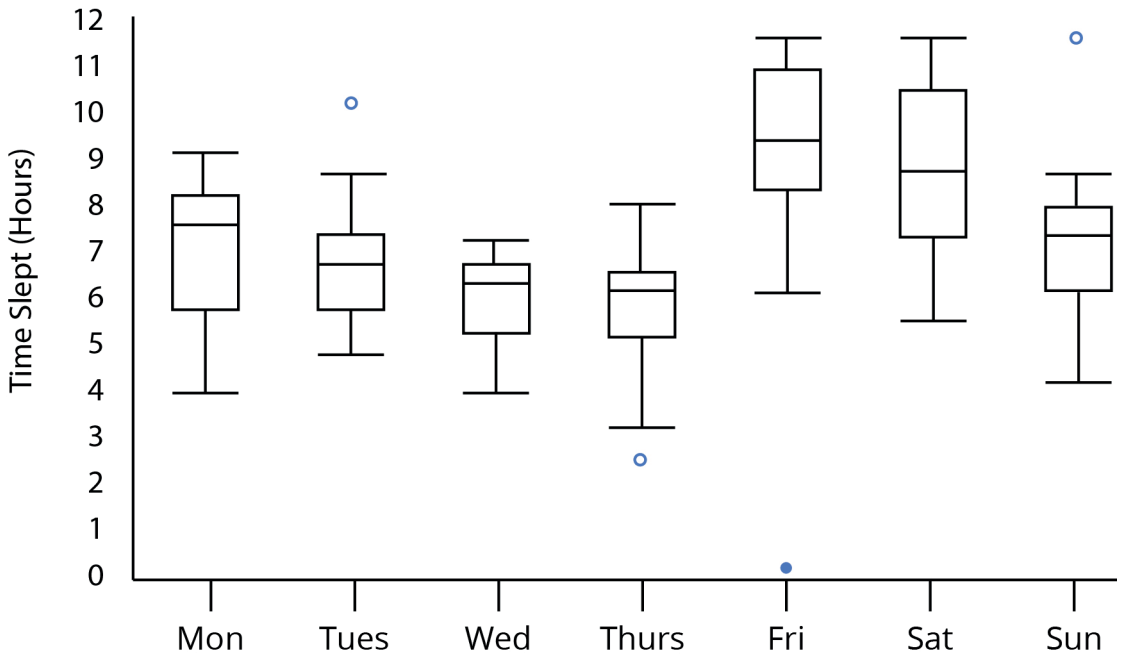
3.3.5 Plošný diagram

Plošný diagram je založen na spojnicovém diagramu. Plocha mezi křivkou a vodorovnou osou má barevnou výplň. Diagram je vhodné použít, pokud je záměrem ukázat trend vývoje jedné nebo více veličin v čase, ale přesné hodnoty těchto veličin nejsou důležité.

3.3.6 Krabicový diagram

Krabicový diagram (anglicky boxplot) je velmi užitečným typem diagramu. Slouží ke znázornění numerických hodnot za pomoci kvartilů. Kvartily jsou speciálním typem kvantilů, které rozdělují statistický soubor na čtvrtiny. Kvartily jsou tedy tři – první, druhý (medián) a třetí (někdy jsou též označovány jako dolní, střední a horní). Pro vzath mezi kvartily a ostatními prvky souboru platí následující: 25 % prvků má mešní hodnotu než první kvartil, 50 % procent prvků má hodnotu menší než druhý kvartil a 75 % prvků má hodnotu menší než třetí kvartil.

Digram je znázorněn jako obdélník (box), jehož spodní hrana odpovídá prvnímu kvartilu a horní hrana třetímu kvartilu. V boxu je pak linií naznačena poloha mediánu. Pomocí tzv. **vousů** ve formě linií vystupujících ze třední části boxu, jsou zobrazeny hodnoty pod prvním a nad třetím kvartilem. Tyto linie mohou být zakončeny znázorněním maximální a minimální hodnoty.



Obr. 3.2 – Krabicový graf [23]

3.4 VÝBĚR PROGRAMOVACÍHO JAZYKA

Programovací jazyky se dle způsobu překladu a spuštění dělí na dvě skupiny – jazyky kompilované a interpretované [19].

3.4.1 Kompilované jazyky

U kompilovaných programovacích jazyků dochází nejprve k překladu zdrojového kódu překladačem (kompilátorem) do strojového kódu platformy. Strojový kód je závislý na architektuře platformy, na které bude výsledný program spuštěn. Tedy, strojový kód přeložený pro platformu A nelze přenést a spustit na platformě B. Namísto toho, musí být zdrojový kód přeložen pro každou platformu zvlášť. Při použití externích knihoven je důležité dávat si pozor na to, aby byla knihovna dostupná a přeložitelná pro všechny platformy, na kterých má být výsledný program spuštěn (tj. aby nepoužívala funkce specifické jen pro jednu určitou platformu). Externí knihovny je obecně třeba překládat stejným překladačem, jako zdrojový kód programu.

Výhodou kompilovaných jazyků je jejich rychlost. Nejvíce časově náročné práce je odvedeno již při překladu programu (typová kontrola apod.). Výsledný strojový kód již stačí jen zavést do paměti a spouštět instrukce. Při reverzním inženýrství je mnohem těžší získat původní zdrojový kód programu.

Mezi nejpoužívanější kompilované programovací jazyky patří C/C++ [4].

3.4.2 Interpretované jazyky

U interpretovaných programovacích jazyků nedochází k překladu zdrojového kódu do strojového. Namísto toho je zdrojový kód spouštěn specializovaným programem, zvaným interpret. Interpret může pracovat několika způsoby.

1. První možností je, že interpret spouští řádku po řádce přímo zdrojový kód programu (např. PHP).
2. Zdrojový kód je nejprve přeložen do tzv. bajtkódu, který je nezávislý na platformě. Interpret za běhu programu nezpracovává zdrojový kód, nýbrž připravený bajtkód .
3. Třetí možností je Just-in-time kompilace (JIT). V tomto případě je bajtkód JIT kompilátorem přeložen do strojového kódu specifického pro danou platformu a poté spuštěn.

Hlavními výhodami interpretovaných jazyků jsou kratší vývojový cyklus (program nemusí být neustále překládán) a přenositelnost (program napsaný na platformě A lze beze změn spustit na platformě B za předpokladu, že pro obě platformy existuje interpret daného jazyka). Nevýhodou je pomalejší běh

programu, který je způsobený interpretováním zdrojového kódu za chodu programu [19].

3.4.3 Srovnání

Kompilované programovací jazyky mohou být použity pro vytvoření softwaru, který je spustitelný na různých platformách, je však nutné předem znát všechny platformy, na kterých bude software provozován, aby mohli být připraveny všechny verze programu. Veškeré použité externí knihovny budou muset být vybírány s tím, že mají být multiplatformní, což omezuje výběr. V neposlední řadě bude problematické splnit požadavek pohodlné rozšiřitelnosti programu. Při přidání nové funkcionality bude muset být program znovu zkompilován pro všechny platformy.

Vhodnými kandidáty jsou tak interpretované programovací jazyky. Software napsaný v interpretovaném programovacím jazyce lze snadno přenést na jinou platformu, stačí, pokud bude na nové platformě k dispozici interpret daného jazyka. Software napsaný v interpretovaném jazyce lze snadněji rozšířit o nové funkce a knihovny. Nevýhodou je pomalejší běh programu, avšak vzhledem k tomu, že výsledný software nebude provádět náročné výpočty, bude rozdíl v rychlosti běhu kompilovaného a interpretovaného jazyka zanedbatelný.

Pro účely programu byl nakonec vybrán programovací jazyk Java. Pro jazyk Java je dostupná široká škála knihoven. Java splňuje jak požadavek na přenositelnost, tak i požadavek na snadnou rozšiřitelnost. Je také nejpoužívanějším programovacím jazykem s podílem 15,368 % (data k 1. 1. 2018 [4]).

3.5 KNIHOVNA PRO TVORBU GUI

Dalším důležitým komponentem používaným při tvorbě programu je knihovna pro tvorbu GUI (z anglického Graphical User Interface – grafické uživatelské rozhraní). Z důvodu přenositelnosti programu nebudou uvažovány knihovny, které pro svůj chod využívají nativní knihovny operačních systémů, jako jsou SWT a Qt Jambi a to z podobného důvodu za jakého byly v předchozí části vyloučeny kompilované jazyky. Tyto knihovny sice produkují grafické rozhraní, které je více podobné nativním aplikacím, ale nemusí podporovat všechny platformy, stejně tak některé funkce nemusí fungovat na všech platformách, což stěžuje snadnou rozšiřitelnost programu.

Zbývají tedy knihovny, které jsou distribuované přímo s jazykem Java, tedy AWT, Swing a JavaFX.

3.5.1 AWT

AWT (Abstract Window Toolkit) je původní knihovna pro tvorbu GUI uvolněná společně s první

verzí Javy v roce 1995. K vykreslování grafických komponent využívala nativních knihoven operačního systému, díky čemuž se vzhled Java aplikací blížil vzhledu nativních aplikací. V dnešní době již není podporován.

3.5.2 Swing

Swing je přímým nástupcem AWT. Swing je již nezávislý na platformě. Každý prvek GUI je tedy vykreslený přímo Javou.

3.5.3 JavaFX

JavaFX je nejnovější knihovna pro tvorbu GUI v Javě. Vzhled programu lez snadno měnit pomocí CSS (kaskádových stylů). JavaFX obsahuje vylepšené rozhraní pro obsluhu událostí a také tzv. **properties**, což jsou datové proměnné, jejichž hodnota je sledovatelná (observable) a pomocí vazeb (bindings) může být spojena s proměnnou jiného objektu. Takto definovaný vztah mezi objekty zajišťuje, že jakákoliv změna na jednom objektu, se projeví na objektu druhém [5]. Pomocí properties lze například velmi jednoduše zajistit, že některé prvky GUI budou nekativní, dokud nebudou splněny určité podmínky. Pokud má být například tlačítko neaktivní dokud uživatel nevyplnění nějaké textové pole, stačí oužít příkaz `Button.disableProperty().bind(TextField.textProperty().not())`.

JavaFX také obsahuje rozhraní pro integraci komponent vytvořených pomocí knihovny Swing [10].

Pro tvorbu programu bude použita knihovna JavaFX a to zejména díky možnosti použití properties.

3.6 KNIHOVNA PRO TVORBU DIAGRAMŮ

Další důležitou součástí programu bude tvorba diagramů. Z bodu 1 v kapitole „Požadavky na software“ plyne, že zvolená knihovna by měla být distribuována pod některou z licencí pro svobodný software.

Z velkého množství dostupných knihoven byly vybrány ty, které se nejčastěji objevovaly v nejrůznějších doporučeních [14][15]. V této části dojde k jejich srovnání. Ještě předtím je však potřeba určit kritéria výběru.

Protože je program psaný za pomoci JavaFX, měla by být i knihovna pro tvorbu diagramů napsána primárně pro použití s JavaFX. Použít by šly také knihovny pracující s grafickým rozhraním Swing, jelikož JavaFX disponuje možnostmi zobrazovat komponenty rozhraní Swing.

Dalším důležitým kritériem je počet typů diagramů, které knihovna umí vykreslit. V první verzi programu sice nebudou použity všechny typy diagramů, ale výběrem bohaté knihovny roste počet možností jak program v budoucnosti rozšířit. Diagramy by mělo být možné jednoduše ukládat jako obrázek či data diagramu exportovat například do CSV.

Neméně důležitým kritériem jsou pak možnosti týkající se vzhledu diagramů. Tím je myšlena barva datových řad, pozice legendy, velikost, popisky os, pozadí diagramu, atd. V ideálním případě by měl ít možnost jednoduše vše změnit uživatel přímo v zobrazeném diagramu, tedy oblasti diagramu by měly být aktivní (reagovat na kliknutí).

Posledním kritériem je pak jednoduchost implementace, jinak řečeno, knihovna by neměla být příliš složitá.

3.6.1 JFreeChart

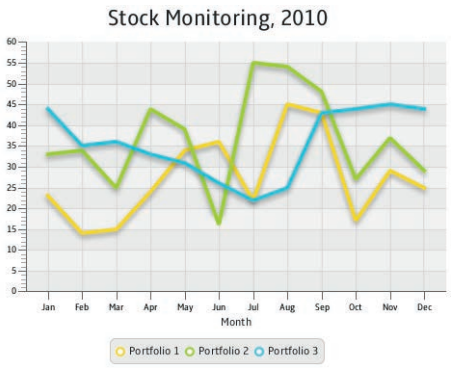
JFreeChart je knihovna distribuovaná pod licencí LGPL (Lesser General Public License). Jedná se o nejpoužívanější knihovnu pro tvorbu diagramů[17]. Knihovna je primárně určena pro rozhraní Swing, ale v nedávné době byla přidána i podpora pro JavaFX, vývoj je však teprve na začátku (ke dni 23. 6. 2018 je aktuální vydaná verze 1.0.1 [18]). Knihovna obsahuje velké množství nejrůznějších diagramů, od základních – sloupcový, liniový, bodový, až po velmi složité grafické diagramy, jakými jsou například teploměry.

Knihovna disponuje širokou škálou nastavení diagramů a oblast vykreslených diagarmů je aktivní.

3.6.2 JavaFX Charts

JavaFX Charts je balík diagramů, které jsou přímo součástí JavaFX [16]. Balík obsahuje základní typy diagramů - sloupcový, liniový, koláčový, plošný, bodový a bublinový. Diagramy mohou být vykreslovány v reálném čase, mají aktivní oblasti a jsou jednoduché na implementaci. Možnost exportu diagramu do souboru by musela být doprogramována pomocí funkce snapshot().

Velkým omezením je však změna vizuální stránky diagramů. Ta probíhá za pomoci CSS, tudíž vzhled diagramu je třeba nastavit před spuštěním programu. Změna za běhu programu je teoreticky možná avšak velmi nepohodlná a často nevede ke kýženým výsledkům. Standardní podoba diagramů obsahuje stíny a velmi pestré barvy, které se nehodí pro prezentaci seriózních vědeckých poznatků.



Obr. 3.3 – Podoba diagramu v JavaFX Charts [16]

3.6.3 XChart

XChart [9] je malá knihovna pro tvorbu diagramů distribuovaná pod licencí (Apache License 2.0). Přes svou malou velikost nabízí velké množství diagramů, včetně histogramů, či radarových diagramů. Nabídka vizuálních změn je široká. Knihovna nabízí uložení diagramu do několika grafických formátů nebo export dat do CSV. Knihovna je primárně určena pro grafické rozhraní Swing a je velmi jednoduchá na implementaci.

Součástí knihovny jsou grafická témata. Jedním z nich je i téma „Matlab", díky kterému je jednodušší vytvářet diagramy podobné těm, které vykresluje Matlab.

Jako knihovna pro tvorbu diagramů byla zvolena knihovna XChart. Knihovna je malá, její implementace je velmi jednoduchá a disponuje dostatečně širokými možnostmi nastavení.

3.7 FORMÁT DAT

Dalším krokem při návrhu programu je definování formátu dat, která jsou předávána mezi softwarem DIC a navrhovaným programem.

3.7.1 Obsah dat

Důležitým faktorem při návrhu vhodného formátu, je obsah dat. Obsah předávaných dat lze rozdělit na dvě skupiny.

První skupinou jsou samotné snímky sledovaného objektu. V každém časovém úseku - **kroku** je pořízeno několik snímků sledovaného objektu z různých kamer. Počet kamer se může pro různé experimenty lišit. Uživatel by měl být při vizualizaci dat schopen mezi jednotlivými snímky libovolně přepínat. Je tedy nutné, aby bylo možné pořízené snímky jednoznačně přiřadit k jednotlivým krokům a snímky v konkrétním kroku přiřadit ke kamerám, kterými byly pořízeny.

Používaný software DIC umí pořízené snímky exportovat do několika grafických formátů(tiff, png, …). Tento způsob však vede k velkému množství souborů. Pokud je průběh experimentu snímán třemi kamerami a snímky jsou pořizovány v 60 krocích, výsledný export pak obsahuje 180 souborů. Snímky jsou navíc přiřazeny k jednotlivým krokům a kamerám pouze za pomoci názvu souboru. Například snímek z kamery č. 1 pořízený v kroku 2 může mít název „camera_1_step_2.tiff". Pokud dojde k přejmenování snímku informace o kroku, ke kterému snímek patří či zdrojové kameře, je nenávratně ztracena.

Druhou skupinou dat jsou pak vypočítané deformace, vektory posunu a informace o zvolených subsetech. Naměřených dat může být velké množství. Existuje totiž mnoho přístupů, jak měřit přetvoření (deformaci) tělesa. V každém kroku je pro každý subset spočítána poloha v prostoru, deformace (různými způsoby) a odchylka naměřených hodnot. Tyto hodnoty mají podobu dvoudimenzionálního pole, kdy index [x, y] odpovídá subsetu v x-tém řádku a y-tém sloupci. Takto strukturovaná data mohou být exportována do textového souboru. Data jsou do souboru uložena ve formě tabulky, kdy jeden řádek souboru odpovídá jednomu subsetu. Sloupce pak představují jednotlivé hodnoty. Nejprve jsou uvedeny jeho souřadnice subsetu x a y, poté souřadnice v prostoru, dále posuvy, atd. Informace o povaze uložených hodnot je obsažena v první řádku souboru, kdy jsou uvedeny názvy sloupců a jejich jednotky. První dva řádky souboru, tak mohou vypadat tako:

```
x y coordinate_x coordinate_y displacement_x/mm displacement_y/mm
250 300 14.1245985 15.4458754845  0.484888 0.26784187 -1.455798778
```

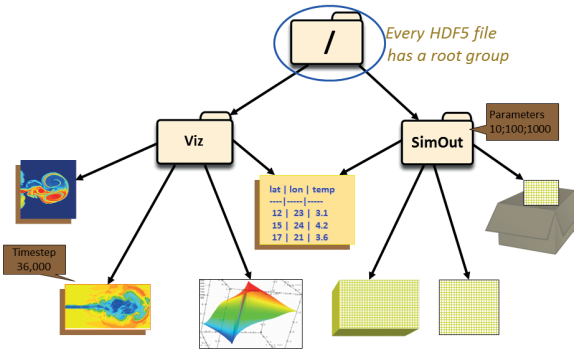
Takto strukturovaný soubor je velmi nepřehledný. Nevhodnou editací souboru lze také data nenávratně zničit nebo způsobit posunutí názvu sloupců a tím zkreslit vyzualizovaná data.

Výše uvedené alternativy formátu souborů s daty a pořízenými snímky se nejeví jako vhodné. Software DIC však podporuje export do datového formátu, který je přímo určen pro zjednodušení uchovávání velkého množství komplexních dat. Tímto datovým formátem je HDF5.

3.8 HDF5

HDF5 je nejnovější verze datového formátu HDF (z anglického Hierarchical Data Format), původně navržený v NCSA (National Center for Supercomputing Applications), nyní spravovaný neziskovou organizací The HDF Group. Cílem formátu je usnadnit organizaci, uložení a předávání zejména vědeckých dat, která se vyznačují velkou komplexností.

Jedná se o hierarchický datový formát, jehož hlavními prvky jsou **skupiny** (group) a **datové sady** (dataset). Celková struktura formátu je inspirována souborovým systémem Unixu [2].



Obr. 3.3 – Struktura HDF5 souboru [2]

3.8.1 Skupiny

Skupiny slouží k organizaci objektů souboru. Každý soubor obsahuje kořenovou skupinu s názvem „/“. Skupiny mohou obsahovat další skupiny nebo datové sady.

3.8.2 Datové sady

Datové sady obsahují data. Datová sada je popsána pomocí datového typu (datatype), datového prostoru (dataspace) a vlastnostmi (properties). Každá datová sada může mít navíc přiřazené atributy, které uložená data detailněji popisují.

3.8.3 Datové typy

Datový typ popisuje jednotlivé hodnoty datové sady. Datové typy jsou rozděleny do dvou skupin:

1. Předdefinované datové typy

   Tyto datové typy jsou definovány a vytvářeny samotnou knihovnou HDF5. Rozdělují se na dva druhy

   a) Standardní – standardní nebo také souborové datové typy jsou stejné pro všechny platformy a používají se při vytváření datových sad. Příkladem je například standardní datový typ integer délky 32 bitů ve fromátu little-endian (`H5T_STD_I32LE`).

   b) Nativní – používají se pro reprezentaci hodnot v paměti. Jejich implementace závisí na platformě, na které HDF5 knihovna pracuje. Standardní datový typ `H5T_STD_I32LE` bude mapován na nativní datový typ `H5T_NATIVE_INT`.

2. Odvozené datové typy

   Odvozené datové typy jsou uživatelem vytvořené datové typy. Tyto typy se vytváří z předdefinovaných datových typů. Typickým příkladem je typ `string`, který může být odvozen z předdefinovaného datového typu `H5T_C_S1` (`char`). Odvozené datové typy mohou být velmi komplexní (mohou být tvořeny několika předdefinovanými datovými typy).

3.8.4 Datový prostor

Datový prostor popisuje uspořádání dat v datové sadě. Základními vlastnostmi datového prostoru jsou hodnost (rank) – počet dimenzí a rozměry dimenzí (dimensions). Datový prostor se rozděluje na tři druhy:

1. Skalární (scalar) – datový prostor reprezentující jeden element. Element může být odvozeného datového typu, takže může být velmi složitý. Hodnost skalárního datového prostoru je 0.
2. Jednoduchý (simple) – prostor reprezentující multidimenzionální pole vybraného datového typu. Hodnost je tanovena při vytváření datové sady, avšak rozměry pole se mohou měnit.
3. Nulový (null) – datový prostor popisující prázdnou datovou sadu.

Datový prostor může být také použit pro definici podprostoru datové sady a následně využit pro částečné čtení/zápis datové sady.

3.8.5 Vlastnosti

Vlastnosti udávají některé důležité charakteristiky datové sady. Podávají například informace o tom, zda jsou data uložena kontinuálně (contiguous), nebo „po kusech" (chunked). Dále indikují, zda jsou data komprimována (compressed) a jaký typ komprese je použit.

3.8.6 Atributy

Atributy dále popisují datovou sadu. Atributem je dvojice název-hodnota. Atributem může být například čas pořízení snímku, či jiné informace, které jsou užitečné při práci s datovou sadou.

3.8.7 Obrazová data v HDF5

HDF5 neobsahuje předdefinované objekty pro ukládání obrazových dat. Místo toho je specifikována sada atributů, která slouží pro identifikaci obrazových dat v HDF5 souborech. Každá datová sada, která má být HDF5 knihovnou interpretována jako digitální obraz, musí mít tyto atributy nastavené. Nejdůležitějším atributem je atribut „CLASS", který musí mít hodnotu „IMAGE". Dalšími hodnotami jsou pak informace o druhu obrazu (grayscale, truecolor, indexed, …), použité barevné paletě atd. [3]

3.9 STRUKTURA EXPORTOVANÝCH DAT

Z předchozích odstavců vyplývá, že formát HDF5 je vhodným datovým formátem pro přenos dat mezi softwarem DIC a vizualizační programem. Následuje popis struktury předávaných dat. Přestože by struktura HDF5 souborů umožňovala ukládat snímky z kamer a z nich vypočtená data do jednoho souboru, není tomu tak a software DIC vygeneruje dvě sady souborů. Jedna sada obsahuje snímky

pořízené kamerami a druhá vypočtená data. Počet souborů v každé sadě odpovídá počtu provedených kroků.

3.9.1 Snímky

Všechny snímky pořízené v jednom kroku jsou uloženy do jednoho HDF5 souboru. Tento soubor je pojmenován tak, aby z názvu souboru bylo jednoznačně identifikovatelné, k jakému kroku obsah souboru patří. Typický název vypadá takto: „series_step_x.hdf5“, kde **x** je číslo kroku ke kterému soubor patří. Obsahem tohoto souboru je skupina, jejíž název odpovídá názvu projektu a obsahem této skupiny jsou pak snímky z jednotlivých kamer. Struktura souborů v rámci jednoho projektu zůstává neměnná, tzn., že cesta ke snímku z kamery č. 1 bude stejná v kroku 1 i v kroku 70.

Snímky jsou v souboru označeny jako obraz typu „grayscale“ s hodnotou pixelu reprezentovanou datovým typem „16-bit unsigned integer“.

3.9.2 Data

Data vypočtená sofwarem DIC jsou také uložená v souborech po krocích. Název HDF5 souboru opět indikuje, k jakému kroku data patří. Typický název pro soubor s vypočtenými daty je „series_step_x.hdf5“, kde **x** je číslo kroku ke kterému soubor patří. Samotná data jsou pak uložena jako dvourozměrné datové sady různých datových typů (float, double, integer, byte, …). Datové sady mohou, ale nemusejí, být uspořádány do skupin. Například datové sady vztahující se k výpočtu posuvů jednotlivých subsetů budou pravděpodobně uloženy ve skupině s názvem „displacements“ (přesuny), záleží však na nastavení exportu.

Každý datový soubor také obsahuje důležitou datovou sadu s názvem „Mask“ (maska). Maska udává, které subsety aktivně vstupovaly do výpočtů. Protože součástí snímku není jen snímaný objekt, ale i jeho bezprostřední okolí, některé subsety vůbec nemusejí pokrývat prostor, který v obrazu zabírá sledovaný objekt.

Subsety, které se softwaru DIC podařilo identifikovat jako subsety, které jsou součástí sledovaného objektu, jsou v masce označeny hodnotou 1, v opačném případě jsou označeny hodnotou 0.

4 ZÁKLADNÍ FUNKCE PROGRAMU

Výsledný software by měl mít implementovány následující funkce.

1. **Funkce načtení dat, zobrazení a jejich editace**

   Software bude umožňovat načtení dat, která jsou výstupem softwaru používaným na Katedře mechaniky a umožní jejich editaci.

   Software bude dále schopen načíst pořízené snímky a zobrazit je. Zobrazeny budou současně dva snímky vedle sebe, tzn., že bude možné souběžně sledovat snímky ze dvou různých kamer. Zdroj snímků bude moci uživatel měnit. Budou-li se v datech nacházet snímky ze tří kamer, bude možné vlevo zobrazit snímek z kamery č. 1, vpravo snímek z kamery č. 2 a v případě potřebu přepnout zdroj levého snímku na kameru č. 3 apod.

2. **Funkce přehrávače**

   Software umožní přehrání všech načtených kroků v animaci. Uživatel bude moci měnit rychlost animace. Současně bude možné některý ze snímků v určitém kroku pozastavit, takže při přehrávání animace se bude měnit např. pouze levý snímek a pravý snímek zůstane zafixovaný v určitém kroku.

3. **Funkce zobrazení teplotní mapy**
   Na základě načtených dat bude možnost zobrazit teplotní mapu. Mapa bude zobrazena přes původní snímek a bude mít nastavitelnou průhlednost. Datová sada, na základě které bude teplotní mapa vygenerována, bude nastavitelná pro levý a pravý snímek nezávisle.

4. **Funkce označování bodů zájmu**
   Uživatel bude mít možnost označit ve snímku body, kterého zajímají, a pro takto označené body bude možné vygenerovat diagram.

5. **Funkce generování diagramů**
   Vzhled diagramu bude co nejvíce nastavitelný. Důležitá je nastavitelnost popisků jednotlivých os, pozice legendy, barva a název datových řad a celková velikost diagramu. Digramy bude možné exportovat do různých formátů.

5 IMPLEMENTACE

5.1 ARCHITEKTURA

Při návrhu softwaru byl kladen důraz na oddělení datové a prezentační části. Architektura programu vychází z návrhového vzoru MVC (z anglického Model-View-Controller).

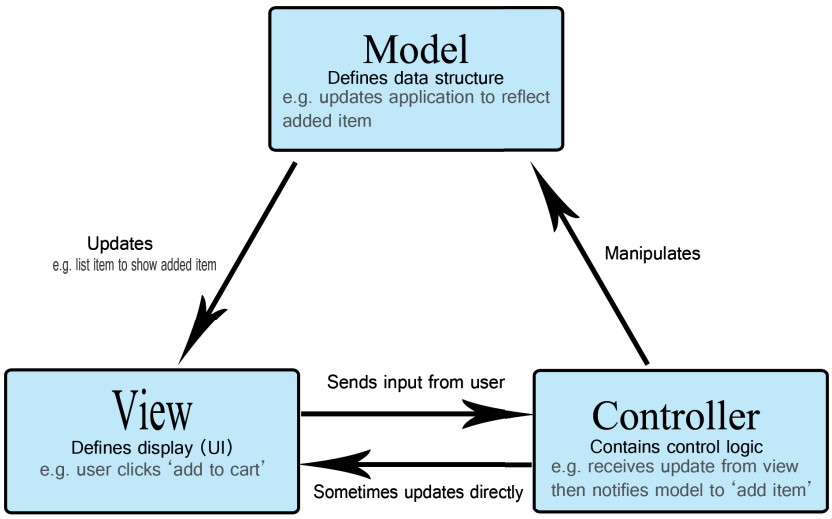
5.1.2 MVC

MVC je návrhový vzor, který dělí aplikaci na tři části - view (pohled), model a controller (řadič).

Pohled zobrazuje data.

Model udržuje do data, se kterými program pracuje, současně s daty manipuluje – načítá nové, mění stávající. V případě změny dat model notifikuje pohled, které data následně zobrazí.

Controller funguje jako prostředník mezi modelem a pohledem. Pokud uživatel provede akci, například stiskne tlačítko, Pohled upozorní controller, který akci vyhodnotí a následně manipuluje s modelem [6].



Obr. 5.1 – Diagram MVC [6]

5.2 STRUKTURA

Celý projekt je rozdělen do několika balíčků. Následuje jejich popis.

5.2.1 Classes

Obsahuje pomocné datové třídy.

- **`CameraView`** – objekt pro reprezentaci jednoho snímku. Udržuje si zdroj snímku (ze které kamery snímek načítat), aktuální načtená data obrázku, aktuální načtená data z datového souboru a také aktuální podobu teplotní mapy pro načtená data
- **`ChartData`** – objekt reprezentuje jednu datovou řadu diagramu. Uchovává informace o barvě datové řady, značce, která bude v diagramu použita pro vykreslení bodů. Dále obsahuje načtená data pro obě osy a souřadnice subsetu, ke kterému načtená data patří
- **`ChartSettings`** – objekt reprezentuje obecné nastavení diagramu. Tedy, pozici legendy, název diagramu, popisky os a seznam všech datových řad (seznam objektů třídy `ChartData`).
- **`DataLoader`** - objekt načítá datové sady z datových HDF5 souborů. Konstruktoru je předána cesta ke složce obsahující datové soubory. Obsah složky je načten, seřazen sémanticky dle názvu a uložen do struktury `ArrayList`. Po zavolání metody `loadStep(int step)` je otevřen soubor odpovídající požadovanému kroku, reference na tento otevřený soubor je uložena, současně je načtena datová maska pro daný krok. Pomocí metod `getHObject`, `getDataset`, `getGroup` a dalších lze z otevřeného souboru načítat datové sady, skupiny, apod. Třída také obsahuje metody, pomocí kterých lze načítat data i z jiného než právě načteného kroku.
- **`DicVizLoggerConfiguration`** – načítá nastavení pro `java.util.logging.logger`.
- **`FilenameComparator`** – komparátor pro řazení souborů. Pokud by pro řazení souborů bylo použito přirozené řazení, vznikaly by následující posloupnosti:
  `soubor_1.hdf5, soubor_11.hdf5, soubor_2.hdf5`
  Toto chování je nežádoucí, protože pořadí souborů by neodpovídalo pořadí kroků. Proto musí být soubory seřazeny sémanticky. Soubory z předchozího příkladu budou sémanticky seřazeny následovně:
  `soubor_1.hdf5, soubor_2.hdf5, soubor_11.hdf5`
  Pro implementaci sémantického řazení slouží právě třída `FilenameComparator` [6].

- **`ImageLoader`** – pracuje podobně jako `DataLoader`. Konstruktoru je předána cesta ke složce s obrazovými soubory. První soubor je otevřen a prohledán. Pro všechny nalezené obrázky je uložena cesta. Zavoláním metody `loadStep(int step)` je otevřen soubor odpovídající požadovanému kroku, pomocí metod `loadImage` jsou načítány konkrétní snímky. Stejně jako v případě třídy `DataLoader` i tato třída obsahuje metody, pomocí kterých lze načítat data i z jiného, než právě načteného kroku.
- **`IntegerFilter`** – třída implementuje číselný filtr. `IntegerFilter` je používaný jako `TextFormatter` pro textová pole, do kterých lze zadat pouze celé číslo. Pokud se uživatel pokusí zadat nečíselné hodnoty, jsou pomocí této třídy odfiltrovány.
- **`Marker`** – objekt reprezentující značky, které uživatel umisťuje do snímků. Udržuje si údaje o poloze značky a také souřadnice subsetu, do kterého označené místo spadá.

5.2.2 Controllers

Obsahuje controllery pro jednotlivé pohledy. Každý controller má definovánu sadu paramterů, které příjímá z ostatních částí programu. Tyto parametry slouží k předávání informací. Například controller spravující pohled detailu datové sady dostane v rámci parametrů informaci o tom, kterou datovou sadu má zobrazit. Pokud nejsou controlleru předány všechny povinné parametry, pohled není zobrazen.

Tabulka číslo 1 – Parametry controlleru

Název controlleru	Parametry (název, typ, datový typ)
ChartController	chartSettings - povinný - ChartSettings
ChartSettingsController	dataDir - povinný - String markers - povinný - ObservableList<Marker>
DatasetController	item - povinný - HObject
ImagesController	dataDir - povinný -String imagesDir - povinný - String
SourcesController	Žádné

5.2.3 Exceptions

Obsahuje definice speciálních výjimek programu.

- **`NoDataMaskException`** – výjimka je použita pokud soubor s vypočtenými daty neobsahuje datovou masku.
- **`NoSourceFilesException`** – výjimka indikuje, že uživatelem zadané složky neobsahují žádné použitelné HDF5 soubory.

- **`NotADatasetException`** – použije se, pokud aktuálně zpracovávaný HObject není instance třídy Dataset, ale měl by být.
- **`NotAGroupException`** – použije se, pokud aktuálně zpracovávany HObject není instance třídy Group, ale měl by být.

5.2.4 Helpers

V tomto balíčku jsou umístěny třídy obsahující statické pomocné metody. Tyto metody jsou využívány různými částmi programu a jejich implementace nezávisí na instanci třídy. Pro větší přehlednost jsou metody rozděleny do několika tříd.

- **`DialogHelper`** – obsahuje metody pro tvorbu různých dialogových oken.
- **`ColorHelper`** – obsahuje metody pro práci s barvami a paletami (tvorba palet, převod barev mezi AWT a JavaFX, …).
- **`HDF5Helper`** – obsahuje metody, které pracují s HDF5 soubory.
- **`Helper`** – obsahuje všechny ostatní pomocné metody, například metodu pro výpočet minima a maxima v poli.

5.2.5 Interfaces

V tomto balíčku jsou umístěna všechna rozhraní.

5.2.6 Misc

Obsahuje různé pomocné datové struktury, například enumerátory nebo třídu s konstantami.

5.2.7 Models

Obsahuje modely pro všechny pohledy.

5.2.8 Views

Obsahuje FXML soubory všech pohledů.

5.3 ZMĚNA POHLEDŮ

Protože výsledný program bude mít více pohledů, je potřeba vyřešit, jak se bude mezi jednotlivými pohledy přepínat. Pro přepínání pohledů byla vytvořena třída `WindowManager`, která využívá návrhový vzor **singleton**.

Singleton je návrhový vzor, který zaručuje, že od dané třídy bude existovat pouze jedna instance. Pokud se nějaká část programu dožaduje instance dané třídy, je nejprve zkontrolováno, zda již instance neexistuje, pokud ne, vytvoří se nová instance a odkaz na instanci se uloží do statického atributu třídy. Při další žádosti o instanci třídy se již nevytváří instance nová, ale je předán odkaz na již existující instanci [13].

Při spuštění programu je vytvořena instance třídy `WindowManager`, instanci je předána reference na hlavní okno (stage) programu. `WindowManager` do hlavního okna vloží `BorderPane`, do kterého pomocí metody `openView` vkládá jednotlivé pohledy.

Každý controller použitý v programu dědí z abstraktní třídy `BaseController`. To znamená, že každý controller má tyto atributy:

- **`Stage stage`** – reference na okno. Nastaveno pokud je pohled spuštěn ve vlatním okně
- **`String taks`** – krátký popis činnosti controlleru/pohledu.
- **`boolean ownStage`** – indikuje, zda má pohled pracovat ve vlastním okně.
- **`BooleanProperty finished`** - indikuje, zda controller dokončil svou činnost.
- **`WindowManager wm`** – reference na instanci třídy WindowManager.

Každý controller současně obsahuje tyto metody:

- **`String getTask()`** – vrací textový popis činnosti, kterou controller provozuje. Tento popis je použit jako nadpis okna.
- **`void end()`** – volá se před ukončením controlleru. V této metodě dojde k nastavení atributu finished.
- **`void prepareOwnStage(Stage stage)`** - slouží k nastavení reference na okno, ve kterém je pohled zobrazen.
- **`boolean hasOwnStage()`** – vrací hodnotu atributu ownStage.
- **`void setParams(Map<String, Object> params)`** – slouží pro předání parametrů, které controller potřebuje k činnosti.

5.3.1 Metoda `openView`

Metoda má dva parametry:

- **`String viewName`** – název ohledu (jméno FXML souboru bez přípony), který má být zobrazen,
- **`Map<String, Object> params`** – parametry, které mají být předány controlleru.

Průběh metody vypadá následovně.

Nejdříve je načteno FXML pohledu, tím také dojde inicializaci controlleru. Poté je pomocí metody controlleru `hasOwnStage` zjištěno, zda mát být pohled zobrazen v novém okně. Pokud ano, `WindowManager` okno vytvoří a pomocí metody `prepareOwnStage` ho controlleru předá. Jestliže pohled vlastní okno nepotřebuje, vloží pohled do středu `BorderPane`, který je součástí hlavního okna. Na závěr jsou za pomoci metody `setParams` controlleru předány parametry.

Následuje výpis kódu metody.

```
/**
   * Opens desired view
   *
   * @param viewName view to open
   * @param params  params to pass to view controller
   */
  public void openView(String viewName, Map<String, Object> params) {
    FXMLLoader loader = new FXMLLoader();
    try {
      Parent root = (Parent) loader.load(this.getClass().getResourceAsStre-
am(Helper.getViewPath(viewName)));

      // check controller properties
      BaseController controller = loader.getController();

      if (controller.hasOwnStage()) {
        // show view in new stage
        Stage st = new Stage();
        st.initModality(Modality.APPLICATION_MODAL);
        st.setScene(new Scene(root));
        setStageTitle(controller.getTask(), st);
        controller.prepareOwnStage(st);
        st.show();
```

```
    } else {
      // inject view into primary stage
    ((BorderPane) this.primaryStage.getScene().getRoot()).setCenter(root);
      this.setPrimaryStageTitle(controller.getTask());
    }
    controller.setParams(params);
  } catch (IOException e) {
    LOGGER.log(Level.WARNING, e.toString(), e);
  }

}
```

Pokud některý z controllerů potřebuje zobrazit jiný pohled (například po tom, co uživatel stiskl tlačítko pro vytvoření diagramu), připraví nejprve controller mapu s parametry a příkazem `wm.openView(„viewName", params)`, řekne třídě `WindowManager`, jaký pohled má být zobrazen.

5.4 NAČÍTÁNÍ DATOVÝCH SAD

Následující část se zabývá procesem načítání datových sad z HDF5 souborů a následnou prací s nimi.

Nejdříve jsou pomocí metod třídy `HDF5Heleper` ze souboru načtena surová data. Data jsou HDF5 knihovnou vrácena jako `Object`. V programu se s tímto formátem pracuje co nejdéle. Ve chvíli kdy je z dat třeba získat konkrétní hodnotu nastávají dvě možnosti, jak takovou hodnotu získat. Obě možnosti vycházejí z předpokladu, že v datových souborech jsou uložena číselná data, která reprezentují dvoudimenzionální pole (tabulka). Tento předpoklad lze ověřit tím, že program zkontroluje, zda je datová sada, ze které mají být data načtena, instancí třídy `ScalarDS`. Do objektu typu `ScalarDS` lze uložit pouze hodnoty, které lze interpretovat jako obrázek, nebo tabulku. Hodnoty mohou nabývat typů `char, byte, short, int, long, float a double`[13]. Přestože data reprezentují dvoudimenzionální pole, jsou uložena jako jednodimenzionální pole po řádcích (Row-Major order).

Jelikož obsahem datových souborů mají být vypočtené hodnoty posuvů, deformací a souřadnic, je rozumné předpokládat, že data budou numerická. Nechť jsou data načtena do proměnné `Object data` a `i` je index, ze kterého je třeba získat hodnotu. Pomocí příkazu `Double.parseDouble(Array.get(data, i))` lze získat hodnotu na pozici `i` ve formátu `double`. Při použití této metody dojde k oříznutí několika číslic za desetinnou čárkou (z čísla 1,888293259819443E-14 se stane 1.8882933E-14), taková změna nevadí při vykreslování teplotní mapy, kde by vedla jen k nepatrnému rozdílu odstínu. V případě počítání maximální a minimální hodnoty pole by ale vedla k nepřesným výsledkům.

Druhá metoda zachovává přesnost čísel. Nejprve je příkazy

```
String className = data.getClass().getName();

Char classSymbol = className.charAt(className.lastIndexOf(„[„)
+ 1);
```

zjištěn znak, kterým je datový objekt (o kterém je známo, že je pole určitého typu) reprezentován v JVM (Java Virtual Machine). Vztah mezi získaným znakem a datovým typem ukazuje následující tabulka.

Tabulka číslo 2 – Tabulka vztahu mezi datovým typem a znakem

Element pole	**Znak**
boolean	Z
byte	B
char	C
class or interface	L*classname*;
double	D
float	F
int	I
long	J
short	S

Pomocí příkazu `switch` je pak možné přetypovat datový objekt na pole a dále s ním pracovat.

5.4.1 Příprava hodnot datové sady pro TableView

Standardním způsobem jak připravit data pro vložení do `TableView` je vytvořit datový model a pro každý atribut tohoto modelu, který má být zobrazen v tabulce, vytvořit nový sloupec. V případě datový sady tento postup není možný, protože počet sloupců není předem znám.

Dvoudimenzionální pole lze do `TableView` vložit tak, že je z dat vytvořen `ObservableList` jednotlivých řádků. V případě datové sady bude `ObservableList` vytvořen jako `ObservableList<Object[]>`.

Samotná data z datové sady jsou načtena jako `Object[][]`, tedy dvoudimenzionální pole objektů. O načtení objektů se stará metoda `readDatasetAs2dArray` ve třídě `HDF5Helper`. V ní je využito funkce knihovny HDF5, kdy lze pro právě čtenou datovou sadu nastavit, že nemá být přečtená celá sada, ale pouze její část. K tomu slouží atributy `startDims`, `selectedDims` a `stride`. Pomocí těchto atributů je nastaveno, aby se z datové sady přečetl vždy jeden řádek dat. Výsledkem je pak požadované dvoudimenzionální pole.

Toto pole je pak příkazem `Arrays.asList(array)` převedeno na již zmiňovaný `ObservableList<Object[]>`. Následně je vygenerován počet sloupců typu `TableColumn<Ob-`

ject[], Number>, který je roven počtu sloupců v datové sadě a příkazem TableView.setItems(data) jsou data vložena do tabulky.

5.5 NAČÍTÁNÍ DIGITÁLNÍCH OBRAZŮ

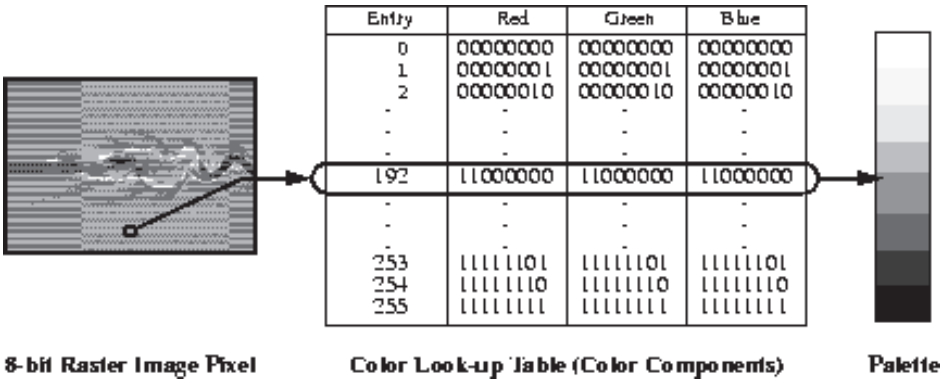
Tato část obsahuje popis procesu načtení dat digitálního obrazu ze souboru HDF5 a jich následného zobrazení.

Nejprve jsou pomocí metod třídy HDF5Helper načtena surová data z HDF5 datové sady. Data obrazu jsou reprezentována jako dvoudimenzionální pole čísel, kde každá hodnota pole odpovídá jednomu pixelu. Obraz je do souboru ukládán od levého horního rohu. Tedy hodnota pole na indexu [0,0] dopovídá pixelu v horním levém rohu, hodnota pole na indexu [0,20] dvacátému pixelu v prvním řádku obrazu, atd.

Načtená data je dále potřeba interpretovat, tedy asociovat hodnoty pole s určitou barvou. Protože se jedná o černobílý obraz, je použita černobílá paleta o 256 barvách. Pro každou hodnotu pixelu je v paletě vyhledána intenzita červené, zelené a modré barvy. Jelikož má paleta pouze 256 barev a hodnoty pixelů jsou typu 16-bit unsigned integer, tedy hodnoty v rozmezí 0 – 65 535, je nejprve nutné hodnoty pixelů převést na byte. Převod se provádí za pomoci lineární transformace

v'=(v-min)/(max-min) *255 [11],

kde v je původní hodnota pixelu, min je minimální hodnota a max maximální hodnota načtených pixelů. Pro takto upravené hodnoty již lze v paletě nalézt odpovídající odstín jednotlivých barev. Tyto tři hodnoty jsou pomocí bitových operací spojeny do jedné proměnné typu int. Význam bitů této proměnné zleva je následující – prvních osm bitů je informace o průhlednosti, dále vždy osm dalších bitů náleží jednotlivým barvám v pořadí červená, zelená, modrá. Tímto způsobem jsou vypočítány hodnoty pro každý pixel, které jsou následně využity k vytvoření objektu BufferedImage typu TYPE_INT_ARGB.



Obr. 5.2 – Vyhledávání barvy na základě palety [3]

5.6 UMISŤOVÁNÍ UŽIVATELSKÝCH ZNAČEK

Uživatelské značky lze umisťovat do levého snímku. Jedná se o `ImageView`, které má nastavený posluchač na kliknutí myši. Po kliknutí do `ImageView`, se spustí kód metody `onImageClick` v controlleru `ImagesController`. Z provedené akce `MouseEvent` jsou získány souřadnice místa kliku. Souřadnice jsou společně s referencí na `ImageView` předány konstruktoru třídy `Marker`. S pomocí těchto dat je vytvořena nová značka.

Za pomoci metody `ImageView.getBoundsInParent()` je získána informace o aktuální velikosti `ImageView` a následně informace o tom, v kolika procentech šířky a výšky se značka nachází. Metoda `getBoundsInParent()` je pužita z toho důvodu, že jiné metody `ImageView` např. `getFitWidth()` a `getFitHeight()` vracejí hodnotu 0.

```
xPercentage = x / parent.getBoundsInParent().getWidth();
yPercentage = y / parent.getBoundsInParent().getHeight();
```

Jako značka je použit obrazec Rectangle. Velikost jeho strany jsou 3 % celkové šířky `ImageView`. Tato hodnota byla zvolena po otestování několika možností. Protože je velikost nastavena relativně, může se velikost značky měnit s velikostí okna, tudíž značka zabírá opticky stále stejně velkou část snímku.

V konstruktoru třídy `Marker` jsou také nastaveny posluchače na změnu velikosti šířky a výšky `ImageView`. Pokaždé, když dojde ke změně velikosti `ImageView`, jsou zavolány metody `resize()` a `changePosition()`. Změna pozice značky je spočítána příkazy

```
symbol.setX((parent.getX() + parent.getBoundsInParent().getWidth() * xPer-
centage) - (size / 2));
symbol.setY((parent.getY() + parent.getBoundsInParent().getHeight() * yPer-
centage) - (size / 2));
```

`parent.getX()` vrací x pozici `ImageView`, **size** je aktuální velikost značky. `Size/2` je odečteno proto, aby byl střed značky v místě, kam uživatel klikl.

U každé značky je také spočítáno, na který subset v datové sadě ukazuje. O to se stará metoda `getMaskCoordinates` umístěná ve tříde `Helper`. Kód metody je následující.

```
/**

     * Get mask coordinates for given image pixel. Each image from camera is
divided to regions. This method  calculates the region coordinates in which
given pixel is in

     *

     * @param x x coordinate of pixel
```

```
     * @param y y coordinate of pixel

     * @param maxX max possible x value (aka width of an image)

     * @param maxY max possible y value (aka height of an image)

     * @param mask datamask

     * @return mask coordinates

     */

  public static double[] getMaskCoordinates(double x, double y, double maxX,
double maxY, DataMask mask) {

         double[] coordinates = new double[2];

         coordinates[0] = Math.floor(x / (maxX / mask.getWidth()));

         coordinates[1] = Math.floor(y / (maxY / mask.getHeight()));

        return coordinates;

    }
```

`maxX / mask.getWidth()` vypočítá šířku jednoho subsetu. Pozice středu značky je vydělena touto hodnotou a tím je získán index subsetu.

Příklad:

X souřadnice značky v `ImageView` má hodnotu 110. Celková šířka `ImageView` je 1000px. Počet subsetů v jednom řádku je 20. Šířka jednoho subsetu je tedy 50px. 110/50 je rovno 2,2. Výsledná **x** souřadnice v masce je 2, tedy třetí subset.

6 GUI

Program obsahuje pět pohledů. Kapitola obsahuje popis jednotlivých pohledů.

6.1 POHLED – ZDROJOVÉ SOUBORY

Tento pohled se nachází v souboru `sources.fxml`. Jedná se o úvodní pohled, který se zobrazuje po spuštění programu, nebo po uzavření načtených dat. Obsahuje dvě textová pole, jedno pro zadání cesty k souborům se snímky, druhé slouží pro zadání cesty k datovým souborů. Cesty lze také zadávat pomocí `DirectoryChooser` dialogu.

6.2 POHLED – SNÍMKY

Pohled se nachází v souboru `images.fxml`. Tento pohled je stěžejním pohledem programu. Je rozdělen na čtyři části: horní, levý, střední a pravý panel. Funkce jednotlivých panelů je rozebrána níže. Pomocí tohoto pohledu uživatel ovládá načítání snímků a vybírá, která data mají být se snímky načítána. Má možnost zobrazit teplotní mapu přes právě zobrazované snímky. Může také zobrazit masku dat, aby věděl, které subsety jsou aktivní.

Horní panel

Horní panel slouží jako panel nástrojů. Jeho součástí jsou tlačítka pro ovládání přehrávače („Play“, „Pause“, „Prev“, „Next“), dále indikátor momentálně načteného kroku, ovládání rychlosti přehrávání, tlačítko pro vyvolání diagramu („Plot“) a tlačítko pro uzavření pohledu („X“).

Levý panel

Levý panel obsahuje informace o právě načtených souborech a uživatelských značkách. V horní části se nachází seznam souborů s obrazovými daty. Vybráním některého ze souboru dojde k načtení příslušného kroku. Seznam je implementován pomocí `TreeView<File>`.

Další seznam, je seznam dostupných datových sad pro daný krok. Jedná se o `TreeView<HObject>`, pro vykreslení `TreeView` je použita vlastní továrna (factory) `DatasetTreeCellFactorz`. Pomocí továrny je na jednotlivé položky navázáno kontextové menu. Kontextové menu má tři položky („Edit“, „Set as LEFT image dataset“ a „Set as RIGHT image dataset“).

Posledním seznamem v levém panelu je seznam uživatelských značek, jedná se o `ListView<Mar-`

ker>. Pomocí privátní třídy MarkerConverter je zajištěno zobrazování jména značky a možnost jejího přejmenování. Položky lze ze seznamu vymazat pomocí tlačítka „DELETE“ nebo vyvoláním kontextové nabídky.

Střední panel

Střední panel je SplitPane, tedy panel rozdělený na dvě části. V každé části se zobrazuje jeden snímek. Do levého snímku má možnost uživatel kliknutím myši umisťovat značky a tím označit subsety, kterého dále zajímají.

Pravý panel

Pravý panel obsahuje nástroje pro ovládání zobrazení snímků ve středním panelu. Pro oba snímky lze nezávisle nastavit zobrazení masky, datový zdroj pro teplotní mapu, nebo zdroj snímku. Nachází se zde také ovládání průhlednosti teplotní mapy. Uživatel zde má také možnost nastavit nezávisle pro každý snímek, ve kterém kroku má být snímek pozastaven.

6.3 POHLED – NASTAVENÍ DIAGRAMU

Pohled je umístěn v souboru chartSettings.fxml. Pomocí tohoto pohledu se nastavují veškeré vlastnosti diagramu – název, popisky os, pozice legendy. Dalším důležitým nastavením je nastavení zdrojových dat pro jednotlivé osy a nastavení kroků, pro které má být diagram zobrazen. Podmínkou pro zobrazení tohoto pohledu je, aby uživatel v pohledu „Snímky“ nastavil alespoň jednu uživatelskou značku. Nastavené uživatelské značky se pak zobrazí v tabulce datových řad, která je umístěna v dolní části okna.

Tabulka datových řad umožňuje uživateli nastavit vlastnosti jednotlivých řad. Jeden řádek v tabulce odpovídá jednomu objektu třídy ChartData. Počet objektů této třídy, které byly vytvořeny, odpovídá počtu uživatelských značek, které byly vytvořeny v pohledu „Snímky“. Každému objektu třídy ChartData je předán název a souřadnice datové značky, na základě které byl vytvořen.

Tabulka má tři sloupce „Dataset Name“, „Dataset color“ a „Dataset Marker“. První sloupec představuje jméno datové řady, jméno lze změnit.

Druhý sloupec umožňuje vybírat barvu datové řady. Jde o speciálně upravenou buňku, která zobrazuje ColorPicker. Vlastnosti buňky jsou nastaveny pomocí privátní třídy ColorTableCell<T> [8].

Třetí sloupec slouží k výběru grafické značky, která bude v grafu reprezentovat bod. Jedná se o ChoiceBox, jehož položky jsou nastaveny pomocí enumerátoru SeriesMarker. ChoiceBox je vykreslen za pomoci privátní třídy MarkerTableCell<T>.

Datové řady lze z tabulky také vymazat. Při vymazání nebo přejmenování datové řady nedochází k editaci uživatelské značky, na základě které byla datová řada vytvořena.

6.4 POHLED – DIAGRAM

Tento pohled je umístěn v souboru `chart.fxml`. Ve FXML souboru se nachází pouze `StackPane`, do kterého je výsledný diagram vložen. Pohled se zobrazí, pokud uživatel potvrdí, že chce na základě nastavení, které provedl v pohledu „Nastavení diagramu“ zobrazit diagram.

Data s nastavením diagramu jsou předána modelu, který pomocí knihovny XChart vygeneruje diagram. Knihovna pracuje s grafickým prostředím Swing, diagram, který knihovna vrátí, je umístěn do `XChartPanel`, což je třída, která dědí od třídy `JPanel`. Aby mohl být panel diagramu vložen do JavaFX `StackPane`, je nejprve nutné jej umístit do objektu `SwingNode`, což je objekt určený k zobrazování Swing komponent v JavaFX [10]. Vytvořený SwingNode je pak umístěn na `StackPane`, čímž dojde k zobrazení diagramu. Diagram lze uložit jako obrázek nebo jeho data exportovat do CSV.

6.5 POHLED – DETAIL DATOVÉ SADY

Pohled se nachází v souboru `dataset.fxml`. Funkcí pohledu je zobrazit data uložená v datové sadě a umožnit jejich editaci. Pohled se skládá ze dvou částí – panelu nástrojů a tabulky s daty.

Panel nástrojů obsahuje tlačítko pro uložení změn („Save“), které není aktivní, dokud nedojde ke změně dat. Dále jsou v panelu umístěny indikátor pozice kurzoru a textové pole, které slouží pro zobrazení hodnoty v buňce tabulky. Toto pole není editovatelné, změna hodnoty v buňce se provede dvojím poklepáním do oblasti buňky.

Tabulka je implementována za pomoci `TableView<Object[]>` (viz „Načítání datových sad“).

7 TESTOVÁNÍ

Testování probíhalo dvěma způsoby. Prvním bylo testování za pomoci Unit testů, které probíhalo během vývoje programu. Původním záměrem bylo otestovat všechny metody s modifikátorem `public`, zejména pak metody sloužící k načítání dat. Celkové pokrytí Unit testy však nejde vyhodnotit (viz kapitola Nedostatky).

Druhým typem testování byly uživatelské testy. Za pomoci dvou testerů, kteří nejprve nebyli seznámeni s funkcemi programu, byla ověřována intuitivnost uživatelského prostředí. Pomocí tohoto testování se podařilo odhalit některé nelogičnosti v umístění ovládacích prvků. Na základě zpětné vazby testerů byla doplněna do programu chybová dialogová okna, která se zobrazují, pokud se nepodařilo správně nastavit controller některého pohledu (k čemuž dochází prakticky pouze v případě, že uživatel špatně vyplní cesty ke zdrojovým HDF5 souborům). Na základě výsledků testování byla také přidána funkce zvýrazňování uživatelské značky, která je právě vybrána nebo editována (takováto značka se zobrazuje červeně). Testeři neměli problém s orientací v uživatelském rozhraní, avšak na některé funkce museli být upozorněni. Jednalo se například o existenci kontextového menu v seznamu datových sad, také měli problém s tím, že změna hodnoty v textových polích musí být potvrzena klávesou „ENTER“, což je ale způsobené implementací `ValueProperty()` v JavaFX.

Poté co byli testeři seznámení s chodem programu, byla testována funkčnost. Na základě těchto testů se podařilo odstranit například duplikující se animaci při přehrávání snímků, která nastala, pokud byl během animace změněn počet snímků za vteřinu (při každé změně došlo k vytvoření kopie animace, kterou nebylo čím zastavit). Další chyba byla nalezena při výběru barvy datové řady v nastavení diagramu. Pokud uživatel nezvolil jednu z předpřipravených barev, ale zobrazil okno s možností nadefinovat si barvu vlastní, nedošlo po potvrzení barvy k její změně v objektu datové řady. Chyba byla odstraněna změnou z `colorPicker.valueProperty().addListener` na `colorPicker.setOnHiding()`. Počet nalezených chyb nebyl velký. Testovací scénáře nebyly pro testování připraveny.

K lepšímu nalézání chyb pomohlo také využití logu. Pro logování je použit `java.util.logging.logger`. Program loguje všechny výjimky. Logování probíhá ve dvou hlavních úrovních. Výjimky, které znemožňují pokračovat v probíhající činnosti důvodu vnitřní chyby, jsou logovány s úrovní „SEVERE“. Jedná se například o případy, kdy nemůže být zobrazen některý pohled, protože k němu příslušící controller nemá dostatek informací k incializaci všech prvků nebo případy, kdy dojde k chybě při čtení datové sady. Výjimky, které jsou způsobeny nevhodným uživatelským vstupem, jsou logovány s úrovní „WARNING“.

8 NEDOSTATKY

V této kapitole budou popsány nedostatky programu.

8.1 NEDOSTATKY V TESTOVÁNÍ

Nedostatků v testování je několik. Prvním nedostatkem je neexistence testovacích scénářů, což může být problém v budoucnu, kdy bude program nabývat na funkcích. V době tvorby programu však nebyl program natolik složitý, aby ověření všech důležitých funkcí činilo potíže.

Druhým nedostatkem v testování bylo (ne)použití Unit testů. Na začátku vývoje byly Unit testy používány velmi intenzivně. Problémy s implementací funkcí knihovny HDF5 a nedostatečná abstrakce na začátku vývoje vedli k tomu, že Unit testy musely být neustále přepisovány, s tím, jak se měnila struktura programu. Časté změny ve struktuře programu a tím i v kódu testů vyústily ve stav, kdy bylo od psaní jednotkových testů upuštěno s tím, že budou dopsány na konci vývoje. Tím byla popřená samotná podstata jednotkových testů. K dopsání Unit testů nakonec nedošlo a při kontrole kvality je tak nutné spoléhat se pouze na výstupy testerů, což je velký nedostatek.

8.2 ZNÁMÉ CHYBY

Dalšími nedostatky jsou chyby, které byly nalezeny, ale nepodařilo se je opravit.

8.2.1 Dialog pro uložení diagramu

Při ukládání diagramu nebo exportu dat z diagramu se má uživateli zobrazit dialogové okno, kde má vybrat cílové umístění výsledného souboru. Použitá knihovna XChart pracuje s grafickým rozhraním Swing a dialogové okno, vytvořené knihovnou se zobrazí pod okny vytvořenými v JavaFX. Uživatel tak může nabýt dojmu, že k zobrazení dialogu nedošlo. K dialogu se lze dostat například pomocí klávesové zkratky alt+tab a poté, co je dialogové okno přeneseno do popředí, lze data diagramu uložit. Chybu se nepodařilo odstranit, protože vykreslení dialogového okna je plně v režii knihovny XChart.

8.2.2 Nevhodně zvolená knihovna pro tvorbu diagramů

Volba knihovny pro tvorbu diagramů se neukázala příliš dobrou. Zejména z důvodu použití rozdílné-

ho uživatelského rozhraní. JavaFX sice podporuje vkládání komponent grafického rozhraní Swing, ale předchozí chyba dokazuje, že integrace se neobejde bez problémů. V budoucnu by tak bylo vhodné knihovnu XChart nahradit jinou knihovnou, založenou na JavaFX.

8.2.3 Detail datové sady

Velmi zřídka dochází k případu, kdy program při pokusu o otevření detailu datové sady zahlásí chybu čtení. Příčinu chyby se nepodařilo nalézt.

9 ZÁVĚR

Cílem práce bylo vytvořit program, který by uměl načíst a vizualizovat data ze softwaru, který je na Katedře mechaniky FAV používán pro korelaci digitálního obrazu.

Na začátku byly uvedeny požadavky na software, na základě kterých proběhl výběr nástrojů k vývoji programu.

Program měl umožnit zejména větší flexibilitu v oblasti generování diagramů. Před samotným vývojem byly stanoveny funkce, které by měl program obsahovat (viz kapitola „Základní funkce programu“). Všechny vyjmenované funkce se podařilo implementovat.

Při tvorbě programu byl kladen velký důraz na budoucí rozšiřitelnost, což se projevilo zejména ve struktuře programu, kdy se autor snažil co nejvíce oddělit datovou a prezentační část. V budoucnu by tak neměl být problém například vyměnit použitou knihovnu na tvorbu diagramů za jinou.

Cennými zkušenostmi do budoucna jsou i chyby, kterých se autor při vývoji dopustil, zejména chyby v oblasti testování.

Pro splnění požadovaných cílů bylo nutné seznámit se souborovým formátem HDF5, což byla pro autora cenná a zajímavá zkušenost. Výstupem je kapitola věnovaná tomuto formátu, která může ulehčit práci s daným formátem jiným vývojářům.

Výsledný program sice není plně použitelný, ale dle názoru autora se jedná o dobrý základ pro budoucí software, který by mělo být možné rozšířit o další funkce tak, aby plně vyhovoval požadavkům.

10 ZDROJE

[1] DAIKA, Aneta. *Měření deformací pomocí digitální korelace obrazu*. Plzeň, 2011. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Ing. Robert Zemčík, Ph.D.

[2] *Introduction to HDF5.* [online]. 2018. Dostupné z: https://portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5

[3] *Image Specification.* [online]. 2005. Dostupné z: https://support.hdfgroup.org/HDF5/doc/ADGuide/ImageSpec.html

[4] TIOBE Index | *TIOBE Index for June 2018* [online]. 2018. Dostupné z: https://www.tiobe. com/tiobe-index/

[5] *Using JavaFX Properties and Binding | JavaFX 2 Tutorials and Documentation.* [online]. 2013. Dostupné z: https://docs.oracle.com/javafx/2/binding/jfxpub-binding.htm#

[6] *MVC architecture* [online]. 2018. Dostupné z: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

[7] *How to Order Versioned File Names Semantically in Java.* [online]. 2018. Dostupné z: https://blog.jooq.org/2018/02/23/how-to-order-file-names-semantically-in-java/

[8] SIMONS, Michael. *Custom editor components in JavaFX TableCells* [online]. 2014. Dostupné z: https://info.michael-simons.eu/2014/10/27/custom-editor-components-in-javafx-tablecells/

[9] *XChart* [online]. 2018. Dostupné z: https://github.com/knowm/XChart

[10] *Embedding Swing Content in JavaFX Applications* [online]. 2013. Dostupné z: https://docs.oracle.com/javafx/8/embed_swing/jfxpub-embed_swing.htm

[11] *Image Viewer* [online]. Dostupné z: https://support.hdfgroup.org/products/java/hdfview/UsersGuide/ug06imageview.html

[12] *ScalarDS (HDFView 2.14).* [online]. 2017. Dostupné z: https://support.hdfgroup.org/ftp/HDF5/hdf-java/current/javadocs/hdf/object/ScalarDS.html

[13] *Singleton pattern.* [online]. 2018. Dostupné z: https://en.wikipedia.org/wiki/Singleton_pattern

[14] *10 Excellent Free Open Source Java Chart Library for Developers* [online]. 2012. Dostupné z: https://www.fromdev.com/2012/09/Free-Open-Source-Java-Charting-Library.html

[15] *What is the best open-source java charting library? (other than jfreechart).* [online]. 2016. Dostupné z: https://stackoverflow.com/questions/265777/what-is-the-best-open-source-java-charting-library-other-than-jfreechart

[16] *Using JavaFX Charts: Introduction to JavaFX Charts [online].* 2014. Dostupné z: https://docs.oracle.com/javafx/2/charts/chart-overview.htm#CJAHHJCB

[17] *JFreeChart.* [online]. 2017. Dostupné z: http://www.jfree.org/jfreechart/index.html

[18] *jfree/jfreechart-fx: Extensions for JFreeChart to support JavaFX client applications.[online]. 2017. Dostupné z: https://github.com/jfree/jfreechart-fx*

[19] *Differences between compiled and Interpreted Languages* [online]. 2013. Dostupné z: https://www.codeproject.com/Articles/696764/Differences-between-compiled-and-Interpreted-Langu

[20] *Heat map* [online]. 2018. Dostupné z: https://en.wikipedia.org/wiki/Heat_map

[21] Bing Pan, Zhaoyang Wang, Zixing Lu, „Genuine full-field deformation measurement of an object with complex shape using reliability-guided digital image correlation," Opt. Express **18**, 1011-1023 (2010)

[22] Digital Image Correlation. [online]. 2018. Dostupné z: http://www.veryst.com/what-we-offer/mechanical-testing-modeling-validation/Testing-Library/digital-image-correlation

[23] *Intro to Box Plots* [online]. 2018. Dostupné z: https://help.plot.ly/what-is-a-box-plot/

PŘÍLOHY

UŽIVATELSKÁ PŘÍRUČKA

NASTAVENÍ ZDROJOVÝCH CEST

Po spuštění programu je uživatel vyzván k zadání cest ke složkám se zdrojovými soubory. Cesty mohou být zadáni relativně nebo absolutně. Cestu lze do příslušných textových polí vepsat ručně, popřípadě lze vybrat zdrojové složky pomocí dialogu, který se zobrazí po kliknutí na tlačítko umístěné vedle textového pole.

Program potřebuje ke svému běhu dva druhy zdrojových souborů. Jedním jsou HDF5 soubory obsahující snímky sledovaného předmětu („Image directory"), druhým druhem souborů jsou HDF5 soubory obsahující naměřená data („Data directory"). Oba druhy souborů musí být v různých složkách.

Po nastavení obou cest a kliknutí na tlačítko „Load" se program pokusí načíst informace ze zadaných složek. Pokud dojde k chybě, je zobrazen chybový dialog, v opačném případě se zobrazí hlavní obrazovka programu.

HLAVNÍ OBRAZOVKA

Popis hlavní obrazovky

Hlavní obrazovka je rozdělena na čtyři části.

Levý sloupec

a) Images – seznam kroků

   Seznam dostupných souborů obsahujících snímky. Seznam je seřazený podle názvu vzestupně. Jeden soubor odpovídá jednomu kroku.

b) Datasets – seznam datových sad

   Pro každý načtený krok zobrazuje seznam dostupných datových sad.

c) Markers – uživatelské značky

   Seznam uživatelských značek

Prostřední sloupec

Je rozdělen na dvě části („Left image“ a „Right image“). Každá zobrazuje jeden načtený snímek.

Pravý sloupec

Obsahuje nastavení pro zobrazené snímky.

Panel nástrojů

V horní části je umístěn panel nástrojů. Ten obsahuje tlačítka pro posouvání snímků, spuštění a zastavení animace, ovládání rychlosti animace, indikátor momentálně zobrazeného kroku apod.

Změna kroku

Změnu právě načteného kroku lze provést několika způsoby.

a) Tlačítky „Prev“ (předchozí krok) a „Next“ (další krok)

b) Výběrem kroku ze seznamu dostupných kroku v levém sloupci

Změna načítané datové sady

Standardně není v rámci jednotlivých kroků načítána žádná datová sada. Datové sady jsou používány k vykreslení teplotní mapy. Datovou sadu lze změnit jednotlivě pro každý zobrazovaný snímek. Kliknutí pravým tlačítkem myši na požadovanou datovou sadu v seznamu datových sad vyvolá kontextovou nabídku. Pro nastavení datové sady pro levý snímek je třeba vybrat možnost „Set as LEFT image dataset“, varianta „Set as RIGHT image dataset“ nastaví tuto datovou sadu pro pravý snímek. Po nastavení datové sady je tato sada načítána při každé změně kroku.

Editace datové sady

Jednotlivé datové sady lze editovat. Po vyvolání kontextové nabídky datové sady (kliknutím pravého tlačítka myši na datovou sadu) a výběru možnosti „Edit“ se zobrazí editační okno. Více v sekci „Detail datové sady“.

Nastavení uživatelských značek

Uživatelské značky jsou používány pro označení bodů zájmu. Pro body zájmu lez dále vygenerovat

diagram. Značky se nastavují pouze v levém snímku, protože hodnoty v datových sadách jsou pro oba snímky shodné. Pro umístění nové značky stačí kliknout na požadované místo v zobrazeném snímku. Nově umístěná značka se zobrazí v seznamu uživatelských značek.

Název značky je automaticky vygenerován. Tento název lze změnit přímo v seznamu značek dvojím poklepáním na název značky.

Odstranění uživatelské značky

Odstranění značky probíhá dvěma způsoby.

a) Výběrem značky v seznamu a následným stiskem klávesy „Delete“

b) Vyvoláním kontextové nabídky kliknutím pravého tlačítka myši na požadovanou značku v seznamu značek a výběrem možnosti „Delete“

Zobrazení masky

Každý snímek je rozdělen do několika regionů. Data jsou spočítána pouze pro regiony, které překrývají sledovaný objekt. Indikátorem aktivních regionů je maska. Masku příslušného snímku lze zobrazit zaškrtnutím políčka „Mask“ v pravém sloupci.

Zobrazení teplotní mapy

Pokud uživatel přiřadil k načtenému snímku datovou sadu, může načtená data zobrazit jako teplotní mapu překrývající snímek. Teplotní mapa se zobrazí po zaškrtnutí políčka „Heatmap – Show“ v pravém sloupci.

Teplotní mapa má nastavitelnou průhlednost. Průhlednost lze změnit posuvníkem „Opacity“.

Změna načítaného snímku

Každý soubor se snímky obsahuje více snímků z různých kamer pořízených v daném kroku. Změna načítaného snímku se provádí výběrem ze seznamu snímků „Image source“ v pravém sloupci.

Přehrání všech kroků

Všechny dostupné kroky lze přehrát jako animaci. Rychlost animace se nastavuje pomocí přepínače „FPS“ v panelu nástrojů. Animace se pouští tlačítkem „Play“ a zastavuje tlačítkem „Pause“.

Nastavení referenčního snímku

Levý nebo pravý snímek lze nastavit jako referenční. Tzn., že uživatel má možnost nastavit z jakého kroku budou snímek a přiřazená data načteny. Referenční snímek není přenačítán během animace.

Referenční snímek lze nastavit po zaškrtnutí políčka „Freeze at step“, poté je zpřístupněno nastavení kroku, ze kterého mají být data pro referenční snímek načtena. Krok lze nastavit prostřednictvím tlačítek se symboly šipek, nebo vložením hodnoty do textového pole. Ve druhém případě je třeba hodnotu potvrdit stisknutím klávesy „Enter“.

Uzavření načtených dat

K uzavření načtených dat dojde po kliknutí na tlačítko „X“ v panelu nástrojů.

Zobrazení diagramu

Pokud uživatel nastavil alespoň jednu uživatelskou značku, může pro ně být vygenerován diagram. Nastavení diagramu se zpřístupní po kliknutí na tlačítko „Plot“.

NASTAVENÍ DIAGRAMU

Možnosti nastavení

Uživatel má možnost nastavit následující prvky diagramu.

a) Název

b) Datový zdroj pro osy

c) Popisky os

d) Umístění legendy

e) Rozmezí kroků, pro které má být diagram vytvořen

f) Název datové řady

g) Barva datové řady

h) Značka datové řady

Nastavení názvu

Název lze nastavit vyplnění textového pole „Title“.

Nastavení popisků os

Pro nastavení popisků os slouží textová pole „X-axis label“ – osa x a „Y-axis label“ – osa y.

Nastavení legendy

Pozici legendy lze nastavit výběrem ze seznamu dostupných pozic „Legend position“. Možnosti jsou následující:

a) InsideNW – uvnitř vlevo nahoře

b) InsideNE – uvnitř vpravo nahoře

c) InsideSE – uvnitř vpravo dole

d) InsideSW – uvnitř vlevo dole

e) InsideN – uvnitř nahoře

f) InsideS – uvnitř dole

g) OutsideE – vně vpravo

h) OutsideS – vně dole

Nastavení rozmezí kroků

Pomocí polí „Steps from“ a „Steps to“ lze nastavit rozmezí kroků, pro které má být diagram zobrazen. Hodnotu lze měnit pomocí tlačítek se symboly šipek, nebo vepsáním hodnoty do textového pole, v ta-

kovém případě však musí být hodnota potvrzena stisknutím klávesy „Enter“.

Nastavení datových zdrojů

Pro každou osu lze nastavit datový zdroj, ze kterého budou data načítána. Datový zdrojem může být datová sada – nastavuje se výběrem ze seznamu datových sad „Select dataset“, nebo lze záškrtkem „Use steps“ nastavit, že se na příslušné datové ose mají zobrazovat jednotlivé kroky.

Příklad:

Uživatel chce zobrazit pohyb jím určeného bodu mezi kroky 15 – 20 po ose x. Pro datovou osu x použije možnost „Use step“, osa tak bude nabývat hodnot 15 – 20. Pro datovou osu y vybere datovou sadu „coordinate_x“.

Nastavení názvu datové řady

Změnit název datové řady lze po dvojím poklepání na název datové řady v seznamu datových řad.

Nastavení barvy datové řady

Po kliknutí na barvu příslušné datové řady v sezamu datových řad se zobrazí dialog pro výběr barvy. Lze vybrat z předpřipravených barev, nebo nastavit barvu vlastí.

Nastavení symbolu datové řady

Pro každou datovou řadu lze vybrat symbol, který bude v diagramu označovat jednotlivé body datové řady. Možnosti jsou:

a) None – žádný

b) Circle – kruh

c) Diamond – kosočtverec

d) SQUARE – čtverc

e) TRIANGLE_DOWN – trojúhelník se základnou nahoře

f) TRIANGLE_UP – trojúhelník se základnou dole

g) CROSS – křížek

h) TRAPEZOID – lichoběžník

i) OVAL – ovál

j) RECTANGLE – obdélník

Zobrazení diagramu

Diagram je zobrazen po stisknutí tlačítka „Create“. Diagram lze zobrazit pouze v případě, že jsou nastaveny datové zdroje pro obě osy.

Uložení diagramu

Po kliknutí prvým tlačítkem na zobrazený diagram je zobrazena kontextová nabídka diagramu. Diagram lze uložit po výběru možnosti „Save as“, kdy je uživateli zobrazen dialog pro uložení. Dostupné formáty jsou PNG, JPEG, GIF a BITMAP.

Výběrem možnosti „Export To“ lze vytvořit CSV soubor s daty diagramu.

DETAIL DATOVÉ SADY

Okno detailu datové sady tvoří tabulka s daty a horní panel nástrojů. Pomocí klávesnice či myši lze vybírat jednotlivé buňky. Aktuální poloha a hodnota buňky se zobrazí v panelu nástrojů.

Pro editaci hodnoty buňky je třeba dvojí poklepání myší. Poté se zpřístupní editace. Novou hodnotu je třeba potvrdit stiskem klávesy „Enter“. Hodnota je před vložením zkontrolována, na případnou nevalidní hodnotu je uživatel upozorněn. Uložení provedených změn se provede až po stisknutí tlačítka „Save“.