Todo list

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Manuál k softwaru GPU-DIC

Ing. Petr Ječmen

Fakulta mechatroniky, informatiky a mezioborových studií
Technická Univerzita v Liberci
Studentská 2
461 17 Liberec

1 Úvod

Algoritmus digitální obrazové korelace (dále jen DIC) se historicky ukázal jako algoritmus vhodný pro analýzu záznamu mechanických dějů pro potřeby analýzy chování materiálu s co největší přesností. Nevýhodou algoritmu je citlivost na vstupní parametry a dlouhá výpočetní doba. Cílem aplikace GPU-DIC je hlavně zkrátit dobu výpočtu a alespoň částečný odhad vhodných vstupních parametrů. Popis jak funguje algoritmus DIC lze nalézt v "Two-dimensional digital image correlation for in-plane displacement and strain measurement:a review, Bing Pan et al; 2009 Meas. Sci. Technol. 2;, http://iopscience.iop.org/0957-0233/20/6/062001." V tomto textu budou zmíněny pouze vybrané části, které byly vylepšeny nebo jejichž pochopením se uživateli ulehčí volba vstupních parametrů. V následující kapitole budou zmíněny pouze teoretické předpoklady pro úspěšnou volbu parametrů na základě teorie, detailnější doporučení týkající se volby parametrů budou pak uvedeny v kapitole 6.

2 Vybrané části algoritmu DIC

V této části textu bude odkazováno na text uvedený v minulé kapitole, je tedy velmi doporučeno výše zmíněný text z webu stáhnout. Odkazy budou ve formě "(DIC-5)", kde číslovka označuje číslo strany.

2.1 Dělba obrazu na facety

Prvním krokem výpočtu je rozdělení obrazu na oblasti, v případě algoritmu jsou nazývány "facety" (ukázku lze nalézt v (DIC-5)). Výpočet probíhá nad jednotlivými facety a lze tušit, že volba velikosti a umístění facetu je klíčová pro úspěch algoritmu. Menší velikosti umožní zachycení detailů, zatímco větší pak nabízejí vetší odolnost proti šumu. V praxi se běžně používají velikosti okolo 15 pixelů, pro zašuměná videa pak není problém používat velikost 20 pixelů a více.

2.2 Odhad deformace facetu

Výpočet pole posunů tkví v odhadu, jak se facet deformoval. Facet se deformuje dle tzv. tvarové funkce, která popisuje možné varianty posunu jednotlivých pixelů ve facetu. Detailní popis funkce lze nalézt v (DIC-5). Hlavním parametrem tvarové funkce je řád funkce - nultý řád popisuje pouze posuny, první řád pak je schopen popsat protažení nebo zkosení, druhý řád už pak dokáže popsat i nelineární deformace. Nesmíme zapomenout, že daná funkce popisuje facet jako celek, nikoliv pixel po pixelu. To je výhodou a zároveň nevýhodou celého algoritmu. Výhodou je podobnost s realitou, kde se testované vzorky deformují opravdu jako celky, ne jako malinkaté pod-části. Nevýhodou je, že algoritmus je schopen otestovat pouze omezený počet variant deformací facetu a pokud řešení není v množině testovaných řešení, algoritmus nám předloží nějaké jiné řešení a není lehké stanovit, jestli je vybrané řešení opravdu to pravé nebo jen nejlepší ze skupiny zadaných. Řešením je předat algoritmu dostatečně velkou množinu možných deformací, to ale velmi prodlužuje dobu výpočtu, takže je nutné najít nějaký kompromis na základě předpokládaných deformací.

2.3 Odhad pole přetvoření

Výstupem algoritmu DIC je pole posunutí jednotlivých bodů (typicky bod = pixel). Pro potřeby mechaniky by bylo dobré kromě samotným posunů znát i pole přetvoření (hodnoty epsilon). Hodnota přetvoření je dána poměrem koncové délky ku počáteční, v diskrétní oblasti lze tedy definovat jako rozdíl hodnot posunů v daném směru (prostá diferenciace hodnot). Bohužel vlivem šumu nelze provést výpočet takto jednoduše. Existují dva přístupy, které dokáží potlačit vliv šumu na výsledné pole. Prvním přístupem je vyhlazení pole posunutí před vlastní diferenciací. Problém tohoto přístupu je volba vhodného vyhlazovacího algoritmu, který by nepotlačil užitečná data. Druhým přístupem a zároveň přístupem, který je implementován v aplikaci, je využití metody nejmenších čtverců. Podrobný popis lze nalézt v "Digital image correlation using iterative least squares and pointwise least squares for displacement field and strain field measurements. Bing Pan, Anand Asundi, Huimin Xie, Jianxin Gao. 2008." Principem metody je aproximace pole

posunutí za pomoci lineární plochy (její předpis lze nalézt ve zmíněném článku) a hledání jejích koeficientů. Vzhledem k přímé závislosti koeficientů rovnice a hodnot posunutí a přetvoření, je možno koeficienty hledat za pomoci lineární algebry. Problém této metody je opět nutná volba parametru. Tím je velikost okolí, ve kterém se bude pole deformací aproximovat plochou. Pokud předpokládáme homogenní deformace, je možno volit velikost okna větší, protože se lépe potlačí vliv šumu ne přesnost výsledku. V případě velkých variací v poli posunutí je nutné volit menší okno, aby byly zachyceny detaily, a tím pádem může docházet ke zkreslení výsledků vlivem šumu. Typicky se velikost okna volí v rozsahu 11 - 21 pixelů.

3 Kroky výpočtu

V této kapitole budou zmíněny kroky, které by měl uživatel vykonat před spuštěním vlastního výpočtu. Nastavení lze provést buď za pomoci grafického rozhraní (viz kapitola 4) nebo lze parametry také nastavit za pomoci skriptu úlohy (viz kapitola 5).

3.1 Vyznačení oblasti zájmu (ROI)

Vzhledem k zaměření této aplikace na analýzu trhací zkoušky je i výpočet tomu uzpůsoben. Typicky záznam obsahuje region, kde je zkoušený předmět uchycen do čelistí a zbytek obrazu je pro analýzu nezajímavý. Proto uživatel může buď vyznačit obdélníkovou oblast(-i), na jejichž ploše bude probíhat výpočet, nebo využije možnosti dynamické tvorby ROI na základě polohy čelistí. V případě dynamické tvorby je nutno vyznačit úvodní polohu čelistí za pomoci kružnic, program poté sleduje polohu čelistí a upravuje oblast zájmu tak, aby byla mezi nimi.

3.2 Vyznačení reálné velikosti

Tento krok je volitelný a slouží k možné unifikaci velikosti plochy pro odhad plochy přetvoření. Aplikaci používá poměr px / mm, který lze opět zadat dvěma způsoby. Buď ve formě čísla do skriptu úlohy nebo za pomoci GUI, kde se vyznačí známý rozměr a jeho reálná velikost a poměr se automaticky dopočítá.

3.3 Velikost facetu

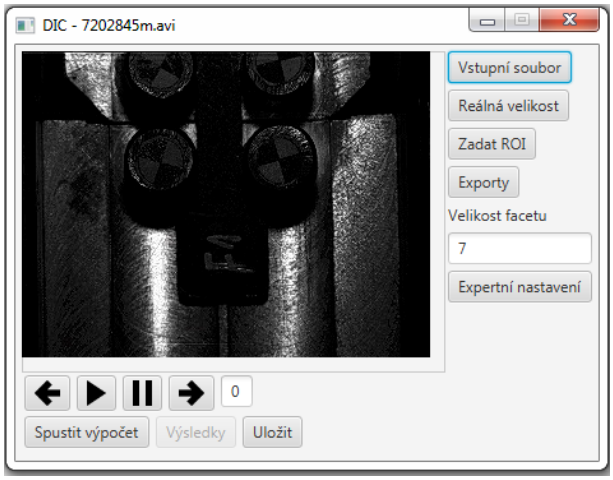
Velikost facetu je hlavním parametrem algoritmu a kvalita výsledky je na něm dost závislá. Doporučené hodnoty lze nalézt v kapitole 6. Číslo musí být celé a opět ho lze zadat přes GUI či skript.

3.4 Expertní nastavení

V některých speciálních případech může být žádoucí změnit i další parametry výpočetního enginu (jako limity deformací, limity kol výpočtu apod.). Co lze a nelze nastavit lze nalézt opět v kapitole 6, některá nastavení lze měnit v GUI, jiná se musí specifikovat za pomoci skriptu úlohy.

4 Ovládání aplikace

4.1 Úvodní okno



Hlavní okno programu

Vstupní soubor načte vstupní souboru(y), podporovány jsou videa (*.avi), obrázky (*.bmp, *.jpg), konfigurace úlohy (*.config), soubor projektu (*.task) a skript pro více úloh (*.scr)

Reálná velikost otevře okno pro vyznačení známého rozměru v obraze

Zadat ROI otevře okno pro vyznačení ROI

Exporty otevře okno pro specifikaci exportů provedených po dokončení výpočtu

Velikost facetu slouží pro zadání velikost facetu použitou během výpočtu

Expertní nastavení slouží pro nastavení specifických parametrů enginu

Šipky slouží pro prohlížení obrázků

Spustit výpočet spustí výpočet

Výsledky otevře okno pro prezentaci výsledků (dostupné po dokončení výpočtu)

Uložit otevře dialog pro uložení skriptu úlohy dle konkrétního nastavení a pro uložení celého projektu (včetně výsledků)

4.2 Výběr vstupu

Video soubor musí mít koncovku avi, video se rozseká na obrázky za pomoci programu VirtualDub

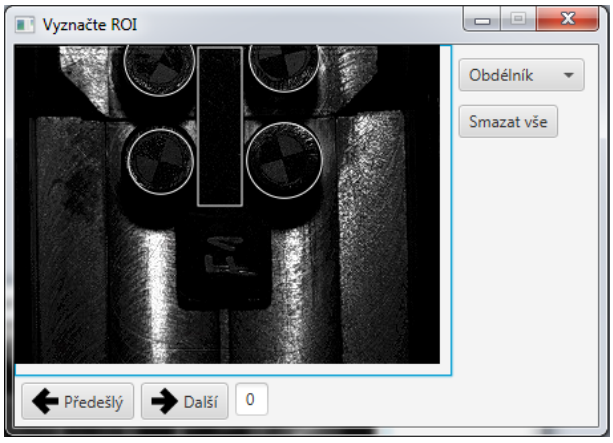
Obrázky se řadí dle přirozeného třídění (tj. většinou dle abecedy)

Config soubor je textový soubor obsahující konfiguraci výpočtu, typicky po načtení mlže být rovnou spuštěn výpočet

Task soubor obsahuje kromě konfigurace úlohy také vypočtené výsledky, které lze okamžitě po načtení souboru prohlížet

Scr soubor by měl obsahovat seznam konfigurací, které se mají spočítat, cesty ke konfiguracím musí být v absolutním tvaru (např. C:\temp\prvni.config)

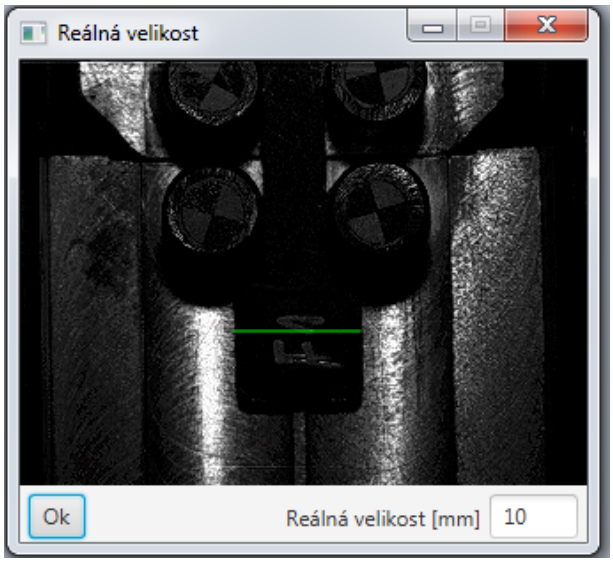
4.3 Zadávání ROI



Okno pro vyznačení ROI

jak se ovládá, jak se chová

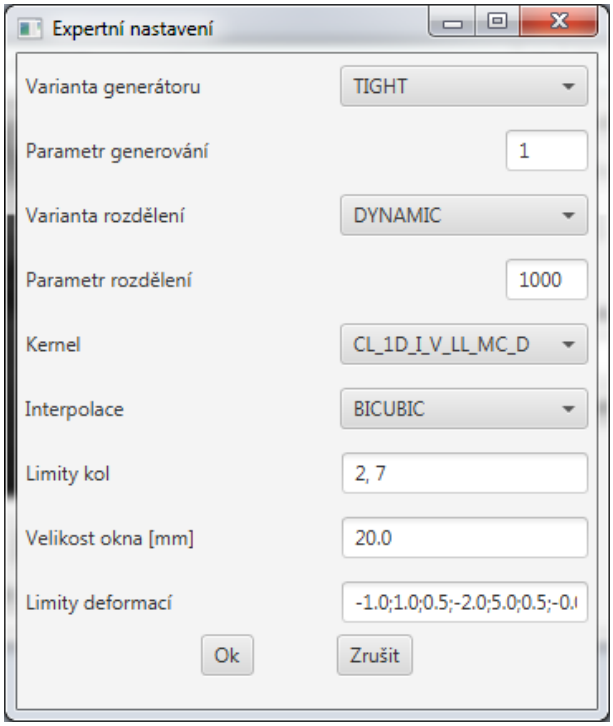
4.4 Vyznačení reálné velikosti



Okno pro vyznačení reálné velikosti

jak se ovládá, jak se chová

4.5 Expertní nastavení



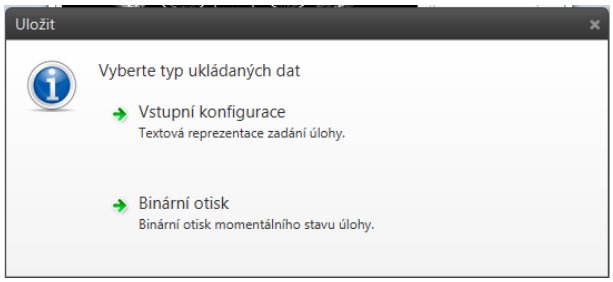
Expertní nastavení

odkaz na sekci s konfigurací úlohy

4.6 Spuštění výpočtu

nic extra, pouze poznámka, že nenastavené parametry se nastaví na výchozí (uživatel se nemusí starat)

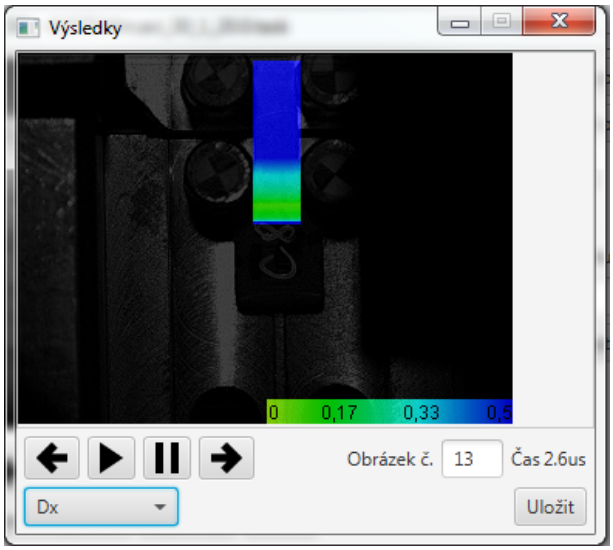
4.7 Uložení



Uložení projektu

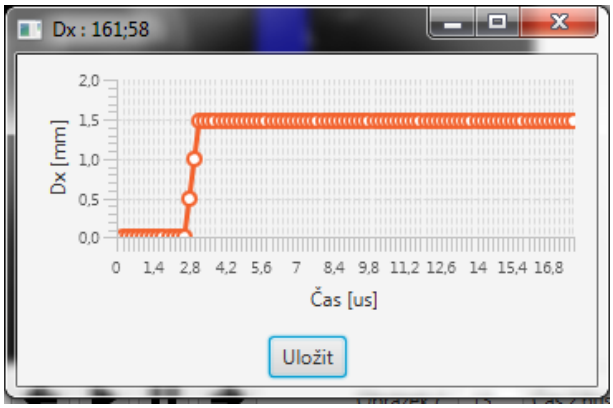
uložení configu / bin. otisku

4.8 Výsledky



Okno s výsledky

zobrazení výsledků a případné uložení



Výsledky pro bod

5 Konfigurační soubor úlohy

detailní popis možných položek v konfiguračním souboru

6 Doporučená nastavení

velikost facetu, parametr výpočtu přetvoření atd.

7 Source code examples

7.1 Instance creation and messaging

Constant declaration

```
final UUID ID = UUID.randomUUID(); // define ID for messaging
```

Client side code

```
public static void main(String[] args) {
  Client client = Client.Client.initNewClient();
  client.getListenerRegistrator().setIdListener(ID, new MessageHandler()
      );
}

class MessageHandler implements Listener<Identifiable> {
  @Override
  public Object receiveData(Identifiable data) {
    if (data instanceof Message) {
      Message m = (Message) data;
      System.out.println(m.getData()); // print content to console
      return m.getHeader(); // send valid response
    } else {
      return "ERROR"; // report error
    }
  }
}
```

7.2 Job Management

Server side code

```
public static void main(String[] args) {
  double start = -100.0, end = 100.0; // define range of values
  double step = 0.01;
  // divide the range in small pieces and create a Set<double[]>, where
     double[] defines start and end of each sub-range and step size
  for (double[] d : subranges) { // submit all jobs
    s.getJobManager().submitJob(d);
  }
  s.getJobManager().waitForAllJobs(); // wait until all jobs are
     complete
```

```
  // use s.getJobManager().getAllJobs() to get all submitted jobs and
     find the best result
}

class DataStore implements DataStorage {
  @Override
  public Object requestData(Object o) {
    switch (o.toString()) {
      case "data":
        return dataArray; // return data
      default:
        return "Illegal data request";
    }
  }
}
```