

Accelerate Digital Image Correlation in C++/Armadillo

Swati Meshram

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology

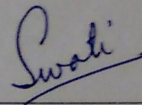


Department of Computer Science & Engineering

June 2016

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.



(Signature)

SWATI MESHARAM

(Swati Meshram)

CS14MTECH11016

(Roll No.)

Approval Sheet

This Thesis entitled Accelerate Digital Image Correlation in C++/Armadillo by Swati Meshram is approved for the degree of Master of Technology from IIT Hyderabad

M. M. [M RAMS]

(——) Examiner
Dept. of Mech
IITH

U. Raha
U. RAMAKRISHNA

(——) Examiner
Dept. CSE
IITH

N. R. Aravind

(Dr N.R. Aravind) Adviser
Dept. of CSE
IITH

M. M. [M RAMS]

(——) Chairman
Dept. of CSE Mech
IITH

Acknowledgements

I would like to express my sincere gratitude to my thesis adviser Dr. N.R.Aravind for his constant encouragement, patience and immense knowledge. Without his guidance and persistent help, this dissertation would not have been possible. I would like to thank Dr. M. Ramji and Navaneeth for their support and help in understanding the domain related concepts. Further, I would also like to thank Dinesh Singh for numerous fruitful conversations and time. Finally, I thank almighty, my family and friends for their support and constant encouragement. .

Dedication

This work is dedicated to my parents and all of my friends without whom none of my success would be possible.

Abstract

Strain and displacement are important parameters in engineering projects. The measurement of these parameters requires an experimental setup to obtain results with accuracy and less cost than the conventional techniques outside the lab. The process of Digital Image Correlation is ideally suitable for carrying out the study for material deformation in real world applications. It has the potential to become a simple, less costly and accurate solution. The thesis work is to implement the process on a computer program using C++/Armadillo which is a faster implementation than its MATLAB version, in order to achieve a speed-up. The application is made to run parallelly using openmp threads to achieve speed gain.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Dedication	v
Abstract	vi
Nomenclature	vii
1 Introduction	1
1.1 What is Digital Image Correlation	1
1.1.1 Definitions	2
1.1.2 DIC Analysis	2
2 DIC Implementation	4
2.1 DIC in MATLAB	4
2.2 DIC in C++/Armadillo	4
2.3 Armadillo vs Matlab	4
2.3.1 Armadillo in detail	5
2.3.2 Armadillo Implementation	5
3 DIC Using GPUs	7
3.1 What are GPUs?	7
3.2 MATLAB with GPU	7
3.2.1 Passing parameters to GPU	8
4 Conclusion	9
References	10

Chapter 1

Introduction

1.1 What is Digital Image Correlation

Digital Image Correlation (DIC) is an optical method in Solid Mechanics. It uses the image processing techniques in an attempt to solve this problem. A large number of images are clicked from the time, load is applied onto some object, till it is fully deformed. These large number of images have to be processed to calculate the deformation rate, strain etc. There are two images which are of primary concern, called the reference image which is actually the initial image under consideration and the other is the deformed image, which is the final image when the object is in deformed state.

A region of interest is selected from the image, where the difference is observed in displacement

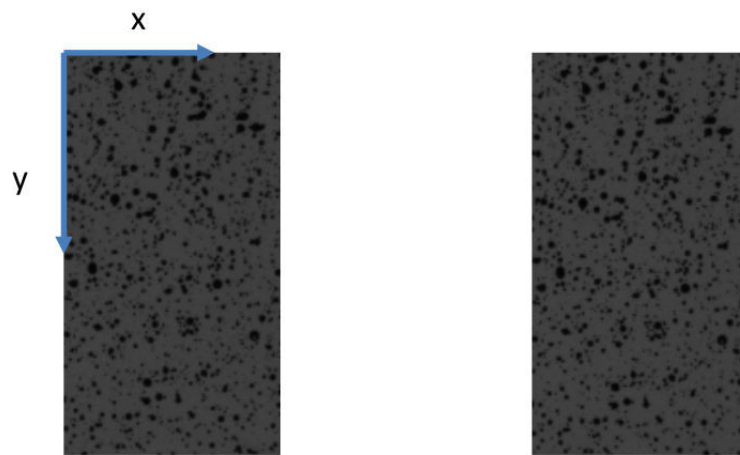


Figure 1.1: Reference Image and Deformed Image

and strain fields. Image processing techniques are used in an attempt to solve this problem. The steps involved in the process of DIC are:

- Prepare the specimen and experimental setup
- Capture images before and after applying load.
- Process these images on computer to obtain the displacement rate.

1.1.1 Definitions

Reference Image : The image under consideration before applying the load.

Deformed Image : The image after applying the load.

The DIC process calculates the transformation from the reference image to the deformed using a computer program.

Region of Interest (ROI) : It is the region in the image where we want to obtain the displacement and strain fields.

Grid points : These are those points in the ROI where we are actually finding the displacement field.

Subsets : The images are divided into overlapping subsets which contains the grid points as their centers.

Correlation criterion : Correlation criterion is used to evaluate the similarity degree between the reference and deformed subsets. Although different definitions of correlation criteria can be found in the literature, these correlation criteria can be categorized into two groups, namely Cross Correlation (CC) criteria and Sum of Squared difference (SSD) correlation criteria. It can be seen that SSD correlation criterion is nothing but the sum of squared errors and both CC and SSD criteria are equivalent. Interpolation is required as it is possible that x_i' and y_i' can acquire inter-pixel values.

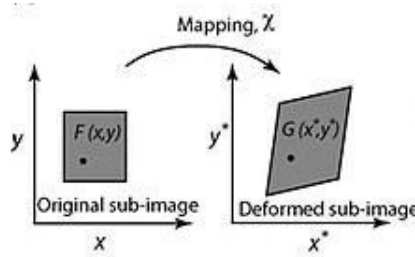


Figure 1.2: Basic concept of deformation mapping

1.1.2 DIC Analysis

The process of DIC following the ICGN algorithm can be summarized as below.

- The reference and deformed images are selected.
- A region of interest (ROI) is selected where we want to study the deformation from the reference and deformed images.
- The points in the ROI where we want to find displacement are called grid points. These grid points are made to be the center of overlapping subsets. All the subsets are selected in the ROI.
- Seed point is selected. The grid point where the analysis begins is called as seed point. It is given manually by the programmer.
- Correlation criteria is evaluated for the seed point. The full displacement vector is evaluated for all the subsets.

- Perform calibration to get the distance in actual dimension.
- Strain is calculated once above steps are performed.

Chapter 2

DIC Implementation

2.1 DIC in MATLAB

MATLAB stands for matrix laboratory. MATLAB is not just a programming language, it is a programming environment as well. MATLAB is used in technical computing with its easy-to-use interface. The basic data element of MATLAB is a matrix. A simple integer is considered to be a matrix of one row and one column. Applications which heavily works on arrays or matrix, are mostly built in MATLAB. The command window provides an interactive interface to directly see whats happening without the need of compiling.

The process of DIC makes extensive use of matrix operations. It is a good practice to implement such applications in MATLAB, as MATLAB comes handy and provides ease of writing.

2.2 DIC in C++/Armadillo

Armadillo is a linear algebra software library for the C++ programming language. It is extremely light-weight and small, also uses BLAS and LAPACK. It is a C++ templated library and is portable across any platform like iOS, Android.

The study of the MATLAB implementation of this application shows that it can achieve or made faster if it is implemented in C++ architecture. Also, C++ is a compiled language which has its benefit in speed of execution when compared to interpreted language as is MATLAB. MATLAB runs the entire code line by line. Hence these points made clear to convert the application from MATLAB to Armadillo to find speed gain.

2.3 Armadillo vs Matlab

Armadillo is a light-weighted process while matlab is heavy-weight. Armadillo provides a similar syntax as matlab to use the built-in function. This helps in porting the application from Matlab to Armadillo without writing much code in C++. The memory allocation is different in both. In Matlab, we don't need to declare and initialize the variable, they can be used on the go. While in Armadillo, one needs to declare the variable which are needed in the program and also take care of

the scope of variables.

MATLAB supports three types of parallel computing, multithreaded (implicit parallelism), distributed computing, and explicit parallelism. In multithreaded parallelism, one instance of MATLAB automatically generates multiple simultaneous instruction streams. Multiple processors or cores, sharing the memory of a single computer, execute these streams. Element wise computations on big matrices might benefit most from such solutions. In distributed computing, multiple instances of MATLAB run multiple independent computations on separate computers, each with its own memory. In most cases, a single program is run many times with different parameters. In explicit parallelism, several instances of MATLAB run on several processors or computers, often with separate memories, and simultaneously execute a single MATLAB command or M-function. New programming constructions, including parallel loops and distributed arrays, describe this parallelism. In this work, we take advantage of implicit parallelism of matlab.

2.3.1 Armadillo in detail

The Armadillo library is deliberately made similar to MATLAB to provide ease of writing. It is a high quality linear algebra library for C++ which aims towards a good balance between computation speed and ease of use. It is useful for developing algorithms directly in C++ or conversion of algorithms from other platforms. It is used in machine learning, computer vision, pattern recognition, signal processing and many more engineering fields.

The indexing starts from 0 in Armadillo while in Matlab its 1. Since Matlab fully deals with matrices, it has an easy way to upgrade the matrix into any dimension while that is not the case with Armadillo. Also the slash and backslash operators in Matlab comes handy to solve the system of linear equations. The solve function in Armadillo provides approximate solution and is similar to the functionality of backslash operator. Also in Armadillo, one needs to keep an eye on the scope of the variables as in C++ programming language. This in Matlab is taken care internally.

2.3.2 Armadillo Implementation

OpenCV : OpenCV is a computer vision library aimed at using image processing and machine learning functions in many programming languages. OpenCV is used to retrieve the images in the program. The images are taken as input in the program using it. Once the images are loaded successfully, the matrices are read and stored. The matrices read by openCV are transposed to obtain a matlab similar matrix. Now the operations similar to matlab can be performed on the image matrix.

<code>A(k, :)</code>	<code>A.row(k)</code>
<code>A(:, p:q)</code>	<code>A.cols(p, q)</code>
<code>A(p:q, :)</code>	<code>A.rows(p, q)</code>
<code>A(p:q, r:s)</code>	<code>A(span(p,q), span(r,s))</code>

Figure 2.1: MATLAB syntax and Armadillo syntax

The basic syntax is similar to MATLAB. The entire application is then written in C++/Armadillo. A region of interest is selected which is a rectangle of same dimensions from both the images. The matrices corresponding to the ROI from both the image are the main matrices where we want to find the deformation. The entire process will be carried out on the ROI.

OpenMP : For the parallel execution of the for loops, openmp is used. Threads are created to achieve parallelism wherever possible. This is done by writing the `#pragma` over the loops in the beginning. The number of threads can be specified, if not it utilizes the processor power of the system. It creates number of threads equal to the number of cores in your system to execute the instructions parallelly.

Chapter 3

DIC Using GPUs

3.1 What are GPUs?

Graphical Processing Units (GPUs), have emerged as the main component of the High Performance Computing (HPC) systems. Over the years, GPUs have evolved as both a programmable graphics processor and a scalable parallel computing platform, which can be used for parallelizing general purpose applications. Scientific and Engineering computing areas such as Digital Image Correlation, Computational Fluid Dynamics, Computer Vision, Weather and Climate Forecasting etc. have problems which are highly data parallel and deals with input of very large size. These applications, which have an ever-growing demand for the power of high performance computing infrastructure can make use of this power efficient and less expensive GPUs. Programming interfaces for GPU computing like Compute Unified Device Architecture(CUDA) introduced by NVIDIA and OpenCL introduced by Khronos group provide the ease for GPU Programming. The interface used in this work is CUDA by Nvidia. A detailed description of CUDA is provided in reference books[1] and [5].

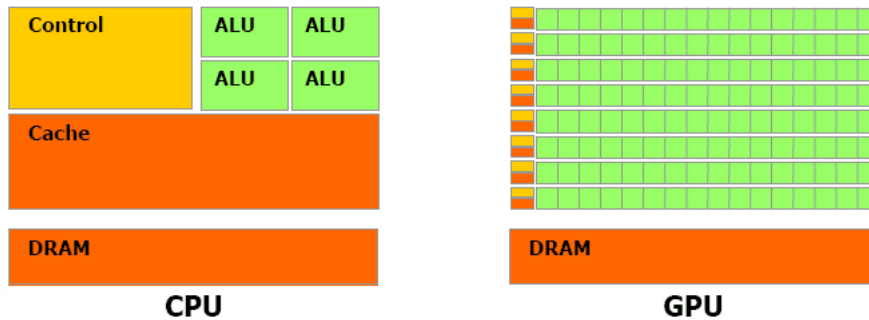


Figure 3.1: CPU and GPU Architectures

3.2 MATLAB with GPU

Using MATLAB for GPU computing, accelerates the applications with GPU more easily than compared to using C or C++ or FORTRAN. The MATLAB function is easy to write and achieves the

same in fewer lines of codes. The MATLAB function takes the input and transfers it to the GPU for the computation. For this, MATLAB provides the parallel computing toolbox which has the inbuilt functions to interact with GPU. The functionality to perform on the GPU can be written in a CUDA code and integrated with MATLAB. The MATLAB launches and run the Kernel, which is a GPU code and is performed on the GPU. For a Kernel to launch, two files are required, the .ptx file along with the .cu file. The programmer can specify the number of threads and blocks as per the requirement, but it should not exceed the maximum number of threads. The kernel is run using the *feval* function by passing all required parameters. Once the Kernel finishes executing the instructions, the result is collected and given back to the MATLAB function.

In order to get started with the problem, a prototype was developed which used the power of both MATLAB and GPU. A program to carry out convolution of an image was built in MATLAB, where the computation is done on the GPU. The filter was applied simultaneously over the image and we got convoluted image. Next, the difference between two images was calculated using Matlab with GPU. The two input images were the initial image and the deformed image. Both the images are taken as input, and transferred to GPU memory using *gpuArray* function. A Kernel is launched which finds the difference between respective pixels and stores it in a new array matrix.

3.2.1 Passing parameters to GPU

The MATLAB application makes use of the *gpuArray* datatype to transfer from host to device memory. Any parameter or variable which we want to use on the GPU must be created on the GPU or it should be transferred before using it. Once the variables are present on the device memory, the calculations are performed over the GPU. The results generated as the output should be transferred back on the host memory. This is done using the *gather* function available in MATLAB. These inbuilt functions in MATLAB provide easy way to carry out the computations over device memory.

Not all applications are suitable for GPU. One has to take care of the trade-off between the number of computations and transfer time. The transfer of variables to and from the device memory has cost. If the application which has less number of computations and more memory transfers, it performs bad on GPU than CPU as most of the time is consumed on transferring the data. The DIC application was made to perform on the GPU by passing the parameters on the device. As it needs more number of transfers, it turns out that it does not perform well on the GPU. Most of the time is consumed in the transfer.

Chapter 4

Conclusion

The process of DIC is performed on different architectures. It is a process which requires huge matrix operations. MATLAB is a chosen platform as it provides ease of working with matrix. But since it is a heavy process, it is not time efficient. For smaller sized images, it processes in comparison with Armadillo, but as the size increases Matlab takes relatively more time.

Using GPU in DIC did not help achieve speed-up as the number of transfers are more than actual computation on GPU. It increases the overall computation time of the program as most time is spent transferring variable to and from the device memory.

The Armadillo matrix library is light weight and fast. It is made to be similar to MATLAB and it has most of the MATLAB functions as inbuilt functions. As it is written on C++ architecture, it is more fast than MATLAB. The study made out of these three implementations shows that Armadillo performs DIC the fastest.

References

- [1] "Two dimensional convolution on the GPU using Cuda" by Dirk-Jan Kroon, Dec 2010.
- [2] "Exploring the Limits of GPUs With Parallel Graph Algorithms" by Frank Dehne, Kumanan Yogaratnam, September 2010
- [3] "GPU Computing with MATLAB" Webinar by Dan Doherty, MathWorks.
- [4] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, Vol. 1, pp. 26, 2016.
- [5] Suh, Jung W., and Youngmin Kim. *Accelerating MATLAB with GPU Computing: A Primer with Examples*. Newnes, 2013.
- [6] Singh, Amarjot, and S. N. Omkar. "Digital image correlation using GPU computing applied to biomechanics." *Biomed Sci Eng* 1 (2013): 1-10.