Threading libraries performance when applied to image acquisition and processing in a forensic application

Carlos Bermúdez

MSc. in Photonics, Universitat Politècnica de Catalunya, Barcelona, Spain
Student of MSc. in Free Software, Universitat Oberta de Catalunya, Barcelona, Spain
Student of Ph.D. in Optical Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: carlibp@uoc.edu

Cristina Cadevall

Ph.D. in Optical Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: cristina.cadevall@upc.edu

Abstract: Based on concerns during ballistics identification, a new system for ballistics image acquisition and data processing is proposed. Since image processing consists of high CPU load rates, a comparison of three different threading libraries is presented, concluding that parallel processing enhances ballistics identification speed.

Keywords: Optical Metrology, Forensics, Ballistics, Linescan, Multithreading

References and links

1. Churchman, J (1949). The reproduction of characteristics in signatures of Cooey rifles. RCMP Gazette (vol. 11, issue 5, pp. 133-140).
2. Smith, C.L, (2002) Linescan Imaging of Ballistics Projectile Markings for Identification. Proceedings of the International Carnahan Conference on Security Technology (pp. 216-222). Atlantic City, NJ: IEEE.
3. Puente, F. (2004) Automated comparison of firearm bullets. Forensic Science International (vol. 156, issue 1, pp. 40-50). Ireland: Elsevier.
4. Li, D.G (2008) Firearm Identification System Based on Ballistics Image Processing. Proceedings of the International Congress on Image and Signal Processing (pp. 149-154). Sanya, China: IEEE.
5. Rafael C. Gonzalez, Richard E. Woods (2002) Digital Image Processing, Second Edition, (vol. 7, pp. 519-566). Beijing: Publishing House of Electronics Industry.
6. Li, D. (2009). Ballistics Image Processing and Analysis for Firearm Identification. In: Chen, Y Image Processing. Intech. 141-174.
7. Akhter, S., & Roberts, J. (2006) Multi-core programming. US: Intel Press
8. Breshears, C. (2009) The art of Concurrency. US: O'Reilly
9. Karlsson, B. (2005) Beyond the C++ Standard Library: An introduction to Boost. US: Addison Wesley Professional
10. Reinders, J. (2007) Intel Threading Building Blocks. US: O'Reilly
11. Sanders, J. & Kandrot, E. (2010) CUDA by Example: An Introduction to General-Purpose GPU programming US: Addison Wesley Professional
12. Chitty, D.M. (2012) Fast parallel genetic programming: multi-core CPU versus many-core GPU. Soft Computing (vol. 16, issue 10, pp. 1795-1814). New York, US: Springer.Malacara, *Interferogram Analysis for Optical Testing*, Second Ed. (Marcel Decker, Inc., New York, 1998).

1. Introduction

Ballistics forensics is a criminalistics branch that studies firearms and projectiles involved in a crime scene. Apart from the cartridge, bullets provide enough conclusive information to identify the firearm that fired it [1]. Nevertheless, nowadays this task is performed manually by an operator by means of comparison microscopes and subjective judgment. In this study, it is presented in Chapter 2 a new automated acquisition system that performs bullet surface imaging automatically and compares it against a forensic database already filled with bullet surface images. This processing operation requires high-density processing so three different parallel processing solutions are tested and compared in Chapter 3, coming to a conclusion in Chapter 4.

2. Image Acquisition

Distinctive features appear in a bullet when is fired. Those features are like a fingerprint of the firearm that fired it. Said fingerprint is printed in the bullet surface as grooves

and lands. At every groove and land appear also small details related to the gun (Figure 1).

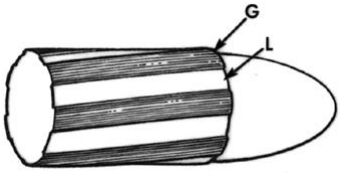


Figure 1 Firearm fingerprint in a bullet. G stands for Groove, L for Land. Source: US Dpt. of Justice

In order to obtain the bullet surface image or unrolled image, an optical microscope in combination with a rotational stage have to be used.

Microscope objectives depth of focus is usually not enough to obtain an area image of a cylindrical sample, particularly with high numerical apertures. Therefore, one way to obtain a well-focused bullet surface image is using a line-scan camera [2]. In this setup, bullet is rotating continuously while camera is acquiring lines at a constant but high frame rate. Said lines are finally stitched to form the unrolled image, containing surface information of one bullet section (Figure 2).

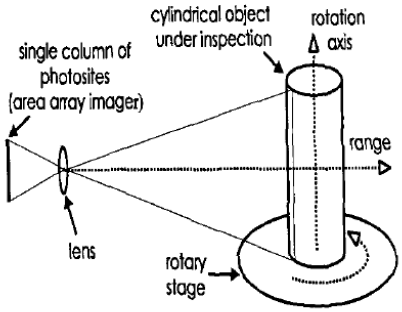


Figure 2 Linescan imaging of a rotating cilyndrical object

Nevertheless, Smith proposes using bright field illumination, which is not a good option to measure bright samples such as bullets because image contrast is poor. Taking image from different angles may solve that situation [3].

Based on the hardware arrangements seen on the literature, a system composed by an optical microscope and a rotational stage has been developed.

2.1. Acquisition system

Although all the previous contributions in terms of bullet rotation are based on a commercial single axis, their projectile holding principle might cause damage to the sample. A new fixation and positioning system has been designed and built in this project, which is composed mostly of stainless steel parts in order to not to contaminate the sample.

Image acquisition system is designed to achieve a lateral resolution up to 1µm at 4000fps, so drive and control loop repeatability has to be greater than 1µm at the bullet surface within this rotational speed, in our case 0.63' at 42.05º/s, with a cylinder diameter of 10.9mm.

In order to achieve said specifications, an optical system has been designed in such a way that, within a 10X magnification objective, image obtained at the sample surface has 1 micron of lateral resolution.

Since a 2D unrolled image has to be composed by a series of 1D images, a line-scan approach has to be used. Even so, instead of using a line-scan camera, one can prepare a CMOS camera to get images of a user selected region of interest (ROI), in our arrangement the central line.

Illumination system is a key factor in order to obtain high quality images, which is not straightforward when dealing with bright surfaces, like bullets are. Even though all the reviewed systems are provided with bright field illumination (Figure 3), it is often not the best setup since multiple reflections might appear on the bullet surface.

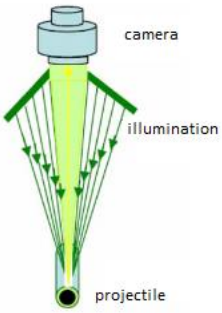


Figure 3 Bright field illumination system

In this project we introduced an illumination system that has not appeared before in the forensics literature, called epi-illumination.

Epi-illumination allows illuminating the sample on the same observation axis avoiding non-desired multiple reflections. Image quality improvements can be seen in Figure 4, where contrast is successfully enhanced on occluded zones (a Vs. b) and in flat areas (b Vs. d), since bright field illumination provides not enough light on those zones.

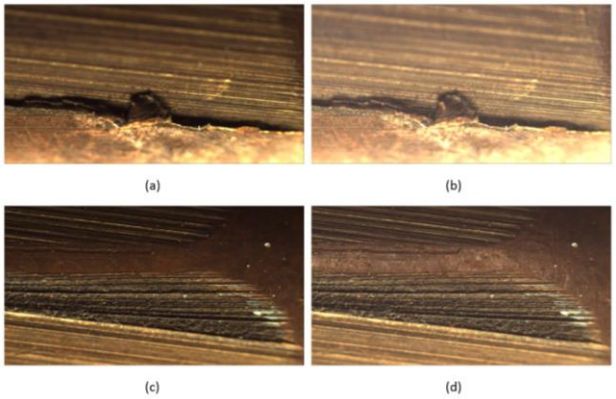


Figure 4 Bright field illumination (a,c) Vs. Epi-illumination (b,d)

2.2 Acquisition software

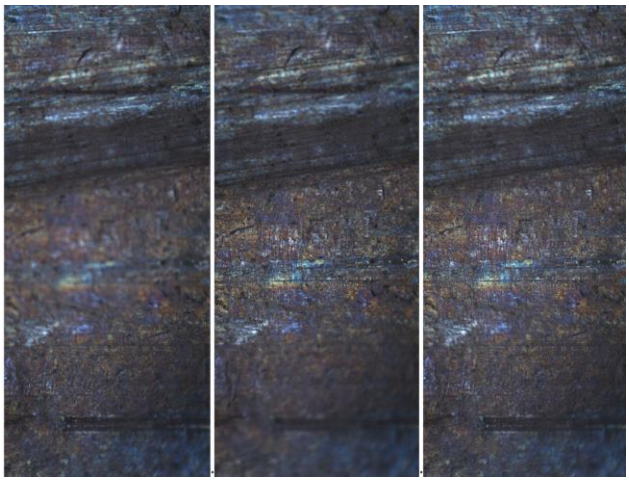
On the one hand, acquisition and processing software architecture consists of a multi-threaded application since hardware drivers are themselves one or more threads. An example is the camera controller thread, which puts images in a shared buffer. Then, acquisition algorithm that runs in another thread reads this buffer in a thread-safe way.

Image processing software can be executed in one or more different threads, depending on the application needs. This issue will be expanded in Chapter 3.

On the other hand, GUI has been developed as a separate executable in order to be able to escalate it in a distributed application. Basically consists of different threads that operate a communications manager, live image update and user event handling.

2.3. Preprocessing algorithm

In spite of having designed a rotational stage with enough precision to acquire quality images, fired bullets have not a completely cylindrical shape. Therefore, observation surface is being displaced in Z direction along the rotation movement, bringing the sample out of focus. For this reason, focused images are available at a certain Z planes. In this project, a set of images were acquired at different Z planes and then were fused by means of the discrete wavelet transform (DWT) giving priority to high frequencies (Figure 5).



(a) (b) (c)

Figure 5 Image preprocessing (a,b) acquired image, (c) fused image.

In the figure above, (a) and (b) are images acquired at two different Z planes, where each one has different focused and unfocused zones and (c) stands for the fused image by the DWT. Since two consecutive image acquisition may have had X and Y displacements, a registration algorithm has to be applied before fusion.

3. Image Processing

Once the system is able to acquire good quality images of the bullet surface that are useful for extracting information, a data abstraction can be performed in order to obtain a unique signature, which contains distinctive information. Another way to identify projectiles could be performed by means of a FFT-based analytical system [4], although an image preprocessing should be performed before in order to avoid noise and enhance contrast [5]. Li mentions the possibility that all the images previously acquired should be stored in a common-access database so every police department could compare a sample against said database [6].

Although FFT matching algorithms are the most common in ballistics identification, they require relatively high computing capabilities mostly due to high resolution images. In the proposed system, acquisition, preprocessing and identification threads are executed at the same time.

In a concurrent system, threads in a single hardware resource are processed interleaved in time [7]. For this reason, this application can be parallelized in different hardware resources, that is, in a multi-core processor.

3.1. Bullet identification process

As seen in the literature, bullet identification process consists of, basically, correlating an image in the frequency domain against a database full of previously acquired bullet images. This sort of image processing can be performed by means of arithmetic operations after applying the DFT (Discrete Fourier Transform), equivalent to a convolution. At the end, the cross-correlation spectrum is taken back to the spatial domain through the Inverse DFT (IDFT) (Figure 6). The value of the maximum peak stands for the image matching result.

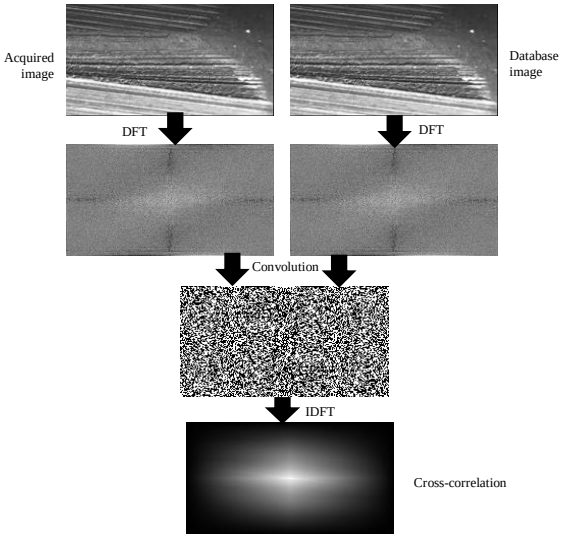


Figure 6 Image processing path

In this study, selected tool for DFT has been the FFTW library, being used in most image processing applications because of its simple integration but high performance and flexibility.

3.2. Threading libraries

Once concurrency has been detected (image acquisition, preprocessing and identification, they all can be processed in parallel) [8], paralleling tools have to be employed. Those tools are called Threading APIs (Application Programming Interface), which contain particular resources to enable a software with thread creation, parallelization, synchronization and shared resource protection. Those tools can enhance performance of a multi-threaded application in a multi-core CPU, where different threads are being executed concurrently.

Even though many threading libraries are currently available, three different ones have been compared in this study: Intel Threading Building Blocks, Boost and Nvidia CUDA.

On the one hand, Intel TBB is a template-based library at a loop level that, instead of managing threads at low level like other libraries do, uses an abstraction layer that process operation as tasks that are distributed dynamically between the different available CPUs [10]. This library provides thread-safe containers as an improvement to the C++ STL not thread-safe containers.

On the other hand, Boost is composed by 80 Free Software threading tools ranging from intelligent pointers and containers to RegEx and iterators [9], being a suitable complement to the C++ standard library. Boost allows the developer to manage threads at low level, which is very useful in high-performance applications.

CUDA (Compute Unified Device Architecture) is a development framework based in C for exploiting GPU (Graphic Processing Unit) [11] outstanding computing capabilities due to its many-core approach, meaningful for intensive applications like cryptography, genetic algorithms or image processing, as this project is [12].

3.3. Results

Since bullet identification is a time-consuming operation, the outcome of this threading libraries comparison is the time in image processing and matching.

A C++ program has been enabled with said three threading libraries and they all perform the same operation: after image processing, an image matching is performed against a database that was previously loaded with 50 images.

Said program was executed 20 times in an Intel core i7-3632QM CPU @ 2.2GHz with Windows 8 64-bit enabled with a Nvidia GeForce GT 640M .

Taking into account the number of CPU available cores, Intel TBB processing implements parallel_do instruction configured to use 8 threads. Boost created one thread for each loaded image (a total of 50 threads were created) through thread_group tool. Finally, CUDA was configured to use cufft tool (a FFTW implementation for GPU processing) and optimum grid and block sizes for arithmetic operations (images are divided into small units).

Each execution was comprised of image processing not only with cited three threading libraries but also with a serial, one single thread execution. Since processing time is in the order of milliseconds, it was measured through a C Time Library clock (Table 1, Figure 7). This repeatability test was performed with a 512x512 pixels image.

Threading Library	Whole Processing	Image Processing
-	543.99 ± 11.19	336.57 ± 6.63
TBB	323.13 ± 15.64	115.79 ± 12.86
Boost	235.61 ± 4.48	29.32 ± 3.33
CUDA	22.5 ± 5.58	3.54 ± 0.17

Table 1 Execution timings with standard deviation, time is measured in milliseconds

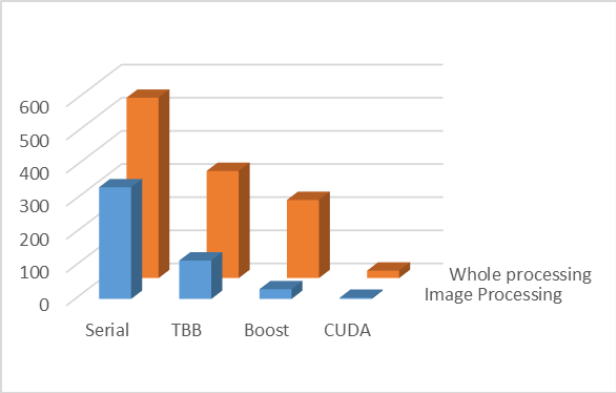


Figure 7 Average processing time (ms)

In the picture above can be seen the comparison of the average processing time when using a single-threaded algorithm against a multi-threaded application. Orange bars stand for average time of not only image processing but also image acquisition and database images loading into memory (in CUDA, loading data to the graphics card memory). Blue bars consist of average spent time by the correlation algorithm.

Obviously, operating system is also demanding CPU time or its own applications, so it may cause processing time variations along all the repeatability test. In all cases standard deviation was below 11% (Figure 8).

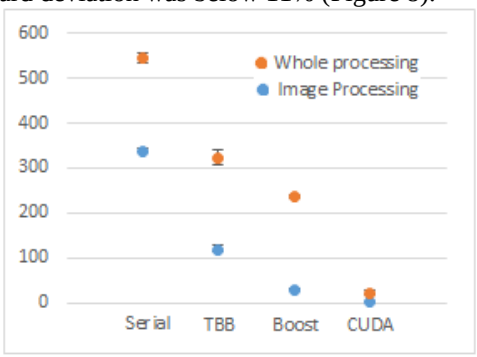


Figure 8 Box plot of average processing time for one single image (ms)

3.4. Discussion

In a 4 core CPU where 8 logical threads can be processed due to Hyper Threading, one may expect reducing serial processing time by 8. Depending on the application, it is not achieved because threads may compete for the same resource, such as cache memory or the ALU.

As a rule, processing time is not expected to decrease linearly with number of cores [7]. However, a multithreading application like this one will increase performance without software changes as new hardware platforms with more processing cores are appearing, while this will not happen with serial algorithms.

In this test, Intel TBB is reducing processing time almost by 2 as regards serial processing time. Results show how Boost library reduces image processing time by 10 although total processing time decreases almost by 3. This indicates clearly that most part of whole processing is spent in image loading rather than in the correlation algorithm.

CUDA, in a 384-core GPU @ 625Mhz has provided outstanding figures in that test, reducing by 25 the time spent by the serialized algorithm, showing image loading into the graphics card being performed much faster than in previous tests. Image processing time is reduced by 100 in respect to serial processing time, yielding the processing power capabilities when dealing with images.

Multi-threading confirms that performance increases in regard to a single-threaded application but not linearly with number of threads simultaneously executing, as Amdahl's Law points out [7]. GPU processing or many-core processing, in the other hand, could not be suitable for certain applications [12] but can enhance performance dramatically for image processing [11], as confirmed in this study.

4. Conclusions

In this paper, a new system for ballistics image acquisition and processing is proposed. On the one hand, in terms of hardware, we not only present a novel lighting arrangement and but also a new, high precision, rotational stage. This new system enabled us to acquire high-resolution and high-contrast projectile images.

On the other hand, as regards to software, an image processing necessity, which executes an identification algorithm, has been detected. It consists mostly in performing arithmetic operations in the frequency domain. In order to enhance image processing performance, said algorithms have been parallelized and compared through three different threading libraries.

The study demonstrates not only that parallel processing could be much faster than single-threaded applications but also that different libraries yield distinctive processing times, being CUDA the singular case that could execute matching algorithm 100 times much faster than the serialized application.