Brno University of Technology

Faculty of Information Technology

Hardware Acceleration of Graphics and Vision Algorithms

A Compilation of Selected Works Published by

Adam Herout, Ph.D.

Brno 2009

Contents

Introduction

The purpose of this document is to summarize the recent research done by Adam Herout, Ph.D. to compose a material for habilitation to the position of associate professor at the Faculty of Information Technology, Brno University of Technology. The form of this text is a commented compilation of important research papers authored or co-authored by me.

The regulations of the Brno University of Technology accept two forms of this habilitation text: commented compilation as this one or an original text containing new, previously unpublished research material. This summary contains three articles submitted during this year to respected journals (Herout and Hradiš 2009), (Herout and Havel 2009), (Herout, Jošth and Juránek, et al. 2009) so technically this text contains previously unpublished research results that – as reported by experiments described in the articles – improve the state-of-the-art significantly. However, I decided for the form of commented compilation of articles for two main reasons. Firstly, when I was making a survey of my publications, they seemed to excess one narrow field, but their span across several fields, yet keeping a common topic, describes my research interests precisely. Thus, describing the very recent findings would not be by far complete in characterization of my research focus and achievements. The second reason is that my research work is co-authored and importantly contributed by a number of my colleagues and I want to acknowledge and recapitulate this fact by listing the papers authored by a set of people in this summarization text. The situation is determined for me by being responsible for the Graph@FIT research group (since January, 2008). I would like to take this opportunity and express my sincere thanks to my colleagues for the cooperation on the research and for their contribution to the work done by the research group.

The one research topic common to all the work summarized in this text, which has become my expertise, is hardware acceleration, focused on acceleration of graphics and vision algorithms. These families of algorithms are characterized by their computational complexity and – what makes them exclusive in the family of demanding algorithms – by a common need to run the algorithms in real time and their relative structural simplicity. In the field of graphics rendering, offline usages, when the time for synthesis of one frame is not limited by the display frame-rate, exist and are useful, but when the rendering is possible in real-time and with low latency, the application potential of any algorithm is multiplied and applications of higher quality level are enabled. The situation in image-processing and computer vision algorithms is very similar and the demands there are yet increasing in recent years, when cameras are becoming exclusively digital and when their prices drop. This obviously calls for new and faster algorithms, but also for solutions that would implement existing algorithms efficiently using special hardware.

This fact can be illustrated on the history of graphics chips developed for real-time rendering. Their complexity (number of transistors on chip), rough computational power and memory bandwidth exceeds the parameters of the best central processors, and the interest in GP-GPU (General Purpose GPU computing), i.e. using the graphics chips for tasks that are not related to graphics or rendering, is enormous. Graphics chips are not used only for rendering graphics or for computer vision (an example is in section III.5 of this document), but also for computation of physics, Finite Elements Methods computation, and different other fields.

This document is divided into three main chapters: Hardware Acceleration of Graphics Rendering, Hardware Acceleration of Hologram Synthesis, and Accelerated Object Detection in Images and Video Sequences. Each of the chapters constitutes a family of algorithms that has been studied by me and includes a number of articles which best describe my contribution to the research field. The last chapter is the most recent one and contains the highest number of articles. In the field of accelerated object detection, the Graph@FIT research group contributed by designing novel low-level image features suitable for fast processing called the *Local Rank Functions*, by proposing the concept of *image pattern inert to object detection* which is more or less an implementation hint that may speed up object detection in some scenarios, and proposed a completely new concept called the *Early non-Maxima Suppression* suitable for several common computer vision tasks.

I. Hardware Acceleration of Graphics Rendering

Graphics rendering has a long history of hardware support and acceleration. First graphics acceleration chips are dated back to 1970's, in 1980's they moved from the domain of supercomputers to commodity hardware available to everyday computer users and stepped into the realm of entertainment computing. Real-time (or nearly real-time) graphics rendered at that time was generally based on polygonal models with low complexity and on rasterization as the rendering algorithm. Rasterization of polygonal meshes in object-order manner is currently the dominant paradigm in hardware acceleration of graphics rendering. Graphics processing units (GPU) today outperform the common central processing units (CPU) by an order of magnitude, and while central processors grow according to the Moor's law, the advancement achieved in graphics chips is described by some as "Moor's law squared". See Figure 1 for an illustration of the development of the computational power of graphics chips compared to central processors.

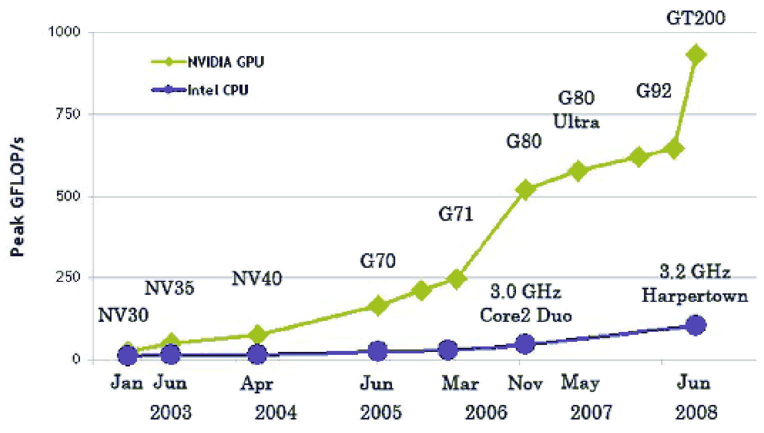

Figure 1. Rough computational power of GPU vs. CPU in time (graph is originated from nVidia marketing material)

The architecture of graphics processing units based on the common paradigm of object-order rasterization of polygonal meshes has some characteristics different from the commonly used and studied central processing units, which allows their apparent or real superiority in the computational power. GPU's rely on high degree of parallelism. While current CPU's contain 4 cores (with hyperthreading 8 virtual cores), commercially available and affordable nVidia GTX 280 chip contains 240 processor cores organized into multiprocessors of 8 cores. This high degree of parallelism is possible thanks to the nature of the rasterization problem – primitives of the polygonal mesh are processed relatively independently and the fragments (pixels) of the primitives likewise. The cores within the (streaming) multiprocessors operate simultaneously in a SIMD (single instruction – multiple data) manner – the multiprocessor has only one control block and the cores execute the same operation. That is possible because the fragments (or primitives) processed by the processor are relatively independent on each other, but typically share the same sequence of operations and branching and conditional execution is rather rare. In this manner, a large portion of the chip area (number of transistors) of the graphics processors is used by ALU's (Arithmetic Logic Unit) which actually perform calculations, achieving the impressive computational power (see Figure 1). In contrast, a major portion of the CPU's area is dedicated to caches and registers.

The technology of the mainstream graphics processing units is a dynamically developing field. It is expanding quantitatively, by increasing the number of computational cores, memory bandwidths etc. Besides that, qualitative changes of the paradigm can be also observed. An important direction of development is ability to process and generate the geometric meshes in the graphics processor. The traditional and still common approach is that the central processor manages the polygonal mesh of the models in the virtual scene and it controls the graphics processor in the way it displays the mesh. Changes to the topology of the mesh need to be done in the central processor and the mesh or its parts re-sent to the graphics memory. In contrast, a current trend is that the graphics chips contain "geometry shaders", which can modify the geometry of the mesh and perform animation, tessellation, simplification and other important operations right in the graphics processor itself, allowing speed-ups of these operations and especially sparing communication bandwidth between the central processor and the graphics chip.

Another trend in the field is called "General-Purpose computing on Graphics Processing Units" (GP-GPU). It is a logical consequence of the graphics chips exhibiting the high computational power and their streaming and highly parallel architecture – the power of the chips can be used for other tasks than graphics rendering, i.e. in a general-purpose manner. First, this movement used dedicated graphics languages (Cg, GLSL, HLSL, ...) to describe common computation problems, but the developments goes towards support of general purpose languages (C and its variants) on the graphics hardware – the main representatives of this approach is the Compute Unified Device Architecture (CUDA) and currently OpenCL.

However the mainstream approach of the GPU is dominant in its achievements and commercially, some important alternatives exist. Some researchers accelerate algorithms based on ray and photon tracing: either by optimization and use of SIMD functionality of the common central processors (SSE and other instruction sets) (I. Wald 2004) or by designing specialized circuitry for FPGA (Field-Programmable Gate Arrays) or ASIC (Application Specific Integrated Circuitry) (Schmittler, Wald and Slusallek, SaarCOR: a hardware architecture for ray tracing 2002) (Schmittler, Woop, et al. 2004).

Another paradigm for representing scene objects and for graphics rendering is point-based computer graphics. The next section introduces one publication that represents a set of publications related to my Ph.D. thesis and the following research. This research activity of mine and my colleagues seems to be a pioneering work in the field that was followed by other researchers. Section I.2 of this compilation is dedicated to an article in a valued visualization and graphics journal; it presents an algorithm for fast computation of ray-triangle intersection. Ray-triangle intersection is a very basic geometric problem and is a key component in many graphics rendering algorithms; its speed highly determines the speed of the whole application. Research in this area has a long history and attracted attention of many. The algorithm presented in the article

I.2 seems to be today's fastest solution in general conditions and thus constitutes an interesting achievement.

I.1 Hardware Pipeline for Rendering Points as Scene Primitives

Using points as rendering primitives was pioneered by Marc Levoy and Turner Whitted in 1985 and revisited by Grossman and Dally (1998), whose work brought this concept to common attention of researchers in the field of computer graphics. Point-based rendering was then approached from different angles in software solutions (Zwicker, et al. 2001) (Pfister, et al. 2000). Several solutions exist also for graphics chips mentioned above, for example (Rusinkiewicz and Levoy 2000) (Botsch and Kobbelt 2003).

My Ph.D. thesis in 2004 under supervision of Pavel Zemčík and the related publications (Herout and Zemčík, Hardware Pipeline for Rendering Clouds of Circular Points 2005), (Herout and Zemčík, Animated Particle Rendering in DSP and FPGA 2004), (Herout and Zemčík, Hardware Implementation of EWA 2004), (Zemčík, Herout and Crha, et al. 2004), (Zemčík, Herout and Tišnovský, Particle Rendering Pipeline 2003) was one of the first solutions of point-based rendering targeted to hardware circuitry: field-programmable gate arrays (FPGA) and application specific integrated circuits (ASIC). The work has been cited by researchers in the field of computer graphics and some FPGA-based point-based rendering systems seem to be inspired by it (Beeckler and Gross 2008) (Weyrich and others 2007).

The following paper presents the final version of the rendering platform's prototype. Its core part is implemented in VHDL targeted for FPGA, namely Xilinx Virtex II. It implements a frame-buffer with depth-buffer and hardware accelerated depth-test. The only supported primitives are ellipses of limited size – projections of circular points of proper radius. Other researchers introduced algorithms for processing and real-time rendering of point clouds adjusting the level-of-detail so that the primitive rendered in the screen space has proper dimensions in units of pixels (Rusinkiewicz and Levoy 2000).

Note that some researchers (Botsch and Kobbelt 2003) use the powerful GPU chips to emulate rendering of elliptical points by applying a texture of circular shape on a polygonal primitive (triangle). The approach suggested by the following paper (and in detail by my PhD thesis) which directly renders the elliptical primitives uses the hardware circuitry much more efficiently. That is, for rendering one primitive at given speed, the GPU needs more transistors and higher clock frequency. However, the GPU's are very powerful compared to the FPGA used in the prototype, so the rendering speed achievable using GPU will be higher compared to the simple prototype reported by the paper.

A complete frame-buffer for a normal screen resolution would hardly fit into the FPGA used by the prototype, or would consume all the chip's resources. To avoid this, the algorithm only uses a horizontal stripe of the target frame-buffer at a time. The stripe slides vertically and the pre-processed rendering primitives are sorted in a memory buffer according to their vertical coordinate. This arrangement resulted in a very compact rendering engine that can fit several times into a relatively small and cheap

FPGA, showing a possible way for embedded designs of low power consumption, low heat emission, low price etc.

I.2 Fast Ray-Triangle Intersection

Ray-triangle intersection is a very common and basic algorithm of computational geometry and of computer graphics. Ray tracing and methods derived from it is a frequently used method for realistic graphics and today's standard means of describing a three-dimensional scene is by using a mesh of triangles. Also the real-time graphics based on graphics chips, object order rendering and rasterization (in contrast to ray-casting) uses this geometric calculation: for physics simulation, visibility testing, collision detection and other frequent tasks. Calls to a routine that calculates ray-triangle intersection are therefore numerous in today's graphics systems; for the performance of the application, the speed of ray-triangle intersection computation is crucial.

One very popular solution to this problem was introduced by Tomas Möller and Ben Trumbore (Möller and Trumbore 1997). Its advantage is that the pre-computed data for one triangle are minimalistic and the calculation is fast. One algorithm based on similar principles was later proposed by Kensler and Shirley (Kensler and Shirley 2006). An important improvement over these algorithms was introduced by Ingo Wald (I. Wald 2004) in his dissertation thesis. He designed a real-time ray tracing system and highly optimized this frequent operation of ray-triangle intersection for common central processors including their SIMD instruction set MMX and SSE. In his solution, a larger amount of data was pre-computed and stored in the memory, and for cache performance, the data was padded with unused alignment bytes to a "round" amount of 16 bytes (by adding 4 bytes). A recent improvement was achieved by Shevtsov and his colleagues (Shevtsov, Soupikov and Kapustin 2007) who designed a variant of the algorithm for recent central processors with support of new versions of the SSE instruction set.

Wald's and Shevtsov's solutions share a common idea that the ray-triangle intersection is not calculated on one ray and one triangle, but instead, a *packet* of coherent rays is shot against the scene and a number (4, 8, 16) of intersections is calculated for a corresponding number of rays. Based on the result of the ray-triangle intersection, the packet of rays may be split to those that hit the primitive (triangle) and those that need to be further tested. Obviously, the splitting is costly and complicates the process and hence the requirement for coherence of the rays. Luckily, the coherence is relatively simple to achieve in typical ray-casting and ray-tracing scenarios. For computing intersection of a packet of rays and a triangle, the SIMD instructions can be easily used, speeding up the whole process several times. One drawback shared by both the solutions (of Wald and Shevtsov) is that the pre-computed triangle's data has a different structure when intersecting with a single ray and when intersecting with a ray packet. The application then has to decide in advance which data to prepare, or both versions have to be prepared and stored, which is costly. The reality shows that intersecting against packets of rays pays back, but as the packets split (by the result of the intersection operations and by traversal of the scene), switching to intersection of single rays against the triangular primitives would be beneficial, but is not possible, due to the different data structure.

Jiří Havel in his MSc. thesis under my supervision compared different ray-triangle intersection algorithms and designed a modification that in a sense merges ideas from different solutions mentioned above. This modification was further developed and tested and resulted in the following article published in IEEE Transactions on Visualization and Computer Graphics (IF2008: 2.445) (Havel and Herout 2009). The experiments described in the article are performed identically as in the Kensler-Shirley paper (Kensler and Shirley 2006) to avoid any bias from us as the authors. Their results report that for packets of rays, the new solution is about as fast as the contemporary best solutions, but for single rays, the state-of-the-art is outperformed by 10 to 25 percent. More importantly, the method uses identical pre-computed data structures for both single rays and for packets of coherent rays, so it allows the application to switch between these according to its needs. The new algorithm is designed to utilize the SIMD instruction set SSE4, but the speed-up is reported also in a plain C implementation, so the solution is not a platform-specific optimization, but an improved algorithm indeed. The cost of using the algorithm is that it requires slightly more pre-calculated data for each triangle – exactly 16 bytes. However, this seems acceptable for today's real-time ray tracing systems and other applications potentially using a ray-triangle intersection algorithm, because they aligned the pre-computed data to 16 bytes anyway. The new algorithm therefore seems to only use memory that was left intentionally unused before.

II. Hardware Acceleration of Hologram Synthesis

Holograms offer a possibility of full 3D display of scenes because they contain all the visual information that passes through the hologram plane in all directions. Because the mathematical model of holography is well studied and exactly described, it is possible to synthesize digital holograms for virtual scenes, digitize traditional chemical holograms, emulate viewing of digital representations of holograms, work with holographic emulations and other operations.

Holograms can be described as recordings of optical field, or more precisely as recordings of interference of optical field with a planar light wave (Goodman 2005). While digital synthesis of optical fields for virtual scenes is possible, it is very computationally demanding. The resolution of usable optical fields is extremely high (the smallest hologram considerable for practical viewing is about 10,000×10,000 pixels). Furthermore, for proper optical field synthesis, the visibility in the scene must be evaluated for each of the field's pixels (i.e. the scene rendered from the pixel's location) and contributions of all visible scene primitives must be accumulated. As explained in the paper, this leads to complexities of O(n^4) where n is the optical field's dimension in pixels. Moreover, the scene primitives cannot be easily planar or non-planar surfaces but commonly, point-based representations of the scene are used (details are discussed in the following article); the smallest numbers of points in realistic scenes is in the order of millions.

Speeding up this extremely demanding process (synthesis of a hologram by commonly used methods can take hundreds of hours of computational time) is obviously highly desirable. There are several approaches to speeding up the process: some reduce the number of calculations by allowing an acceptable error, some transfer important parts of the computation to frequency domain and use Fast Fourier Transform, some ignore the visibility problem in the scene and some use specialized circuitry for hardware acceleration (see the article for a brief summary of previous approaches to solving the problem).

The following article presents an acceleration design for FPGA (Field-Programmable Gate Arrays), which speeds up hologram synthesis to rates which – in the future – may lead to real-time synthesis of (low-resolution) holographic images of simple scenes.

II.1 Hologram Synthesis Accelerated in Field-Programmable Gate Arrays by Partial Quadratic Interpolation

The following paper (Hanák, et al. 2009) is targeted to acceleration of methods of hologram synthesis that describe the scene as a set of point light sources. Visibility of the light sources can be partially or fully calculated as in the case of the reduced occlusion method (Janda, Hanák and Onural 2008). Each point light source contributes to all (not occluded) pixels of the resulting hologram with a varying phase and intensity of its complex contribution. The solution proposed by the paper evaluates the contributions to a part of the hologram's scan-line by using quadratic interpolation of the contribution's phase.

Compared to previous solutions, e.g. (Ito, et al. 2005), this paper seems to reach better efficiency of the FPGA resources. The functional prototype was tested on Xilinx Virtex II-250 FPGA chip at 100 MHz and its one evaluation engine was able to process 100 million samples per second (one sample is one contribution of one point light source). The solution scales very well, so on larger FPGA chips the evaluation engine could be easily repeated many times (hundreds of instances for large contemporary FPGA's). Thanks to its good use of the chip resources, it outperforms the previous solutions in speed on a given FPGA chip. Besides that, this solution calculates with varying intensity of the light sources and therefore compared to most of the other solutions it can reflect material properties of the scene to some extent, not only the shape of uniformly lighted objects.

III. Accelerated Object Detection in Images and Video Sequences

Object detection is an important task in computer vision. There are several vital approaches to solving this problem and it attracts attention of many researchers and research groups. One important and probably dominant class of approaches to object detection is usage of pattern recognition by machine learning, where the objects to be looked for are described by a sufficient set of examples of the object and a set of counter-examples. While the effort for development of a general object detector that detects any class of objects given by the training sets is meaningful and vital, also specialized detectors exist whose construction is tuned for a particular class of objects sharing common characteristics (e.g. detection of writing/letters, detection of human faces, detection of particular products in industrial quality control, etc.).

A common approach to object detection is usage of statistical classifiers, which classify an image window (a sub-image at a given position of appropriate dimensions) resulting in a binary decision: the window contains the object of interest or not. One important breakthrough in this approach was achieved by Viola and Jones (Viola and Jones 2001), who used AdaBoost (Adaptive Boosting) (Freund and Schapire 1995) in combination with very efficient low-level image features and with a focus-of-attention mechanism called a cascade of classifiers.

The interactive frame-rates of the object detector (detector of human faces in the particular case) were achieved thanks all the three mentioned factors. The focus-of-attention cascade allowed the detector to spend very little effort on most locations of an average image, because a majority of locations were rejected already by the first simple classifier in the cascade. This focus on rejection of non-faces (non-objects in general) is a key characteristic of the algorithm and of a variety of algorithms following it. The fast low-level image features based on haar wavelets and their extraction using an integral representation of the original image allowed fast processing and detection in multiple scales with not image resampling or other additional cost. Usage of adaptive boosting helps the algorithm extract a minimal number of important low-level features, and in synergism with the classifier cascade the algorithm does not extract the complete feature-vector commonly used in pattern recognition by machine learning, but the features are extracted gradually, according to the information need.

My research interests in this field include both improvements to the training and evaluation phases of the detection and localization process (Šilhavá, a další 2007) and especially hardware (and software) acceleration of the evaluation algorithm to improve its speed and to allow its operation in embedded and other special environments (the following papers).

An important achievement in this field by the Graph@FIT research group is definition of new low-level image feature set called Local Rank Functions (LRF) which is suitable for fast implementation of various image processing and pattern recognition tasks. This feature set was originally defined for efficient FPGA implementation (or in hardware circuitry – section III.1 of this text), but later in experiments it prove itself useful in the context of the SIMD instruction sets in the common processors (III.2) and when implemented on the graphics chips in the GP-GPU sense (as mentioned in chapter I of this text) – III.3 and III.4. An overall description and evaluation of the Local Rank Functions low-level image feature set was written on request as a chapter in a monograph on Pattern Recognition (III.5).

The research done by me and my colleagues in this field led to some improvements of the training and detection process. These achievements are represented here in this collection of publication by two articles, submitted to journals according to their focus; the impact of the improvements was verified by experiments and in one case seems to improve over the state of the art by speeding up almost twice the currently best solution with the same error rates III.7. The other article (III.6) on the other hand improves the speed of the object detector only in the order of percents, but the Inert Pattern defined there is easily usable in different contexts and it does not require any algorithmic change to the object detector (or other higher-level algorithm that uses it) – thus making an improvement without any implementation costs.

III.1 Acceleration of AdaBoost-Based Object Detector Using Field-Programmable Gate Arrays

This paper (Granát, et al. 2007) presents one of the first efforts to accelerate object detection by AdaBoost or more precisely WaldBoost (Šochman and Matas 2005) classifier in hardware. This research showed that the commonly used haar wavelets calculated on integral image (Viola and Jones 2001) are relatively costly when evaluated in hardware. The reason is that the evaluation of one typical haar wavelet requires 6 or more random memory accesses. The subsequent computation is then very simple (addition or subtraction of the previously fetched values). This is exactly opposite to what is usually desirable for acceleration on FPGA: small data throughput, sequential access to the data, complex calculations. Besides that, the haar wavelets as low-level image features require local energy normalization in the image, which leads to further memory fetches for each detected window and a relatively costly preparatory phase.

Considerations of the disadvantages of the commonly used low-level image features lead to designing a new set of low-level features, which would be more suitable for evaluation in hardware (FPGA or ASIC). The main criteria for the image features' design were:

- low number of memory fetches
- memory fetches preferably aligned or regular to allow simultaneous fetch of several values by using a mechanism of separately implemented memory banks
- no need for energy normalization
- good descriptive power, i.e. good performance of the classifier constructed upon such feature set

Several generations of feature set designs experimented with resulted in Local Rank Functions – a new low-level feature set specially designed for implementation in hardware. A detailed description of the feature set can be found in (Herout, Zemčík a Hradiš, a další 2009) in section III.5 of this document.

The solution described in the paper uses an FPGA containing the main object-detection engine, coupled with a DSP which provides the input/output operations, controls the FPGA, ensures the data flow (feeding the image into the FPGA, fetching the results out), and performs post-processing tasks, mainly non-maxima suppression in the resulting detections found by the FPGA engine.

The achieved design achieves performance that allows real-time detection in images of moderate resolution. The FPGA used in the experiments (Xilinx Virtex II-250) is rather a small one and one detection unit only uses 35% of its slices (Xilinx Virtex II-1000 could accommodate up to 5 identical units). At the same time the FPGA has very low power consumption and heat emission, so this research effort provides an ideal solution for embedded systems, low-power and industrial solutions.

Note that this paper does not use the LRF feature set or its special cases. The acceleration engine based upon these has been created and tested, its results are mentioned in section III.5, but it has not been published intentionally, because we have submitted the design for intellectual property protection and the process still continues.

III.2 Object Detection Accelerated Using the SSE Instruction Set

As mentioned in the previous section (III.1), the Local Rank Functions low-level image feature set was designed to be suitable for implementation in hardware circuitry. However, when experimenting with the feature set it prove to have good descriptive power (i.e. it extracts important information for the classifier constructed upon it) but also it appears efficiently implementable in other environments than programmable hardware.

This paper (Herout, Zemčík and Juránek, et al. 2008) presents an efficient implementation of the Local Rank Differences (a special case of the Local Rank Functions) low-level image feature set on common CPU, namely using its SSE instruction set.

The Local Rank Functions in the particular form used in the paper are characterized by extracting a small grid (3×3) of convolved image values and performing operations on these values. The paper proposes a mechanism of processing these values using the SIMD instruction set so that as many operations are performed in parallel as possible. A very important characteristic of the implementation is the memory layout of the pre-processed (pre-convolved) image. This memory layout ensures that any grid of 3×3 values that compose the response of the feature can be read in two memory reads. The memory layout is very important for the speed performance of the feature set, which outperforms the commonly used haar wavelets notably. Besides requiring only two memory accesses (compared to approximately 6 in the case of haar wavelets), the LRF features still do not require any image normalization, which simplifies the preparatory phase of the classification.

III.3 Object Detection Accelerated using Graphics Processing Units

Similarly to the previous section (III.2) the following paper (Polok, et al. 2008) represents a series of experiments carried out with the LRF low-level image feature set, exploring their ability to be implemented efficiently on the graphics chips (GPU's). This paper describes an implementation on the GPU done in the "traditional" way by using the shading languages and by "rendering" primitives to invoke suitable number of fragment shader instances on different locations of the image. The programming in this way is not as straight-forward as in case of for example C language programming.

As in the previous section, this implementation relies on two main properties of the LRF image features: parallel processing of the 9 extracted grid values, and regular memory layout which speeds up the memory accesses.

III.4 Object Detection Accelerated Using the Compute Uniform Device Architecture (CUDA)

The computational power of the graphics chips is attractive for researches in fields not directly related to graphics rendering, but the traditional means of programming graphics chips are targeted to graphics. CUDA is one of the first viable languages for programming graphics chips for general purposes. In fact, CUDA does not define a new language, but a slight modification of the C programming language which allows distinguishing the code targeted for the host processor and the code for the graphics device and to define the interoperability of the fragments of code.

The following paper presents an implementation of real-time object detector based on WaldBoost (Šochman and Matas 2005) and Local Rank Functions low-level image feature set running on CUDA. This paper is in the review process in the Journal of Real-Time Image Processing, so technically, it contains new unpublished information. The experiments report 3-4× speed-up over the efficient CPU implementation (III.3), which means that the object detection – even in high-resolution videos – can be performed several times faster than what is required for real-time detection.

One important advantage of these implementations running on graphics chips is that the object detection solution can be easily scaled up by adding more graphics cards (today's commodity computers support three advanced graphics cards at a time normally) or by using clusters of graphics boards developed by nVidia under the name Tesla.

This paper extends over previously published work (Herout, Jošth and Zemčík, et al. 2008) which focused on the implementation and performance evaluation of the Local Rank Differences feature set itself. This new paper adds the logic of the whole classifier and deals with issues caused by the (otherwise desirable) WaldBoost property that different locations of the input image are processed by different numbers of classifier stages.

Our current steps that continue in this research lead towards an implementation in the OpenCL programming environment. OpenCL is a GP-GPU solution similar in some aspects to CUDA, but it has a more ambitious design and it should be portable to different GPU processors and other programmable chips. The public functional version of OpenCL is very recent (this year), so the platform is rather immature. However, its future seems promising and we are going to explore the possibilities of using this platform for object detection and other computer vision tasks.

III.5 Local Rank Features – The Low-Level Features for Fast Object Detection

Based on the previously mentioned papers about Local Rank Functions and their efficient implementations I with my colleagues were invited to write a chapter to a monograph on novel trends in pattern recognition (Herout, Zemčík and Hradiš, et al. 2009). This book chapter constitutes a good summary of the Local Rank Functions low-level image feature set, defines its special cases Local Rank Patterns and Local Rank Differences, evaluates the features' classification performance and discusses the possibilities of implementations on different platforms.

III.6 Image Pattern Inert to Object Detection

When working on the fast implementation of the object detector on CUDA, one interesting improvement of object detection occurred to us. On CUDA (and also in the GPU programmed by shaders and in other different practical situations), it seems more beneficial to detect in one image rather than in several images of different resolution. The overhead of constructing a "pyramidal image" containing all the necessary levels of the multi-resolution pyramid seems smaller than the communication overhead related to processing several images of different resolutions. The pyramidal image does not contain only the useful data of the images, but also some small fraction of "unused" pixels between the useful image data.

The object detectors used in our implementations are constructed as cascades of simpler classifiers that result either in rejecting the sample as non-object or in continuing the detection process. Because of this focus-of-attention mechanism, the samples positioned on the unused pixels were rejected very quickly, compared to average samples in the useful image data.

However, we experimented with different filling patterns that fill the unused areas of the pyramidal image to see if they influence the detection performance. The experiments carried out and their results are described and discussed in the following article (Herout and Havel 2009) that has been submitted to Pattern Recognition Letters and is still under review. The conclusions made in the paper are that different classifiers (although trained similarly) prefer slightly different filling patterns – named Inert Patters in the article. For one given classifier, one pattern can be selected, that "suites" the classifier and performs better than the rest of the candidate patterns. Interesting is, that the black color originally used is in most (or all) cases outperformed by other patterns.

The speed-up achieved by using the inert pattern is not high – it is several percent on the "in-between" pixels, and they constitute a fraction of the whole image. However, using the pattern does not require any algorithmic change to the detector – it is a part of the preparatory phase. The unused areas of the image have to be cleaned in some way in any case and cleaning them with the pattern – on today's graphics hardware – makes no additional computational cost. Therefore, some small speed-up is achieved without adding any extra computational load or without changing the detector. Besides that, the concept of inert pattern is usable with different detection principles – it is not limited to any particular kind of object detector.

III.7 EnMS: Early non-Maxima Suppression

A common scheme in object detection is to evaluate a classifier on all locations in the processed image and then to post-process the positively classified locations by a non-maxima suppression algorithm. This approach is motivated by an observation that at the location of the searched object, several neighboring image windows are classified positively, but only one of them needs to be proclaimed as the detected object. Different non-maxima suppression algorithms exist, but typically, the image window with the highest response of the classifier is selected.

The classifiers used for object detection classify different window locations independently on each other. The following article (Herout and Hradiš 2009) submitted for review into the International Journal of Computer Vision presents a new algorithm which combines the two phases of the detection process – classification and non-maxima suppression. This approach allows combining intermediate results of the classifiers with a focus-of-attention mechanism to gain additional information that was left unused by previous approaches.

The algorithm is based on Wald's sequential probability ratio test (A. Wald 1945) that offers a sequential decision strategy, which optimally discerns whether a binary problem can be decided at a given stage, or whether more information needs to be acquired. Our paper proposes a generalization of this sequential strategy named conditioned sequential probability ratio test (CSPRT) – the decisions are conditioned by additional information, in our case extracted from the mutually shared information between the samples of the scanned image.

Experiments carried out and reported in the paper show that the extra information gained allows speeding up the object detection – or more precisely object localization – problem approximately twice compared to the state of the art (see section 6 of the article). Speeding up the highly optimal algorithms used for object detection today twice should be considered an important success. It shows that sharing the intermediate classifiers' information between different samples in one image adds valuable knowledge to the object localization process.

This article opens a wide space for further research. Object localization, tracking, recognition based on image parts and other algorithms can obtain a new category of algorithms. For further research, we also consider other information that may still be left unused because the samples of the image are evaluated apart from each other.

Summary

The research articles in this document are sorted into three topics: Hardware Acceleration of Graphics Rendering, Hardware Acceleration of Hologram Synthesis and Accelerated Object Detection in Images and Video Sequences. The common denominator of the recent research presented here is hardware acceleration of graphics and vision algorithms, which is the research interest of the author. Three of the articles were at the time of printing this summary in the review process in two computer vision or pattern recognition journals – these contain previously unpublished material, which, as verified by experiments, improves over the state-of-the-art notably.

In the field of hardware acceleration of graphics rendering, one article represents several years of research leading into definition and implementation of a hardware platform (using FPGA) for rendering circular points composing point-based three dimensional scenes. This research has been cited and built upon by authors respected in the field of point-based graphics. The second article in the chapter about graphics rendering presented a ray-triangle intersection algorithm, which – at the moment – seems to be the best performing algorithm for the given task on common processors (Intel CPU) and thus it means an important achievement in the domain.

One article is dedicated to hardware acceleration of digital hologram synthesis. This article has been published in a respected optics journal and it leads towards real-time hologram synthesis in the future. This research work was a result of cooperation within a research project targeted to collaboration of important Czech graphics research groups.

The most recent and wide area of research presented here deals with fast object detection in image and video sequences. An important result in this area achieved by the author of this compilation and by his colleagues is definition of the Local Rank Functions low-level image feature set. This feature set is suitable for hardware implementation, but as discovered by experiments, it also performs well when implemented in conventional software platforms. Several articles from chapter III of this compilation are dedicated to fast implementations of the feature extraction and object detection based on it. Besides these, two articles contain novel improvements to the training and detection processes based on statistical classifiers, one of them achieving notable improvement to the state-of-the-art.

Current research of the author and of the Graph@FIT research group primarily extends the knowledge described in chapter III. It focuses on implementation of the Local Rank Functions on other platforms, implementation of other low-level feature extractors, and construction of efficient object detectors and localizers. Based on such fast object detectors, the research will include higher-level methods in video processing that harness fast object detectors and whose applications may be enabled by the detectors' real-time or super-real-time performance.

References

Beeckler, J.S., a W.J. Gross. „Particle Graphics on Reconfigurable Hardware.“ *ACM Transactions on Reconfigurable Technology and Systems, Vol. 1, No. 3, Article 15*, September 2008.

Botsch, Mario, a Leif Kobbelt. „High-Quality Point-Based Rendering on Modern GPUs.“ *11th Pacific Conference on Computer Graphics and Applications (PG'03).* 2003. 335.

Freund, Yoav, a Robert E. Schapire. „A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.“ *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory.* 1995. 23-37.

Goodman, J.W. *Introduction to Fourier Optics.* Roberts and Co. Publishers, 2005.

Granát, Jiří, Adam Herout, Michal Hradiš, a Pavel Zemčík. „Hardware Acceleration of AdaBoost Classifier.“ *Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI).* 2007.

Grossman, J.P., a William J. Dally. „Point Sample Rendering.“ *Rendering Techniques '98*, July 1998: 181-192.

Hanák, Ivo, Pavel Zemčík, Martin Žádník, a Adam Herout. „Hologram Synthesis Accelerated in Field-Programmable Gate Arrays by Partial Qudaratic Interpolation.“ *Optical Engineering*, August 2009: 1-7.

Havel, Jiří, a Adam Herout. „Yet Faster Ray-Triangle Intersection (Using SSE4).“ *IEEE Transactions on Visualization and Computer Graphics*, 2009.

Herout, Adam, a další. „Low-Level Image Features for Real-Time Object Detection.“ V *Pattern Recognition*. Vienna, AT: IN-TECH, 2009.

Herout, Adam, a Jiří Havel. „Image Pattern Inert to Object Detection.“ *submitted to Pattern Recognition Letters*, 2009.

Herout, Adam, a Michal Hradiš. „EnMS - Early non-Maxima Suppression.“ *submitted to International Journal of Computer Vision*, 2009.

Herout, Adam, a Pavel Zemčík. „Animated Particle Rendering in DSP and FPGA.“ *Proceedings of Spring Conference on Computer Graphics.* Bratislava, SK, 2004. 237-242.

—. „Hardware Implementation of EWA.“ *Proceedings of Computer Vision and Graphics.* Warsaw, PL: Springer Verlag, 2004. 593-598.

—. „Hardware Pipeline for Rendering Clouds of Circular Points.“ *Proceedings of Winter School of Computer Graphics.* Plzeň, CZ, 2005. 17-22.

Herout, Adam, Pavel Zemčík, Roman Juránek, a Michal Hradiš. „Implementation of the "Local Rank Differences" Image Feature Using SIMD Instructions of CPU.“ *Proceedings of the Sixth Indian Conference on Computer Vision, Graphics and Image Processing.* Bhubaneswar, IN, 2008.

Herout, Adam, Radovan Jošth, Pavel Zemčík, a Michal Hradiš. „GP-GPU Implementation of the "Local Rank Differences" Image Feature.“ *Proceedings of the*

International Conference on Computer Vision and Graphics. Heidelberg, DE: Springer, 2008.

Herout, Adam, Radovan Jošth, Roman Juránek, Jiří Havel, Michal Hradiš, a Pavel Zemčík. „Real-Time Object Detection on CUDA.“ *submitted to Journal of Real-Time Image Processing*, 2009.

Ito, T., N. Masuda, K. Yoshimura, A. Shiraki, T. Shimobaba, a T. Sugie. „Special-Purpose Computer Horn-5 for a Real-Time Electroholography.“ *Optics Express, 13(6)*, 2005: 1923-1932.

Janda, M., I. Hanák, a L. Onural. „Hologram synthesis from photorealistic reconstruction.“ *Journal of the Optical Society of America, A 25(12)*, 2008: 3038-3096.

Kensler, A., a P. Shirley. „Optimizing Ray-Triangle Intersection via Automated Search.“ *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing.* 2006. 33-38.

Levoy, Mark, a Turner Whitted. *The Use of Points as Display Primitives.* Technical report TR 85-022, The University of North Carolina at Chapel Hill, Department of Computer Science, 1985.

Möller, Tomas, a Ben Trumbore. „Fast, Minimum Storage Ray-Triangle Intersection.“ *Journal of Graphics Tools, Vol. 2, No. 1*, 1997: 21-28.

Pfister, H., M. Zwicker, J. van Baar, a M. Gross. „Surfels: Surface Elements as Rendering Primtives.“ *Proceedings of SIGGRAPH.* 2000. 335-342.

Polok, Lukáš, Adam Herout, Pavel Zemčík, Michal Hradiš, Roman Juránek, a Radovan Jošth. „"Local Rank Differences" Image Feature Implemented on GPU.“ *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems.* Heidelberg, DE: Springer, 2008. 170-181.

Rusinkiewicz, Szymon, a Mark Levoy. „QSplat: A Multiresolution Point Rendering System for Large Meshes.“ *Computer Graphics, Proceedings of SIGGRAPH.* 2000. 242-352.

Shevtsov, M., A. Soupikov, a A. Kapustin. „Ray-Triangle Intersection Algorithm for Modern CPU Architectures.“ *Proceedings of GraphiCon.* 2007. 33-39.

Schmittler, Jörg, Ingo Wald, a Philipp Slusallek. „SaarCOR: a hardware architecture for ray tracing.“ *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware.* Saarbrucken, Germany , 2002. 27-36.

Schmittler, Jörg, Sven Woop, Daniel Wagner, Wolfgang J. Paul, a Philipp Slusallek. „Realtime Ray Tracing of Dynamic Scenes on an FPGA Chip.“ *Proceedings of Graphics Hardware 2004.* Grenoble, France, 2004.

Šilhavá, Jana, a další. „Testbench for Evaluation of Image Classifiers.“ *Computer Graphics and Geometry, ISSN 1811-8992*, 2007: 31-47.

Šochman, J., a J. Matas. „WaldBoost - Learning for Time Constrained Sequential Detection.“ *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2.* 2005.

Viola, Paul, a Michael Jones. „Rapid Object Detection Using a Boosted Cascade of Simple Features.“ *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 2001. 511.

Wald, A. „Sequential tests of statistical hypotheses.“ *The Annals of Mathematical Statistics, 16(2)*, 1945: 117–186.

Wald, Ingo. *Realtime Ray Tracing and Interactive Global Illumination.* Ph.D. dissertation, Saarland University, 2004.

Weyrich, Tim, a others. „A Hardware Architecture for Surface Splatting.“ *Computer Graphics, Proceedings of SIGGRAPH.* 2007.

Zemčík, Pavel, Adam Herout, a Pavel Tišnovský. „Particle Rendering Pipeline.“ *Proceedings of Spring Conference on Computer Graphics.* Bratislava, SK, 2003. 180-186.

Zemčík, Pavel, Adam Herout, Luděk Crha, Pavel Tupec, a Otto Fučík. „Particle rendering pipeline in DSP and FPGA.“ *Proceedings of Engineering of Computer-Based Systems.* Los Alamitos, US: IEEE Computer Society, 2004. 361-368.

Zwicker, M., H. Pfister, J. van Baar, a M. Gross. „Surface Splatting.“ *Computer Graphics, Proceedings of SIGGRAPH 2001.* Los Angeles, 2001.