Improved Sampling for Temporal Anti-Aliasing

(A Sobel Improved Temporal Anti-Aliasing)

Christian Alexander Oliveros Labrador
christianol_01@hotmail.com

April 24, 2018

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Michael Doggett, michael.doggett@cs.lth.se

Examiner: Flavius Gruian, Flavius.Gruian@cs.lth.se

Abstract

Anti-aliasing is a key component of modern 3D computer generated imagery. For Real-Time image generation in applications such as games it is important to increase the sampling rate per pixel to improve overall image quality. But increasing sampling can be expensive, especially for current Deferred Rendering architectures. An innovative solution to this issue is the Temporal Anti-Aliasing (TAA) technique which combines samples from previous frames with the current frame samples to effectively increase the sampling rate. In this thesis, we will explore methods to improve the quality of TAA by using edge detection, of both color and depth, and triangle indexing to ensure only samples belonging to the current frames pixel are blending together. Our objective is to reduce ghosting and other TAA artifacts created with current implementations. Quality improvement will be evaluated by comparing TAA generated images to ground truth images generated by using much higher sample counts that would not be practical in real-time.

Keywords: TAA, Sobel, Anti-Aliasing, Triangle Indexing, TRAA

Acknowledgements

If you want to thank people, do it here, on a separate right-hand page. Both the U.S. *acknowledgments* and the British *acknowledgements* spellings are acceptable.

We would like to thank Lennart Andersson for his feedback on this template.

We would also like thank Camilla Lekebjer for her contribution on this template, as well as Magnus Hultin for his popular science summary class and example document.

Contents

Chapter 1
Introduction

Modern Computer Graphics are based on rendering scenes that are made of objects represented as models composed of primitive polygons, the triangle being the most common one used. This is to take advantage of their simplicity and all their geometric properties to create optimal algorithms to handle their rendering. Triangles are composed of three vertices, each of them which consists on a position and other parameters associated with them, i.e. the color or normals of the triangle, for interpolation.

When we want to render the objects in a scene, we take the vertices and send them to the Rendering Pipeline. There, they are processed and mapped to the pixels of the screen with their respective color.

We have two main uses for this process: Offline Applications, like movies; and Real-time Applications, like videogames. Each of them have their requirements and constraints, but for this project we will only give attention to Real-time Applications.

The focus of this project is to improve Temporal Anti-Aliasing implementation, which is a technique that increases the quality of the images after the process of mapping triangles to pixels by mixing frames previously rendered with current ones.

The main requirement would be to render the highest quality possible representation of the scene, with two main constraints: we must render at least thirty frames per second, with no high frame rate loss; and we must work with a limited amount of memory and bandwidth, because we need to be able to run in an average computer or mobile device.[10, 1]

1.1 Problem Definition

Temporal Anti-Aliasing (TAA) is a relatively new real time technique that provides good results without incurring in heavy memory or processing power costs of other techniques. Edge detection and triangle indexing appear as good candidates to improve the quality of the technique by reducing the ghosting and blurring unwanted effects created by current implementations of TAA.

The aim of this thesis is to improve the Temporal Anti-Aliasing technique by using edge detection, of both color and depth, and triangle indexing techniques to reduce blurring and ghosting without decreasing the quality of the rendered image or incurring in heavy memory or processing power costs.

1.2 Related Work

As the simplest Anti-Aliasing technique, we have Super Sampling Anti-Aliasing (SSAA), it consists on rendering at a higher resolution and then downsampling it to the required resolution. Another technique is the Multi Sample Anti-Aliasing (MSAA), which calculates the color for the final pixel just once [10]. We can learn what would become the base of Temporal Reprojection Anti-Aliasing (TAA or TRAA) in the papers Accelerating Real-time Shading with Reverse Reprojection Caching by Nehab D., Sander P. V., Lawrence J., Tatarchuk N., Isidoro J. R. [11], in which they describe how pixel shaders could use save information and reproject it in the next frame; and Amortized Supersampling by Yang L., Nehab D., Sander P. V., Sitthiamorn P., Lawrence J., Hoppe H. [18] in which they describe how to use the reprojection of old frames in the current one as a method of real time Anti-Aliasing.

Next, we start to see Post Processing techniques like Fast Approximate Anti-Aliasing (FXAA) by Timothy Lottes [9] which uses a form of edge detection to correct aliasing while being compatible with the deferred shading architecture. We also found the Crytek implementation of Temporal Anti-Aliasing (TAA or TXAA) explained by Tiago Sousa on his presentation Anti-Aliasing Methods in CryENGINE 3 [8].

Enhanced Subpixel Morphological Antialiasing (SMAA) by Jorge Jimenez, Jose I. Echeverria, Tiago Sousa and Diego Gutierrez [7] which uses a more complex edge reconstruction technique while being able to work with SSAA, MSAA and a basic form of TAA.

Finally, we have the TRAA implementations of Ke Xu for Uncharted 4 and Lasse Fuglsang for Inside, which implement new advances like the Color Clipping Box and Sharpen Filter. This two last implementations are used the base of this master thesis. [13, 16]

Chapter 2
Technological Background

2.1 C++ and Bonobo Framework

C++ is a compiled general-purpose programming language with imperative, object-oriented programming and low-level memory management features. It is widely used in Computer Graphics due to its performance, especially on real-time applications, and its wide knowledge base. The Bonobo Framework is the base of the laboratories of Computer Graphics (EDAF80) and High-Performance Computer Graphics (EDAN35) courses from Lund University. It is developed in C++ and provides a rendering engine that is easy to modify and use. OpenGL and GLSL The Open Graphics Library (OpenGL) is a 2D and 3D computer cross-platform open source graphics Application Programming Interface (API) that abstracts the programmer from directly interacting with Graphic Processor Units (GPUs) to achieve hardware accelerated rendering. It provides the programmer with a graphics pipeline to use, which is normally implemented through hardware. OpenGL Shading Language (GLSL) is a high-level shading language that allows programmers greater control of the graphics pipeline without requiring use of the OpenGL assembly language or hardware specific languages. MATLAB MATLAB is a proprietary multi-paradigm numerical computing environment. Commonly used for science, engineering and economics. It is popular for image processing applications because of its wide library of algorithms for this purpose, including image metrics which are used in this thesis.

Chapter 3
Formatting

Avoid empty spaces between *chapter-section*, *section-sub-section*. For instance, a very brief summary of the chapter would be one way of bridging the chapter heading and the first section of that chapter.

3.1 Page Size and Margins

Use A4 paper, with the text margins given in Table 3.1.

Table 3.1: Text margins for A4.

margin	**space**
top	3.0cm
bottom	3.0cm
left (inside)	2.5cm
right (outside)	2.5cm
binding offset	1.0cm

3.2 Typeface and Font Sizes

The fonts to use for the reports are **TeX Gyre Termes** (a **Times New Roman** clone) for serif fonts, **TeX Gyre Heros** (a **Helvetica** clone) for sans-serif fonts, and finally **`TeX Gyre Cursor`** (a **`Courier`** clone) as mono-space font. All these fonts are included with the TeXLive 2013 installation. Table 3.2 lists the most important text elements and the associated fonts.

Table 3.2: Font types, faces and sizes to be used.

Element	Face	Size	LaTeXsize
Ch. label	**serif, bold**	24.88pt	`\huge`
Chapter	**serif, bold**	24.88pt	`\Huge`
Section	**sans-serif, bold**	20.74pt	`\LARGE`
Subsection	**sans-serif, bold**	17.28pt	`\Large`
Subsubsection	**sans-serif, bold**	14.4pt	`\large`
Body	serif	12pt	`\normalsize`
Header	serif, SmallCaps	10pt	
Footer (page numbers)	serif, regular	12pt	
Figure label	**serif, bold**	12pt	
Figure caption	serif, regular	12pt	
In figure	sans-serif	*any*	
Table label	**serif, bold**	12pt	
Table caption and text	serif, regular	12pt	
`Listings`	`mono-space`	$\leq$ 12pt	

3.2.1 Headers and Footers

Note that the page headers are aligned towards the outside of the page (right on the right-hand page, left on the left-hand page) and they contain the section title on the right and the chapter title on the left respectively, in SmallCaps. The footers contain only page numbers on the exterior of the page, aligned right or left depending on the page. The lines used to delimit the headers and footers from the rest of the page are $0.4pt$ thick, and are as long as the text.

3.2.2 Chapters, Sections, Paragraphs

Chapter, section, subsection, etc. names are all left aligned, and numbered as in this document.

Chapters always start on the right-hand page, with the label and title separated from the rest of the text by a $0.4pt$ thick line.

Paragraphs are justified (left and right), using single line spacing. Note that the first paragraph of a chapter, section, etc. is not indented, while the following are indented.

3.2.3 Tables

Table captions should be located above the table, justified, and spaced 2.0cm from left and right (important for very long captions). Tables should be numbered, but the numbering is up to you, and could be, for instance:

- **Table X.Y** where X is the chapter number and Y is the table number within that chapter. (This is the default in LaTeX. More on LaTeX can be found on-line, including whole books, such as [5].) or

- **Table Y** where Y is the table number within the whole report

As a recommendation, use regular paragraph text in the tables, bold headings and avoid vertical lines (see Table 3.2).

3.2.4 Figures

Figure labels, numbering, and captions should be formed similarly to tables. As a recommendation, use vector graphics in figures (Figure 3.1), rather than bitmaps (Figure 3.2). Text within figures usually looks better with sans-serif fonts.

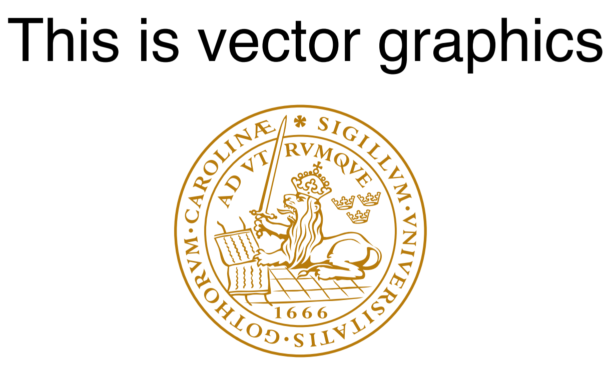

Figure 3.1: A PDF vector graphics figure. Notice the numbering and placement of the caption. The caption text is indented 2.0cm from both left and right text margin.

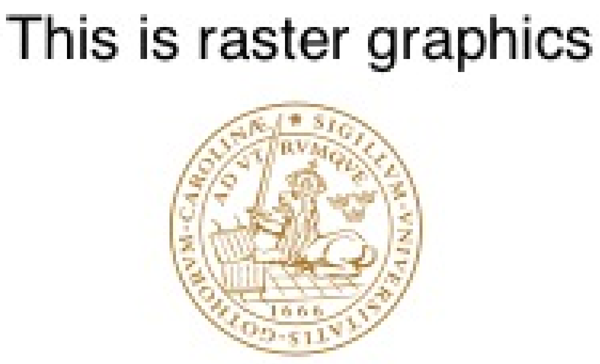

Figure 3.2: A JPEG bitmap figure. Notice the bad quality of such an image when scaling it. Sometimes bitmap images are unavoidable, such as for screen dumps.

For those interested in delving deeper into the design of graphical information display, please refer to books such as [15, 4].

3.3 Mathematical Formulae and Equations

You are free to use in-text equations and formulae, usually in *italic serif* font. For instance: $S = \sum_i a_i$. We recommend using numbered equations when you do need to refer to the specific equations:

$$E = \int_0^\delta P(t)dt \quad \longleftrightarrow \quad E = mc^2 \tag{3.1}$$

The numbering system for equations should be similar to that used for tables and figures.

3.4 References

Your references should be gathered in a **References** section, located at the end of the document (before **Appendices**). We recommend using number style references, ordered as appearing in the document or alphabetically. Have a look at the references in this template in order to figure out the style, fonts and fields. Web references are acceptable (with restraint) as long as you specify the date you accessed the given link [14, 3]. You may of course use URLs directly in the document, using mono-space font, i.e. `http://cs.lth.se/`.

3.5 Colours

As a general rule, all theses are printed in black-and-white, with the exception of selected parts in selected theses that need to display colour images essential to describing the thesis outcome (*computer graphics*, for instance).

A strong requirement is for using **black text on white background** in your document's main text. Otherwise we do encourage using colours in your figures, or other elements (i.e. the colour marking internal and external references) that would make the document more readable on screen. You may also emphasize table rows, columns, cells, or headers using white text on black background, or black text on light grey background.

Finally, note that the document should look good in black-and-white print. Colours are often rendered using monochrome textures in print, which makes them look different from on screen versions. This means that you should choose your colours wisely, and even opt for black-and-white textures when the distinction between colours is hard to make in print. The best way to check how your document looks, is to print out a copy yourself.

Chapter 4
Language

You are strongly encouraged to write your report in English, for two reasons. First, it will improve your use of English language. Second, it will increase visibility for you, the author, as well as for the Department of Computer Science, and for your host company (if any).

However, note that your examiner (and supervisors) are not there to provide you with extensive language feedback. We recommend that you check the language used in your report in several ways:

Reference books dedicated to language issues can be very useful. [6]

Spelling and grammar checkers which are usually available in the commonly used text editing environments.

Colleagues and friends willing to provide feedback your writing.

Studieverkstaden is a university level workshop, that can help you with language related problems (see Studieverkstaden's web page).

Websites useful for detecting language errors or strange expressions, such as

- `http://translate.google.com`
- `http://www.gingersoftware.com/grammarcheck/`

4.1 Style Elements

Next, we will just give some rough guidelines for good style in a report written in English. Your supervisor and examiner as well as the aforementioned **Studieverkstad** might have a different take on these, so we recommend you follow their advice whenever in doubt. If you want a reference to a short style guide, have a look at [12].

Widows and Orphans

Avoid *widows* and *orphans*, namely words or short lines at the beginning or end of a paragraph, which are left dangling at the top or bottom of a column, separated from the rest of the paragraph.

Footnotes

We strongly recommend you avoid footnotes. To quote from [2], *Footnotes are frequently misused by containing information which should either be placed in the text or excluded altogether. They should be avoided as a general rule and are acceptable only in exceptional cases when incorporation of their content in the text [is] not possible.*

Active vs. Passive Voice

Generally active voice (*I ate this apple.*) is easier to understand than passive voice (*This apple has been eaten (by me).*) In passive voice sentences the actor carrying out the action is often forgotten, which makes the reader wonder who actually performed the action. In a report is important to be clear about who carried out the work. Therefore we recommend to use active voice, and preferably the plural form *we* instead of *I* (even in single author reports).

Long and Short Sentences

A nice brief list of sentence problems and solutions is given in [17]. Using choppy sentences (too short) is a common problem of many students. The opposite, using too long sentences, occurs less often, in our experience.

Subject-Predicate Agreement

A common problem of native Swedish speakers is getting the subject-predicate (verb) agreement right in sentences. Note that a verb must agree in person and number with its subject. As a rough tip, if you have subject ending in *s* (plural), the predicate should not, and the other way around. Hence, *only one s*. Examples follow:

incorrect He have to take this road.

correct He has to take this road.

incorrect These words forms a sentence.

correct These words form a sentence.

In more complex sentences, getting the agreement right is trickier. A brief guide is given in the *20 Rules of Subject Verb Agreement* [19].

Chapter 5
Structure

It is a good idea to discuss the structure of the report with your supervisor rather early in your writing. Given next is a generic structure that is a starting point, but by no means the absolute standard. Your supervisor should provide a better structure for the specific field you are writing your thesis in. Note also that the naming of the chapters is not compulsory, but may be a helpful guideline.

Introduction should give the background of your work. Important parts to cover:

- Give the context of your work, have a short introduction to the area.
- Define the problem you are solving (or trying to solve).
- Specify your contributions. What does this particular work/report bring to the research are or to the body of knowledge? How is the work divided between the co-authors? (This part is essential to pinpoint individual work. For theses with two authors, it is compulsory to identify which author has contributed with which part, both with respect to the work and the report.)
- Describe related work (literature study). Besides listing other work in the area, mention how is it related or relevant to your work. The tradition in some research area is to place this part at the end of the report (check with your supervisor).

Approach should contain a description of your solution(s), with all the theoretical background needed. On occasion this is replaced by a subset or all of the following:

- **Method**: describe how you go about solving the problem you defined. Also how do you show/prove that your solution actually works, and how well does it work.
- **Theory**: should contain the theoretical background needed to understand your work, if necessary.

- **Implementation**: if your work involved building an artefact/implementation, give the details here. Note, that this should not, as a rule, be a chronological description of your efforts, but a view of the result. There is a place for insights and lamentation later on in the report, in the Discussion section.

Evaluation is the part where you present the finds. Depending on the area this part contains a subset or all of the following:

- **Experimental Setup** should describe the details of the method used to evaluate your solution(s)/approach. Sometimes this is already addressed in the **Method**, sometimes this part replaces **Method**.
- **Results** contains the data (as tables, graphs) obtained via experiments (benchmarking, polls, interviews).
- **Discussion** allows for a longer discussion and interpretation of the results from the evaluation, including extrapolations and/or expected impact. This might also be a good place to describe your positive and negative experiences related to the work you carried out.

Occasionally these sections are intermingled, if this allows for a better presentation of your work. However, try to distinguish between measurements or hard data (results) and extrapolations, interpretations, or speculations (discussion).

Conclusions should summarize your findings and possible improvements or recommendations.

Bibliography is a must in a scientific report. LaTeX and `bibtex` offer great support for handling references and automatically generating bibliographies.

Appendices should contain lengthy details of the experimental setup, mathematical proofs, code download information, and shorter code snippets. Avoid longer code listings. Source code should rather be made available for download on a website or on-line repository of your choosing.

Bibliography

[1] E. Angel and D. Shreiner. *Computer Graphics A Top-Down Approach with Shader-Based OpenGL 6th Edition*. Addison-Wesley, 2011.

[2] Commonwealth Forestry Association. On-line guide to scientific writing. `http://www.cfa-international.org/ONGSWfinish.html`.

[3] CTAN. Comprehensive tex archive network. `http://www.ctan.org`.

[4] S. Few. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, 2012.

[5] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[6] J.A.W. Heffernan, J.E. Lincoln, and J. Atwill. *Writing: A College Handbook*. W.W. Norton, 2000.

[7] Jorge Jimenez, Jose I Echevarria, Tiago Sousa, and Diego Gutierrez. SMAA: Enhanced subpixel morphological antialiasing. *EUROGRAPHICS 2012 / P. Cignoni, T. Ertl Volume 31 (2012), Number 2*, 2012.

[8] Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, and Tiago Sousa. Filtering approaches for real-time anti-aliasing. In *ACM SIGGRAPH Courses*, 2011.

[9] Timothy Lottes. FXAA. *NVIDIA Docs*, 2009. The Document was last edited in 2011.

[10] Michael Doggett. EDAN35 High Performance Computer Graphics, November 2017.

[11] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro. Accelerating real-time shading with reverse reprojection caching. *In Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware (2007), pp. 25–35. 3, 8*, 2007.

[12] University of Washington. Style points for scientific writing. `http://www.psych.uw.edu/writingcenter/writingguides/pdf/style.pdf`.

[13] Lasse Jon Fuglsang Pedersen. Temporal Reprojection Anti-Aliasing in INSIDE. *GDC Vault*, 2016. Accessed: 2017-11-28, `http://www.gdcvault.com/play/1022970/Temporal-Reprojection-Anti-Aliasing-in`.

[14] Will Robertson and Khaled Hosny. The fontspec package, May 2012. `http://ctan.uib.no/macros/latex/contrib/fontspec/fontspec.pdf`.

[15] Edward R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, USA, 1986.

[16] Ke XU. Temporal Antialiasing In Uncharted 4. *SIGGRAPH 2016*, 2016.

[17] Yale Graduate Writing Center. The most common sentence structure problems. `http://www.yale.edu/graduateschool/writing/forms/The%20Most%20Common%20Sentence%20Structure%20Problems.pdf`.

[18] L. Yang, D. Nehab, P. V. Sander, P. Sitthiamorn, J. Lawrence, and H. Hoppe. Amortized supersampling. *ACM Trans. Graph. 28 (2009), 135:1–135:12*, 2009.

[19] yourdictionary.com. 20 rules of subject verb agreement. `http://grammar.yourdictionary.com/sentences/20-Rules-of-subject-verb-agreement.html`.

Appendices

Appendix A
About This Document

The following environments and tools were used to create this document:

- operating system: Mac OS X 10.10.1
- tex distribution: MacTeX-2014, `http://www.tug.org/mactex/`
- tex editor: Texmaker 4.4.1 for Mac, `http://www.xm1math.net/texmaker/` for its XeLaTeX flow (recommended) or pdfLaTeX flow
- bibtex editor: BibDesk 1.6.3 for Mac, `http://bibdesk.sourceforge.net/`
- fonts `cslthse-msc.cls` document class):

  for XeLaTeX: TeX Gyre Termes, TeX Gyre Heros, `TeX Gyre Cursor` (installed from the TeXLive 2013)

  for pdfLaTeX: TeX Gyre font packages: tgtermes.sty, tgheros.sty, tgcursor.sty, gtxmath.sty (available through TeXLive 2013)
- picture editor: OmniGraffle Professional 5.4.2

A list of the essential LaTeXpackages needed to compile this document follows (all except `hyperref` are included in the document class):

- `fontspec`, to access local fonts, needs the XeLaTeX flow
- `geometry`, for page layout
- `titling`, for formatting the title page
- `fancyhdr`, for custom headers and footers
- `abstract`, for customizing the abstract

- `titlesec`, for custom chapters, sections, etc.
- `caption`, for custom tables and figure captions
- `hyperref`, for producing PDF with hyperlinks
- `appendix`, for appendices
- `printlen`, for printing text sizes
- `textcomp`, for text companion fonts (e.g. bullet)
- `pdfpages`, to include the popular science summary page at the end

Other useful packages:

- `listings`, for producing code listings with syntax colouring and line numbers

Appendix B
List of Changes

Since 2016/04/29

- A better template for the popular science summary, by Magnus Hultin.

Since 2015/09/11

- Added a template for the popular science summary, in the `popsci` directory.
- Added code in the report that imports the one page popular science `pdf` at the end of the document.

Since 2015/04/27

- Improved the **Structure** chapter and added more detailed comments for each part.

Since 2014/02/18

- Added the possibility to specify two supervisors. Use either of the `\supervisor{}` or `\supervisors{}{}` commands to set the names and contacts on the first page.

Since 2013/09/23

- Added missing colon ":" after *Examiner* on the front page.

Since 2013/08/30

- Changed fonts from Garamond (Times New Roman), Helvetica (Arial), Courier (Source Code Pro) to Tex Gyre fonts, namely Termes, Heros, Cursor, which are freely available with TexLive 2013 installation. These are all clones of Times New Roman, Helvetica and Courier, respectively. Garamond is problematic on some systems, being a non-freely available font.

- Corrected the *Face* column in Table 3.2 to correctly depict the font face.

Since 2013/02/22

- Number of words required in the abstract changed to 150 (from 300).

Since 2013/02/15

- Made a separate document class, for clarity.

- made it work with pdfLaTeX and garamond.sty, in addition to XeLaTeX and true type fonts. It is up to the user to get the hold of the garamond.zip from `http://gael-varoquaux.info/computers/garamond/index.html`.

EXAMENSARBETE Application Specific Instruction-set Processor Using a Parametrizable multi-SIMD Synthesizeable Model Supporting Design Space Exploration
STUDENT Magnus Hultin
HANDLEDARE Flavius Gruian (LTH)
EXAMINATOR Krzysztof Kuchcinski (LTH)

Parametrisk processor modell för design utforskning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Magnus Hultin**

Applikations-specifika processorer är allt mer vanligt för få ut rätt prestanda med så lite resurser som möjligt. Detta arbete har en parametrisk modell för att kunna testa hur mycket resurser som behövs för en specifik applikation.

För att öka prestandan i dagens processorer finns det vektorenheter och flera kärnor i processorer. Vektorenheten finns till för att kunna utföra en operation på en mängd data samtidigt och flera kärnor gör att man kan utföra fler instruktioner samtidigt. Ofta är processorerna designade för att kunna stödja en mängd olika datorprogram. Detta resulterar i att det blir kompromisser som kan påverka prestandan för vissa program och vara överflödigt för andra. I t.ex. videokameror, mobiltelefoner, medicinsk utrustning, digital kameror och annan inbyggd elektronik, kan man istället använda en processor som saknar vissa funktioner men som istället är mer energieffektiv. Man kan jämföra det med att frakta ett paket med en stor lastbil istället för att använda en mindre bil där samma paketet också skulle få plats.

I mitt examensarbete har jag skrivit en modell som kan användas för att snabbt designa en processor enligt vissa parametrar. Dessa parametrar väljs utifrån vilket eller vilka program man tänkta köra på den. Vissa program kan t.ex. lättare använda flera kärnor och vissa program kan använda korta eller längre vektorenheter för dess data.

För att kunna välja vilken typ av processor som är rätt för den specifika applikationen krävs det ofta att man snabbt kan testa olika prototyper. Att implementera dessa till hårdvara kan ofta vara tidskrävande och ifall det visar sig att implementationen inte klarar dem kraven man ställt för prestanda och energieffektivitet, måste man designa för nya parametrar och mer tid har blivit slösat. Om den här processen istället kan göras automatiskt utifrån dessa design-parametrar kan man teoretiskt spara en massa tid. Modellen testades med olika multimedia program. Den mest beräkningsintensiva och mest upprepande delen av programmen användes. Dessa kallas för kärnor av programmen. Kärnorna som användes var ifrån MPEG och JPEG, som används för bildkomprimering och videokomprimering.

Resultatet visar att det finns en prestanda vinst jämfört med generella processorer men att detta också ökar resurserna som behövs. Detta trots att den generella processorn har nästan dubbelt så hög klockfrekvens än dem applikations-specifika processorerna. Resultatet visar också att schemaläggning av instruktionerna i programmen spelar en stor roll för att kunna utnyttja resurserna som finns tillgängliga och därmed öka prestandan. Med den schemaläggningen som utnyttjade resurserna bäst var prestandan minst 79% bättre än den generella processorn.