Analyzing the Impact of Occlusion on the Quality of Semantic Segmentation Methods for Point Cloud Data

Bachelor Thesis
13th March 2023 - 13th September 2023

Yufeng Xiao
19-763-663

Supervisors:
Prof. Dr. Renato Pajarola
Lizeth Joseline Fuentes Perez

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

Abstract

The abstract heading is similar to any main section heading but without numbering and is similar to the main body text format. Avoid referencing any bibliography in the abstract text.

A clear organization of a report typically encloses the inner main technical part(s) by an introduction and related work section at the beginning, as well as a discussion and/or conclusion section at the end. The inner technical parts must include the precise problem statement(s), a detailed description of the technical solution (e.g. mathematical models, data structures and algorithms), implementation details if applicable, and experimental results.

Contents

1 Introduction and related works

1.1 Previous work

To the best of our knowledge there is not explicit works that compute oclusion level for an entire scene. Some related works are as follows:

1.1.1 Scene Flow Estimation on 3D Point Clouds

Occlusion Guided Scene Flow Estimation on 3D Point Clouds OcCo: Unsupervised Point Cloud Pre-training via Occlusion Completion

1.2 Positioning and contributions

The introduction is a crucial part of your report and sets the stage as well as motivates the work you have completed. It briefly introduces the problem, emphasizes its importance and shows to the reader what you did in your work. That is, you show the reader that your problem is fundamental and that it needs to be solved as it fills a gap in the known literature and prior work.

The introduction sets the stage and it motivates why the work presented is of any interest, it introduces the problem and emphasizes its importance. It makes the reviewer interested and strategizes the presentation – arguments that lead to the conclusion that the considered problem is important. A practical problem motivates the work and hints at the gaps in the known literature.

Note that the introduction must briefly but clearly state the open problems, goals, or design criteria that previous work did not yet completely solve. However, in the introduction this is done without going into details as the related work section will follow up on this in more detail.

The introduction starts by addressing the problem in general and points the reader in the direction how you want to solve your problem / research question. As a metaphor, John Swales developed a three-stage model for research introductions:

Move 1: Establish a territory (claim centrality of topic)

Move 2: Establish a niche (indicate gap)

move 3: Occupy the niche (outline purpose and indicate research structure/methods).

You may follow that model while writing the introduction. In the first part of the introduction or in a separate section, you are expected to mention and organize supporting literature where you outline the *state of the art* of your research question.

At the end of the introduction you may include a short summary of the structure of the rest of the paper, but this can be omitted for saving space (and avoiding the obvious). The introduction together with the conclusion should give a complete short version of your work and what you have achieved.

1.3 Technical background

1.3.1 Point Cloud Data

A point cloud is a collection of data points situated in a three-dimensional coordinate system. Each point in this system is represented by a set of three numbers, denoting its position in space. This representation is fundamental for capturing the intricate details of 3D environments, be it the bustling streets of a city or the serene interiors of a room. Point clouds are often derived from 3D sensors or scanning mechanisms, and their rich information content makes them the primary input format for semantic segmentation tasks.

Given the complexity of point cloud data and the challenges posed by occlusions, the Minkowski Engine's focus on sparse tensors offers a promising avenue for advancing the state of the art in semantic segmentation. By leveraging its capabilities, researchers and practitioners can hope to achieve more accurate and robust segmentation results, even in challenging environments.

1.3.2 Semantic Segmentation

Semantic segmentation for point cloud data has rapidly ascended as a pivotal research domain, given its profound implications in a myriad of applications. From the intricate pathways navigated by autonomous vehicles to the precise movements of robotics and the detailed analysis of 3D scenes, the ability to accurately segment and categorize each data point in a 3D environment is paramount.

At the heart of this research lies the challenge of dealing with occlusions. In real-world scenarios, objects within a scene often overlap or obstruct each other, leading to partial or even complete occlusions. Such occlusions can significantly distort the spatial distribution of data points, making it challenging to discern the true structure and category of the obstructed objects. For instance, in an urban driving scenario, a pedestrian might be partially hidden behind a parked car, or in an indoor setting, a chair might be obscured by a table. These occlusions can lead to misclassifications, reducing the overall accuracy of the segmentation process.

This project proposal is rooted in the quest to unravel the intricacies of occlusion within point cloud data. Specifically, we aim to delve deep into understanding how varying levels of occlusion in a scene impact the performance and quality of semantic segmentation methods. By systematically analyzing the effects of occlusion, we aspire to shed light on potential strategies to enhance segmentation accuracy, even in highly occluded environments.

Through this investigation, we hope to not only advance the state of the art in point cloud data segmentation but also pave the way for more robust applications in autonomous driving, robotics, and 3D scene analysis.

Minkowski Engine

The Minkowski Engine stands out as a state-of-the-art tool in this domain. It is an auto-differentiation library specifically designed for sparse tensors. In the realm of deep learning, where dense tensors are commonly used, the Minkowski Engine brings a fresh perspective by focusing on sparse tensors. This is particularly beneficial for 3D data, which often exhibits spatial sparsity. The engine supports all standard neural network layers, including convolution, pooling, unpooling, and broadcasting operations, but tailored for sparse tensors. Such capabilities make it an ideal choice for semantic segmentation tasks, especially when dealing with point cloud data.

1.4 Motivation

Among the myriad factors influencing the semantic segmentation of point cloud data, the level of occlusion stands as a paramount challenge. Occlusions, a prevalent phenomenon in 3D scenes, can significantly compromise the quality and integrity of data. When objects are partially or entirely obscured by others, conventional semantic segmentation approaches might falter, leading to inaccuracies in segmentation. While the issue of occlusion has garnered attention in 3D data processing, current research on how different occlusion levels impact the quality of semantic segmentation remains fragmented. Specifically, there's a palpable gap in understanding how to quantify occlusion levels and how these levels influence the performance of advanced tools like the Minkowski Engine.

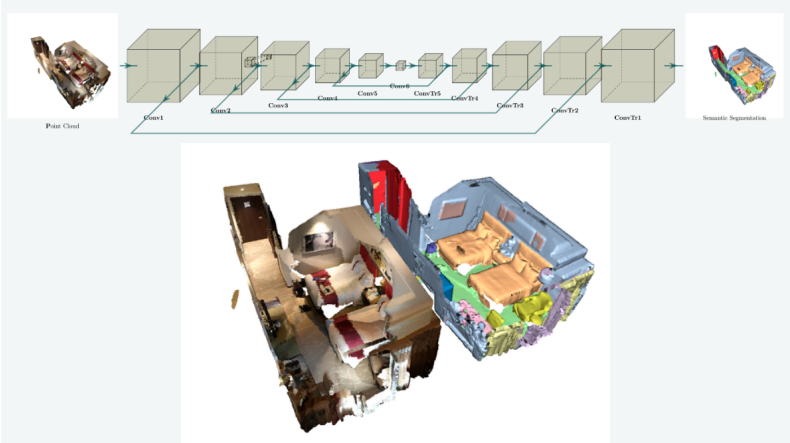

Figure 1.1: Minkowski Engine Indoor Scene Segmentation

Thus, the primary motivation behind this research is to systematically evaluate occlusion levels and delve deep into their implications on point cloud data semantic segmentation. Through this investigation, we aim to offer more precise semantic segmentation methodologies, especially in environments with high levels of occlusion. In essence, our objective is to forge a nexus between occlusion levels and the efficacy of semantic segmentation, providing invaluable insights for future research and applications.

1.5 Outline

In this work we use ray-tracing based methods to scan ground truth cloud and compute occlusion level for the scanned cloud. Before we doing this, we aplly similar methods to estimate occlusion level of mesh to validate that ray-based methods are reliable.

Finally, an interactive web application is developed to visualize the point cloud together with a backend is developed to compute the occlusion level of the point cloud.

2 Problem Statement

The technical sections introduce and motivate the proposed solution in the light of the problem, which requires a precise problem statement together with any assumptions and requirements. Therefore, before the technical solutions are presented in detail, a precise definition of the problem to be solved must be given. Where algorithmic or mathematical descriptions are not appropriate, other technical and implementation problems can be stated that are to be solved in this work.

A simple and straight-forward problem or goal could sufficiently be stated already in the introduction. This could include examples such as insufficient performance, that there was no interactive method for doing something, or that there was no real-time rendering of this data possible so far. The problem statement and definition could also be integrated in the section that describes the new proposed and presented solutions, especially if there are multiple different (smaller) problems to be addressed which can be given at the beginning of the main technical paper sections.

Alternatively, in particular if the open problem to be solved is not that simple, it may have to be described in full detail in a dedicated problem section.

2.1 Occlusion Level of Mesh

We have to validate here that more viewpoints lead to lower occlusion level of the interior scene.

2.1.1 Ground Truth Mesh

We should compute the occlusion level of ground truth mesh.

2.1.2 Estimated Mesh from Point Cloud

We also want to estimate the occlusion level of the mesh generated from ground truth point cloud.

Why do we wanna do thi

2.2 Occlusion Level of Point Cloud

After validation in previous steps, we should directly compute the occlusion level of the ground truth point cloud.

2.3 Evaluation

It's essential to evaluate result of segmentation of point cloud so that we can maybe find correlation between occlusion level and segmentation performance.

3 Technical Solution

One or more sections should be directed towards the detailed description of the proposed solution, including technical details about the used data structures, algorithms and mathematical methods. The technical description should allow the resourceful and interested reader to reproduce and verify your work, together with the implementation information given in a later section.

This is the most important part of the report that should answer every little technical question that arises in the reader's mind. Your algorithm might be a puzzle with many pieces which are described in a linear order in the report – and many such orders may be possible. A good order to describe the different components of your solution is one that allows to clearly explain one component after another based exclusively on what the reader has already seen in any previous sections, thus minimizing any forward references.

This core part of the report should be organized into coherent subsections, giving an overview and introducing formalism first. Following an overview of the necessary steps of the entire method, each step can then be described elaborately in each subsection.

For each subtask or -problem that is solved, a clear definition is needed and it must be explained how it is solved. This needs to be precise, and requires an algorithmic, technical or mathematical rigorous description, e.g. including:

- descriptions of algorithms and data structures and how they are used to process the data
- descriptions of the output data structure or formats that are generated, including intermediate stages as needed
- explaining any data processing or feature extraction step by appropriate mathematical formulas
- define data, formats and functions properly
- make clear the input and output arguments of functions

Note that implementation details should be avoided as much as possible, and the focus should be on the formal and algorithmic solution; the implementation section is specifically targeted to explain any programming details.

3.1 Occlusion Level of Mesh

3.1.1 Uniform Sampling of Triangles

We choose random uniform sampling here. Randomly generate arguments to compute barycentric coordinates of sampled points.

3.1.2 Ray Triangle Intersection

Octree Acceleration

Build Octree Based on Cetroid of Triangles.

3.1.3 Visible Area Ratio

Compute the visible area ratio based on sampled points.

3.2 Estimated Mesh from Point Cloud

We have to segment the point cloud into different clusters first.

3.2.1 Region Growing to Segment Point Cloud

Gain a number of clusters of point cloud.

3.2.2 Build Triangles

First steo is always to build triangles.

Convex Hull Estimate Polygon

Compute convex hull of each cluster to estimate the polygon.

Compute Polygon Cetroid

Compute cetroid of each polygon. And connect cetroid with vertices of polygon to form triangles.

3.3 Occlusion Level of Point Cloud

Since we have ground truth point cloud, we should generate point cloud from certain viewpoints.

3.3.1 Ray Tracing Scanning

We use ray tracing based method to scan the ground truth point cloud, then we can get a point cloud with occlusion.

Spherical Light source

Uniformly sample points on a sphere to simulate light source. Number of rays is given.

Ray Point Intersection

Each point in the point cloud is a sphere with a certain radius.

Ray Openings Intersection

Openings in the scene should not be identified as occlusion.

3.3.2 Ray Based Occlusion Computation

In previous steps, rays are classified based on if it intersect with the scene. If not, it's an occlusion ray. We count occlusion rays to compute occlusion level.

3.4 Evaluation

3.4.1 Metrics

IoU

F1 Score

Accuracy

Recall

Precision

3.4.2 Semantic Classes

4 Implementation

The implementation section focuses on the major programming problems and details such as the overall organization of the source code, major dependencies of different modules etc. In the core of the report this should be kept concise, and should be developed top-down, focusing on anything that is maybe not standard good practice, or that has a non-obvious implementation. An appendix can be used for more details if needed.

4.1 Algorithm

4.1.1 Ray Tracing Scanning

4.1.2 Occlusion Detection

Ray Sphere Intersection

Ray Triangle Intersection

Octree Traversal

We use the octree to store the point cloud data. The octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three-dimensional space by recursively subdividing it into eight octants. Octrees are the three-dimensional analog of quadtrees.

4.1.3 Region Growing

4.2 Software

We present this software in a form of a web application. The backend is written in C++ and the frontend is written in TypeScript. The backend is responsible for the computation of the occlusion detection algorithm and the frontend is responsible for the visualization of the point cloud and the occlusion detection result. The backend and the frontend communicate with each other through a websocket connection.

4.2.1 Backend

Our backend is written in C++. We use the following libraries:

- **PCL** - Point Cloud Library
- **Eigen** - C++ template library for linear algebra
- **JsonCpp** - C++ library for manipulating JSON values
- **Websocketpp** - C++ websocket client/server library

PCL

PCL is a large scale, open project for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to

recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them – to name a few.

Eigen

Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. It supports all matrix sizes, from small fixed-size matrices to arbitrarily large dense matrices, and even sparse matrices. It supports all standard numeric types, including std::complex, integers, and is easily extensible to custom numeric types.

JsonCpp

JsonCpp is a C++ library that allows manipulating JSON values, including serialization and deserialization to and from strings. It can also preserve existing comment in unserialization/serialization steps, making it a convenient format to store user input files.

Websocketpp

Websocketpp is a C++ websocket client/server library. It is intended to be used in conjunction with a websocket implementation that provides: a way to connect, a way to send data, and a way to receive data. Websocketpp does not perform any I/O operations itself, instead it delegates these tasks to the user provided I/O type.

4.2.2 Frontend

Three.js

Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL.

Web Technical Stack

This front-end web interface is built on top of the following web technical stack:

- **TypeScript** - A strict syntactical superset of JavaScript and adds optional static typing to the language.
- **TailwindCSS** - A utility-first CSS framework for rapidly building custom user interfaces.
- **Vite** - A build tool that aims to provide a faster and leaner development experience for modern web projects.
- **Websocket** - A computer communications protocol, providing full-duplex communication channels over a single TCP connection.
- **Three.js** - A cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser.

User Interface

4.2.3 Structure

Flowchart

Class Diagram

Sequence Diagram

4.3 Command Line Only Mode

4.3.1 Usage

Arguments

5 Experimental Results

The evaluation discusses the proposed and competing solutions in the light of the initially stated problem requirements and limitations. This typically involves some sort of experimental evaluation which leads to some type of qualitative or quantitative results.

Quantitative results include observed numbers indicating performance timings (speed) or accuracy measures of the given implementation and test datasets. If possible, statistical tests and analysis should be given, or where applicable formal proofs. Meaningful and informative numerical results must be complete and unambiguous. Explain in detail how the evaluation has been designed, as well as the experimental setup and test cases. This includes accurate description of the test data (type, properties, size etc.) as well as the test setup (e.g. view settings, screen resolution etc.), the values of parameters and the measured variables (frame rate, throughput, accuracy etc.).

Qualitative results may be reported if clear quantitative measures are not feasible or applicable. Qualitative results clearly show the features and functionality of the completed work, indicating if and how they are novel or different from prior work. Qualitative results are especially suitable if something *new* has been achieved that no-one has done before in the same way.

Essentially, the goal of the experimental results is to convince the reader by numbers, tests and images (and maybe user studies), giving some sort of proof why the proposed solution is good, different and/or better than other solutions.

6 Conclusion and Discussion

The last section puts the results in perspective, discussing it in relationship to other related work in one or maximum two pages. Indicate possible (side-)effects and eventual limitations due to the evaluation, but try to keep yourself short and clear. Give a short discussion about your results where you focus on what your findings mean. E.g., show how your results and interpretations agree with the original problem question and with other published work or if there are any other practical applications for your work.

State the *take home message* of the paper that the reader should remember and provide an outlook on possible future work that extends the given solution or fixes specific limitations. Close with a brief description (that is different from the Abstract) of the proposed solution.

Acknowledgements

Acknowledge any data and code sources you used or for help you received.

Document, Writing and Formatting Guidelines

This part of the document uses non-numbered chapter and section headings as they are not part of a regular report structure. In a regular report, `chapters`, `sections` and `subsections` should normally be numbered. Note the use of comment lines and spacings to give more structure to the ASCII text LaTeX document.

In this appendix like part we discuss the structure and formatting rules and guidelines to follow for a written project, research paper, Bachelor or Master thesis report.

Text

Overall Strategies

The appearance, clarity and organizational structure of a paper is as important as its technical content, but the technical content must be there beforehand. The presentation alone, however, can make the difference between a mediocre and great publication. Exploit all suitable mechanical rules that can always be applied and optimized independently of the technical part.

The paper has to be convincing even to the adverse reader, i.e. a critical reviewer evaluating your work. Think of being a reviewer yourself, not really knowing your domain and possibly not specifically interested in your work. All information must be crystal clear to a non-expert reader, and all terms and concepts must be properly introduced in a logical order. Put yourself in the position of reading that topic and your work for the very first time, with the goal of having to reproduce it afterwards. The presentation must be flawless and the length of the paper must match the amount of content.

Your report must present your work and solution, best within a convincing story about a difficult problem challenge and an important application domain. The text has to encompass and sell your work. Build up and identify the key challenges in the introduction and problem description. Show clearly how you solved exactly this very important problem in a great and unique novel way.

Group your ideas and concepts hierarchically into sections, subsections and even paragraphs. Strictly introduce general and common ideas, concepts and techniques before expanding further on them. Use the concepts of *repetition* and *parallelism* in your text and structure. The main concept of repetition is to:

> Introduce what you are going to tell them – then tell them in detail – and finally review what you told them.

This approach of repetition is applied on all levels of a report, overall document, sections, subsections and paragraph, always in an appropriate level of abstraction or detail. Example levels:

Document The introduction briefly describes the main problem, previews your contribution and summarizes the main results. The main technical sections describe your approach in more detail and the experiments show the achieved results. Finally the conclusion, discussion and future work section(s) summarize your contributions.

Section In the first paragraph(s) you introduce the topic or aspect that this section covers, followed by the (technical) details, and the last paragraph typically wraps it up, or leads to the next section.

Paragraph The first sentence of a paragraph leads into the main *message* of this paragraph, and the last sentence concludes it or leads over to the next paragraph.

The concept of parallelism means to apply the same strategy or structuring to different parts. E.g. for each technical problem question or topic covered in one section, first introduce the problem definition and then describe the solution subsequently, possibly in subsections and paragraphs. Or for each type or version of experimental results, or for each data set, describe the data, method parameters, measured variables, report and discuss the results.

It is important to get your text exactly clear to the reader and to avoid the impression that something has been left out or that your contribution is not that significant. But do not over-claim, and more importantly do not under-claim either.

Writting and Structure

Writing is difficult work and usually takes more time than expected. It's beneficial to formalize and write down your progress as early as possible. Generally *you cannot really be sure that you know something until you are able to explain it* well in writing. Hence conveying ideas exactly but in a concise and compact manner is very important and a key to successful writing. Preciseness and compactness are key to be able to describe a large amount of work and results that you have.

Your text must be smooth, forming a clear and logical order of your thoughts and arguments. Use *parallelism* to introduce a set of topics, questions or issues and then elaborate on them subsequently in sections and paragraphs.

Use *repetition*, e.g. introduce the problem, show how to solve it and review the benefits. Do not assume that the reader still remembers what was mentioned only as a passing comment three pages back, use repetition and parallelism.

Do not abruptly jump topics but motivate topic changes. If the topics of text parts change too much, then divide them into subsections or paragraphs. If between paragraphs there are major changes (sequence of different concepts etc.), try to use inline paragraph headings for easy navigation and orientation. One way to integrate such paragraph sequences is to write an introductory paragraph at the beginning of the section and use parallelism.

Sections and Paragraphs

Use `sections`, `subsections` and `paragraphs` to structure your ideas and content into meaningful parts, and make a clear and meaningful order of them. Each section, subsection or paragraph must cover a clearly delineated topic or idea.

Every section must first introduce to the reader what to expect and then tell the details, following the principles of repetition and parallelism. An introductory paragraph is also a good approach to fill the space between the section heading and the first sub heading if subsections are used. The last paragraph of a section can summarize the concepts and lead over to the next topic or section.

Each paragraph should have one clear single message, there should only be one consistent topic or idea what the paragraph is about. The first paragraph of a section should clearly lead into the main topic of that section, and the first line of a paragraph should state or at least clearly lead into the main message of that paragraph.

Every single sentence should be fully comprehensible in its context, taking only minimal preliminary knowledge of previous paragraphs into account.

Wording and Postprocessing

Use single tenses (i.e. the present or present perfect) as much as possible when describing your design, technical or algorithmic solutions. Generally use present perfect for describing implementations and results which were completed, as this implies something that happened as part of this work in the past but continues to be valid in the present as a result of this paper.

"I" versus "we": For a personal opinion, or your specific personal contribution, you can use the first person. In a personal thesis report one can use "I" as this one's own contribution and text.

For most situations a neutral form should be used, passive voice or third person, but be careful to avoid the interpretation of "passive voice" as "someone else did it, and it is not our contribution", e.g. "This mind-boggling observation was made." vs. "We made this mind-boggling observation.".

Read through the text, spell check, and check the text with a grammar tool as well. Obvious errors are unacceptable. Try to take the role of a reviewer or evaluator: Reading your paper or report costs that person significant time, so question everything. The reviewer may not be forgiving if something is not clear. Be the devil's advocate!

Do not use abbreviations (don't -¿ do not).

Formatting and Typesetting

General Rules

The main text area should have a margin of about 2cm on both sides, and the main text should have a top and bottom margin of about 3cm each.

Section and subsection headings have nested numbering and are typeset in sans-serif bold font type, e.g. Helvetica or Arial. Sizes range from 18pt for main sections down to at least one point larger than the main body text size. Some vertical space before and after section headings is placed (automatically according to the LaTeX document class of this template).

The main body text is typeset in 11pt Times font (serif font family). Text body is one-column, justified and single-spaced. First line of a paragraph is generally indented with about 0.8cm, however, first paragraph after any section heading may also be non-indented.

- Avoid extensive and manual spacing
- Use font modifications carefully
- Use proper math and symbol styles

Folder Structure

Use main folder for the main latex .tex and .bib files, the main paper body and bibliography database. Optional: use a folder, e.g. ./sections, to separately manage individual section's latex files.

Use ./images and ./figures subdirectories for the raster images and vector graphics used in the figures and diagrams of the paper. Only use .jpg and .pdf formats respectively for best results. Put source files, e.g. from Illustrator or OmniGraffle, also directly into the ./figures folder along with the PDF versions.

Latex Coding

Follow a clear prologue structure in the LaTeX source file. After the given template components add your `\usepackage{}` block, and `\input{math}` to include our standard math formula definitions. Complete the preamble by any further specific definitions or commands, e.g. `\TODO` and `\FIXME` macros. After the prologue, complete the title and author parts.

If separate .tex files are used for the main report text body, then include them in the main body latex file, and in each of these files indicate the root latex file at the very top:

```
%\!TEX root = ../<main_file_name>.tex!
```

Further LaTeX coding guidelines and recommendations:

- Use the `\emph{}` command fo emphasizing/italicizing not `\textit()`
- Use (our math.tex) style file for math formula consistency
- Use `\mathrm{function()}` for function names
- Use consistent section, equation and figure labelling `\label{sec:introduction}` as well as `eq:rendering`, `fig:system`

- Do not liberally introduce manual horizontal or vertical spacings
- In particular do not use comment codes % to control spacing or indentation before or after elements, e.g. arounc equations, figures or tables

Mathematical Formulas and Equations

Mathematical expressions should follow a consistent set of rules, symbols and formatting as indicated in our `math.tex` style file. LaTeXiT can be used for consistent symbols and formulas in figures. Apply the same scalar, vector and matrix styles as well as use the same variables consistently throughout your text, see also the predefined styles and specific variables in `math.tex`. A few guidelines that are useful are given below, use them as much as possible (but adjust as necessary to make formulas clear):

- use (lower-case) italic letters for normal (scalar) variables such as x or t
- prefer i and j as index variables and m and n to denote number of elements or iterations
- use for example other letters such as a, b and c for constants
- use lower-case bold italic to denote vectors such as $\boldsymbol{u}$ or $\boldsymbol{v}$
- use upper-case bold letters such as $\mathbf{M}$ or $\mathbf{N}$ for sets and matrices
- use upper- or lower-case italic letters such as $f()$ or $G()$ to denote functions
- use regular plain font and decimal point to denote explicit constants such as 2 or 100.12

Use as much formalism, variables and equations, as is useful to clearly understand your description, but not for trivial facts. Use inline equation format also for variables like x and numbers like 12 used in the main text body segments.

Mathematical equations that are important or are referenced should be laid out as a regular equation with consecutive numbering to the right as below:

$$E = m \cdot c^2 \tag{6.1}$$

Such regular free standing equations do not need a punctuation and follow a paragraph that ends with a dot or double-colon, and the following paragraph starts regularly indented.

Equations may also be treated 'inline' with the main flow of the text, thus being part of one regular sentence. Such 'inline' equations as

$$a^2 + b^2 = c^2,$$

typically end with a comma and the text continues below unindented lower-case to finish the sentence. To adjust spacing, the equation can be connected to the paragraph text flow by separating it only using % comments. If an equation ends the sentence of a paragraph it ends with a dot.

Floats, Figures, Screenshots and Graphs

Figures, screenshots, tables, graphs and other floats should support the overall idea of the paper or some description in the text specifically, and are numbered sequentially. Every figure must support a key idea or concept, and it must be meaningful with self-explanatory captions.

Make sure every figure or float is placed correctly and is used as well as referenced in the text.

Placement Each float is centered and *placed in the text directly after* the first paragraph referencing it, never directly after a heading. If formatting constraints are difficult, figure placement may be forced, e.g. to be placed at the top of the next possible new page. In double-column document formats, put large floats spreading both columns at the top of the next following page.

Figures Use a consistent style and appearance for each figure, using clean lines and diagrams. Make careful use of not too bright or disturbing colors, and watch out for grayscale usage when printed.

Use sans-serif fonts in figures and diagrams unless for math symbols and variables. Enforce consistent upper-lower case usage throughout all figures and terms used in the text, and match variables and math formulas or symbols in figures to the ones in in the text. Also match the size of text in figures to the size of the main paper body text.

For figures and diagrams that include any vector graphics elements (e.g. arrows, lines, boxes, points etc.), use the PDF vector graphics format with white or transparent background, and do not use raster image formats. If raster images are included in vector graphics figures, also use the PDF format for the combined figure.

Screenshots Each screenshot must demonstrate a clear visual effect, or major point of your work, or an example or problem of a standard (prior) approach. Some, very few, images may be mostly visual teasers. Each screenshot must include all details such as (data) statistics, used parameters and experimental settings in the caption or in the accompanying text where the figure is referenced from. Use raster image formats only for image-only figures, and then use the JPG file format not PNGs.

Graphs Graphs and plots must clearly demonstrate some numerical test results or data statistics. Strictly avoid clutter within one graph, and clearly relate parameters and results of experiments. Just listing basic values can be done in simple tables. Use expressive and clearly labelled axis in all graph plots. Graphs should basically be understandable on their own along with their caption also without reading the main text. Graphs and plots are usually based on vector graphics, thus use the PDF vector graphics format with white or transparent background as for figures and diagrams.

Cross-References

Cross-references to numbered items such as sections and figures are capitalized as in the following examples. In Section 1 we provide a brief summary of text formatting instruction and Figure 6.1 shows our group logo. A figure can contain multiple subfigures which can be referenced individually as illustrated in Figure 6.2(b).

Figure 6.1: Example of single figure with caption text.

(a) vmml logo

(b) uzh logo

Figure 6.2: Example subfigure with captions referencing (a) the VMML and (b) the UZH logos.

Use the same full or shortened form for all references, e.g. Sec. 1, Fig. 6.2(a), Eq. 6.1.

Pseudo Code

Example for a pseudo code given below

```
nlevels = log2(I) + 1
for l = nlevels : -1 : 1
  loop over all 2x2x2 blocks (i,j,k)
    A_l(i,j,k) = average(block)
  end
end
```

Commands

You can use the command `\FIXME{}` in order to mark sections in the text, which need to be edited and fixed. E.g. **FIX**: check reference XY.

Bibliography

The bibliography with list of references is placed on a new page at the end of the document, titled "References" or "Bibliography" and typeset similar to the main section headings but without numbering. Bibliography entries should follow standard IEEE or ACM proceedings or transaction journal formatting styles, and be referenced by last name abbreviation and year such as e.g. [Paj07] and [GSSP10], or by number as for [1].

Formatting References

Bibliography entries should be clean, accurate, compact and consistent across all entries. Thus for the entries in the BibTeX bibliography .bib file follow the following basic rules:

- Have complete entries, including full author names (first and last), full conference paper (inproceedings type with name, year, pages) and journal (article type with journal name, volume, number/issue, month) data
- Use capitalized title and special terms, e.g. "Point Set Processing for Data Analysis on the GPU".
- Use consistent venue description (name the same conference/journal in the same way).
- Use compact *inproceedings* booktitle for conferences without year, i.e. "Proceedings IEEE Visualization" instead of "8th Int. Conference on Visualization (VIS'08), IEEE 2008".
- Do not use separate organization field if clear from the conference or journal, and usually no publisher, month and address for conferences.
- Keep entries concise, avoid unnecessary or duplicate information such as e.g. address or location for conferences or even journals, redundant numpages fields, or duplicated association/organization data etc.
- Use these fields sparsely if at all necessary: howpublished, publisher (don't for conferences, only for books or so), series (don't for conferences, only for LNCS or similar)
- Typically do not use venue, address or location for conferences or even journals
- Keep keywords simple and meaningful (e.g. main first keyword graphics, visualization, geometry, theory, databases, mathematics etc. followed by a few more specific ones)

Example bad BibTeX entry:

```
@ inproceedings{as78439729asf,
  Title = {Tile-based LOD for the Parallel Age},
  Author = {Niski, Krzysztof and Cohen, J.~D.},
  Booktitle = {Proceedings IEEE Visualization Conference (IEEE Vis'10)},
  Volume = {13},
  Pages = {1352},
  Organization = {IEEE},
  Series = {IEEE TVCG journal},
  Publisher = {Computer Society Press},
}
```

Above BibTeX entry will result in a not very nice, incomplete and inconsistent reference:

[NC] Krzysztof Niski and J. D. Cohen. Tile-based lod for the parallel age. In *Proceedings IEEE Visualization Conference (IEEE Vis'10)*, volume 13 of *IEEE TVCG journal*, page 1352. IEEE, Computer Society Press.

The corresponding clean and consistent BibTeX entry would be:

```
@article{NC:07,
  Title = {Tile-based {LOD} for the Parallel Age},
  Author = {Niski, Krzysztof and Cohen, Jonathan~D.},
  Journal = {IEEE Transactions on Visualization and Computer Graphics},
  Month = {November/December},
  Volume = {13},
  Number = {6},
  Pages = {1352--1359},
  Year = {2007}
}
```

This will result in a nice and consistent reference:

[NC07] Krzysztof Niski and Jonathan D. Cohen. Tile-based LOD for the parallel age. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1352-1359, November/December 2007.

More bad examples include the following

> [HBC10a, SWF10a, HXS09a, CH09a, Str09a, FM07a, WGS07a, CMF05a, AGL+02a, KCCO00a, CYH+97a],

and the good ones are these here

> [HBC10b, SWF10b, HXS09b, CH09b, Str09b, FM07b, WGS07b, CMF05b, AGL+02b, KCCO00b, CYH+97b].

Pages

Cover and Abstract

Title of thesis, author, affiliation, date and any other administrative information should be placed on a separate cover page (see first page).

Main title of work should be typeset in large sans-serif bold font type. Title is to be followed by author and affiliation. At the bottom, separated group affiliation is set.

Abstract follows on separate page after cover page and before the table-of-content page(s).

TOC

The table-of-content (TOC) contains all Section and (Sub-)Sub Section headings and their corresponding pages and is listed on separate pages before the first section. It has its own heading typeset as a section heading without numbering.

Bibliography

[AGL+02a] J. Allard, V. Gouranton, L. Lecointre, E. Melin, and B. Raffin. Netjuggler: Running VR Juggler with multiple displays on a commodity component cluster. In *Proceeding IEEE Virtual Reality*, pages 275–276, 2002.

[AGL+02b] Jérémie Allard, Valérie Gouranton, Loïck Lecointre, Emmanuel Melin, and Bruno Raffin. Net Juggler: Running VR Juggler with multiple displays on a commodity component cluster. In *Proceeding IEEE Virtual Reality*, pages 275–276, 2002.

[CH09a] Clement Courbet and Celine Hudelot. Random accessible hierarchical mesh compression for interactive visualization. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1311–1318, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[CH09b] Clement Courbet and Celine Hudelot. Random accessible hierarchical mesh compression for interactive visualization. In *Proceedings Eurographics Symposium on Geometry Processing*, pages 1311–1318, 2009.

[CMF05a] Xavier Cavin, Christophe Mion, and Alain Filbois. COTS cluster-based sort-last rendering: Performance evaluation and pipelined implementation. In *Proceedings IEEE Visualization*, pages 111–118. Computer Society Press, 2005.

[CMF05b] Xavier Cavin, Christophe Mion, and Alain Filbois. COTS cluster-based sort-last rendering: Performance evaluation and pipelined implementation. In *Proceedings IEEE Visualization*, pages 111–118, 2005.

[CYH+97a] Tzi-cker Chiueh, Chuan-kai Yang, Taosong He, Hanspeter Pfister, and Arie E. Kaufman. Integrated volume compression and visualization. In *Proceedings of the 8th conference on Visualization '97*, VIS '97, pages 329–336, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[CYH+97b] Tzi-cker Chiueh, Chuan-kai Yang, Taosong He, Hanspeter Pfister, and Arie E. Kaufman. Integrated volume compression and visualization. In *Proceedings IEEE Visualization*, pages 329–336, 1997.

[FM07a] Nathaniel Fout and Kwan-Liu Ma. Transform coding for hardware-accelerated volume rendering. *IEEE Trans Vis Comput Graph*, 13(6):1600–1607, 2007.

[FM07b] Nathaniel Fout and Kwan-Liu Ma. Transform coding for hardware-accelerated volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1600–1607, 2007.

[GSSP10] Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. Interactive SPH simulation and rendering on the GPU. In *Proceedings ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–64, 2010.

[HBC10a] Mark Howison, E. Wes Bethel, and Hank Childs. Mpi-hybrid parallelism for volume rendering on large, multi-core systems. In James P. Ahrens, Kurt Debattista, and Renato Pajarola, editors, *EGPGV*, pages 1–10. Eurographics Association, 2010.

[HBC10b] Mark Howison, Wes E. Bethel, and Hank Childs. MPI-hybrid parallelism for volume rendering on large, multi-core systems. In *Proceedings Eurographics Symposium on Parallel Graphics and Visualization*, pages 1–10, 2010.

[HXS09a] Chang Hui, Lei Xiaoyong, and Dai Shuling. A dynamic load balancing algorithm for sort-first rendering clusters. In *IEEE International Conference on Computer Science and Information Technology*, pages 515 –519, aug. 2009.

[HXS09b] Chang Hui, Lei Xiaoyong, and Dai Shuling. A dynamic load balancing algorithm for sort-first rendering clusters. In *Proceedings IEEE International Conference on Computer Science and Information Technology*, pages 515 –519, 2009.

[KCCO00a] Vladlen Koltun, Yiorgos Chrysanthou, and Daniel Cohen-Or. Virtual occluders: An efficient intermediate pvs representation. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 59–70, London, UK, 2000. Springer-Verlag.

[KCCO00b] Vladlen Koltun, Yiorgos Chrysanthou, and Daniel Cohen-Or. Virtual occluders: An efficient intermediate PVS representation. In *Proceedings Eurographics Workshop on Rendering Techniques*, pages 59–70, 2000.

[Paj07] Renato Pajarola. Efficient data structures. In Markus H. Gross and Hanspeter Pfister, editors, *Point-Based Graphics*, Series in Computer Graphics, pages 148–165. Morgan Kaufmann Publishers, 2007.

[Str09a] Filip Strugar. Continuous distance-dependent level of detail for rendering heightmaps. *journal of graphics, gpu, and game tools*, 14(4):57–74, 2009.

[Str09b] Filip Strugar. Continuous distance-dependent level of detail for rendering heightmaps. *Journal of Graphics, GPU and Game Tools*, 14(4):57–74, 2009.

[SWF10a] Tim Suss, Timo Wiesemann, and Matthias Fischer. In *Proceedings of the 2010 IEEE Fifth International Conference on Networking, Architecture, and Storage*, NAS '10, pages 448–456, Washington, DC, USA, 2010. IEEE Computer Society.

[SWF10b] Tim Suss, Timo Wiesemann, and Matthias Fischer. Evaluation of a c-load-collision-protocol for load-balancing in interactive environments. In *Proceedings IEEE International Conference on Networking, Architecture, and Storage*, pages 448–456, 2010.

[WGS07a] Chaoli Wang, Antonio Garcia, and Han-Wei Shen. Interactive level-of-detail selection using image-based quality metric for large volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13:122–134, 2007.

[WGS07b] Chaoli Wang, Antonio Garcia, and Han-Wei Shen. Interactive level-of-detail selection using image-based quality metric for large volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):122–134, January/February 2007.