University of Bremen

Faculty 03: Mathematics/Computer Science

Master Thesis
Computer Science

Enhancing and Analyzing Apollonian 3-d Sphere Packings for Arbitrary Objects

Joscha Cepok <joscha.cepok@gmail.com>
Matrikelnummer. 2763987

Submission Date	28th of September 2019
Primary examiner	Prof. Dr. Gabriel Zachmann
Secondary examiner	Dr. René Weller
Supervisors	M.Sc. Philipp Dittmann, Dr. René Weller

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlichgemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderen Prüfungsamt vorgelegen.

Ort, Datum: Unterschrift:

Abstract

The goal of this thesis is to enhance the Apollonian 3-d sphere packing algorithm which is implemented in ProtoSphere.
3-d sphere packing is a technique to fill many 3-d spheres into an object, to cover as much volume as possible without any sphere overlapping with another sphere or the bounds of the object. Furthermore, several sphere packing analysis metrics are introduced. The metrics cover aspects like the distribution of the spheres and covered volume differences over the entire area, filling curves and the distribution of the spheres depending on its size to classify models.
To optimize the packing quality, some adjustments were made to enhance the covered volume. In addition the adjustments, several packing configurations are tested which target aspects like computation time, most total covered volume, target number of spheres or any trade-off to compute the most optimal packing depending on the application.
Another feature which is introduced is a classification of objects with similar characteristics by applying similarity calculations on the final packing.

Contents

Chapter 1

Introduction

1.1 Motivation

A 3-d sphere packing of an object is a set of non-overlapping spheres which are all inside of the object and do not intersect with the surface. Sphere packings have several applications like 3-d printing or collision detection in simulations. The collision detection in particular has practical applications, since haptic feedback can be generated for remote operations like surgeries or maintenance on objects, which are hard to reach like satellites in space. The better the haptic feedback, the more natural the workflow feels, which improves the performance. In addition to the remote surgery, other medical applications like laser therapies also benefit from good sphere packings. To laser a tumor, a laser spreads inside of the tumor in a spherical shape to destroy any sick cells. Obviously a better sphere packing enhances the success of any of those applications. The motivation for this thesis is to optimize the covered volume of sphere packings which are produced by ProtoSphere as much as possible for several applications. ProtoSphere covers on average about 85% of the entire volume. This thesis will start with a hypothesis to examine whether after a certain percentage of covered volume is exceeded. The sphere sizes converge to an equal size, which would make any more packing trivial. This hypothesis is covered in the later stages.

1.2 Goal

The major goal for this thesis is to enhance the performance of ProtoSphere for arbitrary models which a condition, which is that all parts are watertight. Models which meet the condition perform best with the ProtoSphere algorithms but the average covered volume for those models is about 85% while the upper bound for the covered volume depending on the model is somewhere in the range of 93% to 95%, 2.2.4. Therefore, the goal is to cover more than 92% of the volume on average. Once the major issues for those models are solved with stable packing, the solutions can be adapted to other kind of models, which do not meet all the conditions.
Aside from the covered volume, another goal of this thesis is to analyze sphere packing in several aspects. The only measured criteria for the non-parallelized state of ProtoSphere was the covered volume percentage regardless of the distribution of the volume over the model. There are several other factors aside from a total covered volume percentage, like distribution of spheres, volume coverage and densities in the different areas of the object. Aside from other visual representations, there are also metrics like the packing speed, distribution of sphere sizes or the entropy which can be used to measure sphere packings for several aspects. Furthermore, the observed metrics are providing data which can be used to classify objects if the results are similar.
Ultimately this thesis aims to easily verify that future adjustment produces a valid sphere packing.

1.3 Structure

This thesis is structured into four parts: Previous work 2 and architecture 2.11.1, measuring sphere packings 3, improving the packing algorithm 4, interpretation of the results 5 6. The first part is about related work and techniques which were used to achieve the goals. In the next part, several measurement and comparison techniques which can be applied on the packing are introduced. For the third part, observed issues are listed and if possible solved. Enhancements or mitigations are suggested for any part for which other data structures could solve possible issues. Finally the results are presented and discussed.

1.4 Initial Hypothesis

In the PhD thesis [Wel12], assumptions were formulated, which are investigated in this thesis. He wrote: 'An other open problem is the analysis of the voids between the spheres. If it is possible to estimate the voids a priori, we could derive error bounds for our collision detection algorithm.' Based on this statement, two hypothesis are derived. A third hypothesis is set as general goal.

Hypothesis 1 : The more spheres become inserted, the remaining voids become smaller, more similar in shape and size, therefore new spheres add no more information.

Hypothesis 2 : With more spheres the upper bound for the volume approximates 100%.

Hypothesis 3 : A dynamic break criteria to stop the packing when the spheres become too small exists.

In this thesis, all hypothesis are analyzed after several measurement criteria are implemented and applied to the sphere packing.

Chapter 2

Previous Work

There are some publications which focus on several aspects of sphere packing. Some publications try to predict the most covered volume for several dimensions 2.2.4, others describe heuristics for sphere packings with a small number of spheres 2.2.1. As a starting point for my research papers which focus on a meaningful visualization of complex 3-d structures were evaluated as inspiration 2.3 2.4. Additionally, basic mathematical algorithms 2.5 2.6 and the limits of datatypes 2.2.5 which handle uncertainty were considered to handle edge cases. Of course the pioneering work of my predecessors which implemented the initial states of the ProtoSphere framework 2.11 are summarized over the relevant parts of this thesis.

2.1 Models

Any physical object of the real world can be converted into a 3-d model in a mathematical representation. Aside from the polygons several other formats exist which have, depending on the application, some advantages or disadvantages. For this thesis a polygon and Apollonian sphere packing representation are used. The polygon representation is usually called mesh. Meshes are mostly generated using triangular polygons. Since the number of polygons is limited by memory, most models are optically similar but not exactly the same [PAS99]. Depending on the resolution, sharp edges like in Fig: 2.1 appear since meshes do not scale well.

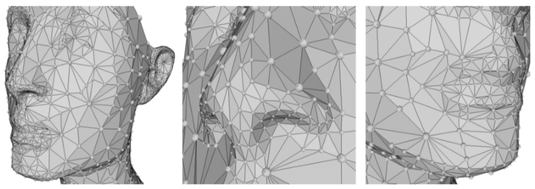

Figure 2.1: Polygon structure of a mesh. [PAS99]

To target that issue, surface meshes exist as alternative representation, since meshes target the volume and surface meshes the optical representation [BHS08]. Since volume meshes constitute the focus of this thesis those are used and for sake of simplicity referred to meshes. Regardless of natural objects, mathematical objects like cubes or pyramids with no curved surface are 100% identical to the real world, given the objects in the real world are considered as perfect mathematical objects.

2.2 3-d sphere packing

Any 3-d sphere packing is a set of spheres, to meet three conditions. All spheres must not overlap with any polygon of the model, all spheres are inside of the model and all spheres are non-overlapping.

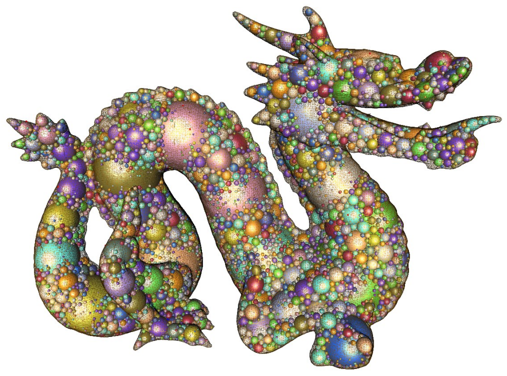

Figure 2.2: 3-d Sphere packing of a dragon. [Wel12]

In Fig: 2.2 a 3-d sphere packing for a Chinese dragon is shown.
There is no lower bound for the minimum distance between spheres, therefore any distance d with $|r_1| + |r_2| \leq |d|$ where r_1 and r_2 describes the radii of two adjacent spheres.

2.2.1 Heuristic sphere packing

In the publication **A Look-Forward Heuristic for Packing Spheres into a Three-Dimensional Bin** [Ake14], Hakim Akeb describes a method to optimize sphere packing with a heuristic instead of a greedy method. The algorithm works for a maximum of about 100 spheres, but needs about one hour to fill a model with these spheres to cover about 55 % of the total volume. Since computation time is also an important factor, and ProtoSphere covers much more volume in a shorter time, a heuristic approach is good for a low number of spheres. For ProtoSphere, any algorithm which has a bigger computation time than one minute for more than 100 spheres is not acceptable.

2.2.2 Kepler's conjecture

In the publication **Circle Packing, Sphere Packing, and Kepler's Conjecture** [Mis16], Roshni Mistry compares several arrangements of spheres of equal size and confirms Kepler's conjecture for the maximum density packing to be 74.048%. But since ProtoSphere is applied on arbitrary models with Apollonian packing, Kepler's conjecture does not fit for this thesis. Still, the lower bound of 70.048% gives an indicator for a lower bound which an Apollonian sphere packing should exceed.

2.2.3 Fractal dimension

In the publication **The Fractal Dimension of the Apollonian sphere packing** [MB95], multiple authors analyzed the fractal dimension of sphere packing, which has the value of about 2.474 for the third dimension the upper bound would be 3. The fractal dimension describes the complexity of shapes in any dimension. As example, the coastline of Great Britain which has a fractal dimension of 1.25, while the mandelbrot set has a fractal dimension of 2, so the fractal dimension increases with the complexity of the shape.

2.2.4 Maximum density

In the publication **How dense can one pack spheres of arbitrary size distribution?** [SDSRH12], multiple authors describe a method to estimate the upper bounds for the maximum density that can be covered by an Apollonian 2-d or 3-d sphere packing. The results depend on the fractal dimension of the object. Therefore the results of [MB95] can be used to predict the upper bound of the covered volume for 3-d sphere packings, which is suggested to be in the range of about 93% to 95% depending on the shape of the object. If the shape of the object is elliptical on which a single sphere could cover more than 95% of the volume, the upper bound is exceeded. The range of 93% to 95% holds for arbitrary objects, which require a decent amount of spheres, to get close to 90% volume coverage.

2.2.5 Limits of floating point

In the paper **What Every Computer Scientist Should Know About Floating-Point Arithmetic** [Gol91], David Goldberg investigates interesting facts about floats. Floats are precise for the first seven digits after the decimal digit. If arithmetic operations are performed on floats, like addition, subtraction, multiplication or division, the result will contain minor errors. This implies that for a sphere with the volume formula $V = \frac{4 \cdot \pi \cdot r^3}{3}$ with $V = 0.0000001$, the minimal radius for acceptable results would be $r = 0.00287941$. For operations with floats which are close to 0.0000001 the results are more likely to have errors. Therefore, any operation on small numbers is avoided if possible. The alternative datatype to replace floats would be doubles, which have a higher precision which would mitigate a majority of the issues. However double precision calculations are available for graphics processing units (GPU) which cost 2,000 € and more (January 19), therefore any GPU calculation is performed with floating precision.

2.2.6 Densest local packing

In the paper **Densest local packing** [HS11], Adam B. Hopkins and Frank H. Stillinger introduce an algorithm to calculate and compare sphere packings for several dimensions. However to apply the similarity formula on two different packings, both packings require the same number of spheres and the same sphere sizes.

$$S = 1 - \frac{\sum_i |n_i - n_i^{ref}|}{2(N-1)}$$

2.3 Image Segmentation

In the journal **A Comparison of X-Ray Image Segmentation Techniques** [SCS13], multiple authors describe how to filter x-ray images for image segmentation with thresholding by the x-ray intensity. This technique is useful to visualize 3-d objects as 2-d representation, Fig: 2.3. In addition to that, the silhouette of objects is clearly visible, which is an important aspect to analyze sphere packings too.

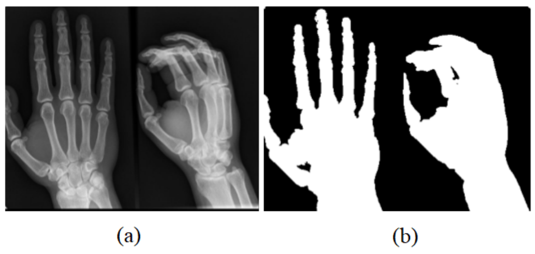

Figure 2.3: a) X-Ray image of a hand b) Segmented image [SCS13]

2.4 Voxel Segmentation

In the paper **A voxel-based technique to estimate the volume of trees from terrestrial laser scanner data**, [BHMv14] multiple authors describe a technique to visualize 3-d volume as 3-d voxel representation Fig: 2.4. Voxel representations are useful to see the thickness of certain parts of a model, which can be applied on any packing to distinguish different areas.

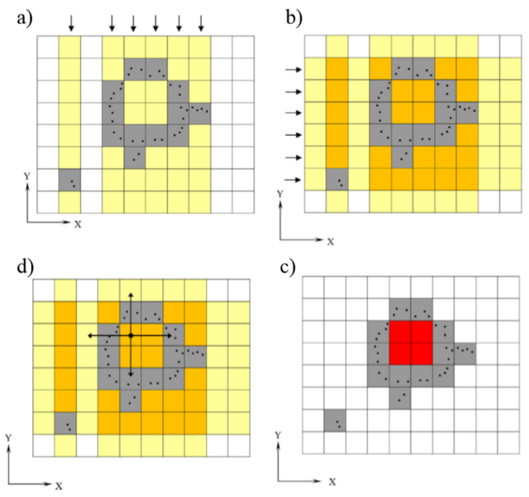

Figure 2.4: Identification of the occluded voxels: a) Scanning in Y-direction; b) Scanning in X-direction; c) Checking of the candidates; d) Occluded voxels. [BHMv14]

2.5 Entropy

In the paper **A Mathematical Theory of Communication** [Sha48], by C. E. Shannon, a formula to calculate the amount of independent information known as the Shannon entropy is introduced.

$$E = \sum_i^n p_i \cdot \log p_i.$$

The entropy indicates how much independent information a data set contains. The bigger the entropy, the more independent information is contained. Applied to a sphere packing, each sphere contains independent information, which scales with the size of the sphere.
Worth mentioning is that each single sphere contributes to the entropy and two different sphere packings with the same amount of covered volume might still differ on the entropy. Given two sets of spheres which cover the same

volume, with the first set containing big and small spheres while the second set contains equal-sized spheres, the second set would have a higher entropy value since each single sphere contributes more volume as do individual spheres on average. Therefore, models with low entropy values depend on a few big spheres to cover the majority of the volume, while models with high entropy values depend on more similar-sized spheres.
An example for the entropy is the Huffman encoding, with the base of 2 the formula predicts the average number of bits which is required to encode a message. For sphere packings, the information is the volume each sphere covers.

2.6 Linear Regression

In this thesis, linear regression is used several times, when no common function was found which fit the data set. The linear regression is calculated by using the method of least squares to calculate the formula $y = m \cdot x + b$ for two data sets to find a possible correlation and regression line [IM]. A linear regression is useful if some processes are almost linear but no other regression fits.

2.7 Voronoi Points

Voronoi points are points in structures in any dimension which have maximum distance to any nearby point. For the first dimension on a line, a Voronoi point would simply be in the middle of the line. A Voronoi point in the second dimension on e.g. a triangle would be the middle of the inner circle of the triangle which touches any line of the triangle. The inner circle covers the most volume of any possible circle, since it has the biggest radius. Therefore, Voronoi points are always the point inside a polygon on which the most area is covered. This characteristic does not change for any dimension, so Voronoi points are a feasible approach for sphere packing.

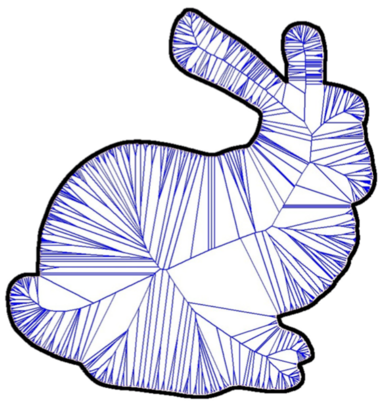

Figure 2.5: Voronoi points of a bunny. Each touching blue line is a Voronoi point with the maximum distance to the outer shell. [Wel12]

In Fig: 2.5 a Voronoi graph is shown, each point in which blue lines intersect is a Voronoi point with the maximum distance to the outer lines.

2.8 Cosine Similarity

In the paper **Plagiarism Detection Between Theory And Practical Calculations** by Intisar H. Albakush [Alb17], several possibilities to detect similarities are discussed. The cosine similarity is mentioned as the best technique to detect similarities of two sets, there also are the Euclidean similarity distance and Jaccard similarity distance. All techniques measure the similarity as percentages: 0% is no similarity at all while 100% means the data sets are identical. Calculating the similarity of two 3-d sphere packings depending on the sphere size distribution can be used to classify objects with similar shapes. The similarity is calculated by the formula:

$$\cos\alpha = \frac{d_1 \cdot d_2}{|d_i| \cdot |d_2|}$$

The result of $\cos\alpha$ multiplied by 100 indicates the similarity as a percentage.

2.9 Kullback-Leibler Divergence

In the paper **Distributions of the Kullback-Leibler divergence with applications** by Belov, Dmitry and D Armstrong, Ronald [BDA11], several possibilities to detect similarities are discussed. Since the similarity implies low difference, the Kullback-Leibler divergence can be used to confirm the similarity. The results are always positive starting with 0 for no divergence and increases depending on the difference between both data sets. The formula to calculate the difference is:

$$D(g||B) = \int |g(x) \log \frac{g(x)}{b(x)}|$$

The result is a number which depends on the base of the logarithm while bigger numbers mean a bigger difference. Since there is no meaningful base, the base e is used. The formula is slightly modified by converting all results to the absolute values, to prevent the case that positive and negative results balance each other which could lead to no divergence.

2.10 Packing Density and Covered Volume

It is important to note, that in this thesis, the packing density and covered volume refers to similar characteristics but are not the same thing. The covered volume is the volume of all spheres inside the model which does not necessarily need to be distributed evenly. If some areas of the models are packed with spheres and other areas are not packed with spheres at all, the density is different for the empty and filled areas, while it is still possible to calculate the covered volume. In the case of the Kepler packing, the density is evenly distributed over the entire area, therefore the covered volume percentage and density are the same. Therefore, in most parts of this thesis, the term *covered volume* is used, except for the parts where the density is explicitly required.

2.11 ProtoSphere

The initial algorithm for ProtoSphere was introduced by Dr. René Weller [Wel12]. As a basic idea the, greedy algorithm inserts a random point inside of the model. Those points are pushed towards the closest current Voronoi point until it stops converging. Once the approximation is close enough, the currently approximated Voronoi point is inserted as the sphere with the maximum non-overlapping radius. By doing this, the volume of the model is covered by a 3-d sphere packing with a growing density per each new sphere.

The initial algorithm for ProtoSphere was enhanced by Jörn Teuber [Teu13]. In that thesis, parallelization on the GPU was applied to the algorithm to test three different approaches: an explicit grid, an implicit grid and a hybrid grid to reduce the computation time by culling different areas.
The first approach uses a grid which is called an Explicit Grid [Teu13], this grid splits the model into several parts to reduce the number of checked polygons for each area. This split especially performs well on models with a high polygon count. The second approach is a grid which is called an Implicit Grid [Teu13]; the purpose of that grid is to use as many cores on modern GPUs as possible. To achieve that, the implicit grid should be as fine as possible, which usually means, until the memory runs out. Implicit grid cells are also marked as inside or outside, which makes it easy to determine if a prototype is inside ore outside of the model to discard outside prototypes at early stages.
To optimize the advantages of the explicit and implicit grid approaches, a third approach was evaluated too. This approach is called a Hybrid Grid [Teu13]. In this thesis, only the hybrid grid is used, since that type performs best. The state after the adjustments is referred to in this thesis as a parallelized state..

There are several other projects implemented by Gabriel Zachmann, Rene Weller and Jörn Teuber which use ProtoSphere. Since ProtoSphere was used on those publications but no adjustments were made, these are not mentioned in particular [WZ10a] [RWT13] [WZ10b].

2.11.1 Architecture

The algorithm of ProtoSphere consists of several steps. For the architecture the hybrid packing procedure is considered, since the entire architecture and any setup would not provide any meaningful information for this thesis. The splitting step, which is done during the filling process, will also be skipped, since that step depends on efficient memory management. With more time, GPUs will receive more memory, therefore those issues will be either solved or reduced with newer graphic card generations. On the other hand, major issues in the algorithm to pack the model with 3-d spheres, adjustments will enhance the performance of ProtoSphere for any future GPU as well. For the remaining chapter, the hybrid setup with a split grid, which contains all information to pack the prototypes, is assumed.

2.11.2 Initialization

On initialization ProtoSphere requires a model, which is converted into a more useful data structure. Once the model is converted, the GridManager is the class to control all procedures of the algorithm. There are several routines, to pack the model with spheres. The sphere packing itself is performed in three major steps, which loops until a break criteria is hit. The three steps are: initialization for the packing, looping over prototype optimization processes and exiting packing and converting valid prototypes to packed spheres.

2.11.3 Inserting prototypes

In the first step Fig: 2.6, the major goal is to distribute initial prototype starting positions inside the model on a random position in each cell.

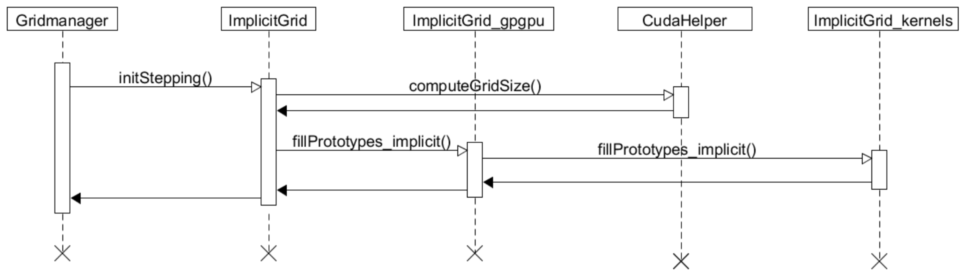

Figure 2.6: The procedure to spawn the prototypes at a random non-intersecting position. The horizontal boxes contain the class names which calls the methods on the vertical lines. The vertical boxes show when a method starts and responds.

If the prototype is outside of the model, it is discarded; to check that the initial position of the prototype is validated against each polygon for that certain cell, outside prototypes are discarded. For every prototype which is still valid, each is validated against each intersecting sphere and each sphere which is inside of the grid. Prototypes which are placed inside of a sphere are pushed outside in the direction opposite to the center of the sphere. All prototypes which are still inside their initial cells will be considered for the optimization step.

2.11.4 Optimizing prototype positions

After prototypes are distributed over the grid, the position is optimized to cover as much volume as possible.

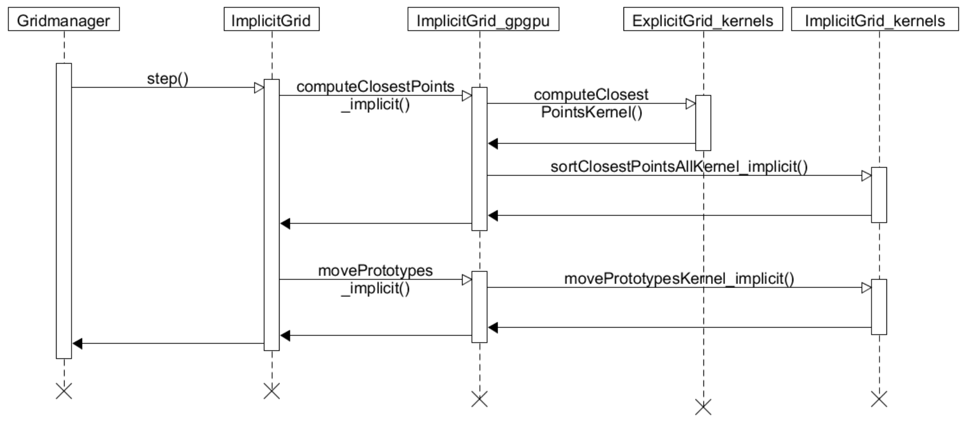
Gridmanager
ImplicitGrid
ImplicitGrid_gpgpu
ExplicitGrid_kernels
ImplicitGrid_kernels
step()
computeClosestPoints _implicit()
computeClosest PointsKernel()
sortClosestPointsAllKernel_implicit()
movePrototypes _implicit()
movePrototypesKernel_implicit()

Figure 2.7: The procedure to approximate the Voronoi points for each prototype. The horizontal boxes contain the class names which calls the methods on the vertical lines. The vertical boxes show when a method starts and responds.

During the second step in Fig: 2.7, the position of the prototype is optimized to cover as much volume as possible. To achieve that, the closest point to the prototypes is calculated. The closest point is either on a polygon or on a sphere. Once the closest point for each prototype is calculated, it is pushed away from that point. This process continues for a configured amount of iterations, each push for each iterations covers slightly less distance with a cooling function [Wel12]. By doing this, the Voronoi points which are covering the most volume for that current step in the current area are approximated.

2.11.5 Convert prototypes to spheres

For the last step Fig: 2.8 several criteria are checked before they are converted to actual spheres.

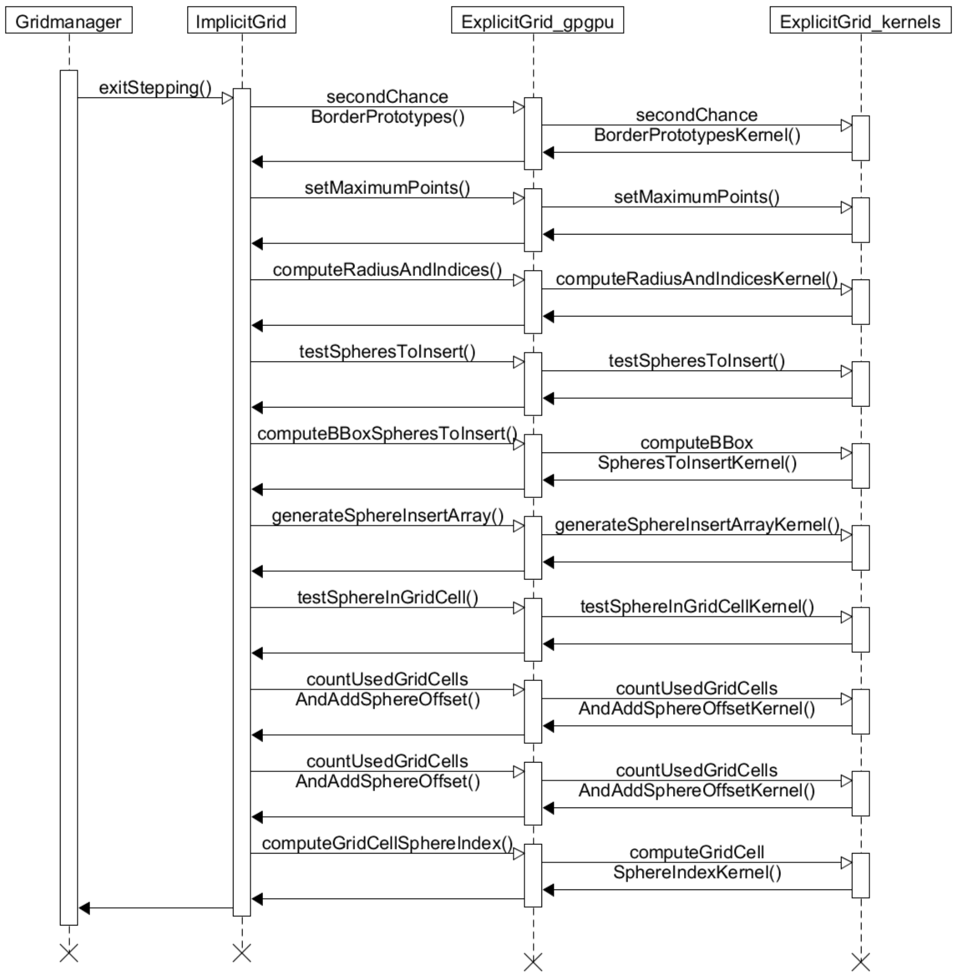
Gridmanager
ImplicitGrid
ExplicitGrid_gpgpu
ExplicitGrid_kernels
exitStepping()
secondChance BorderPrototypes()
secondChance BorderPrototypesKernel()
setMaximumPoints()
setMaximumPoints()
computeRadiusAndIndices()
computeRadiusAndIndicesKernel()
testSpheresToInsert()
testSpheresToInsert()
computeBBoxSpheresToInsert()
computeBBox SpheresToInsertKernel()
generateSphereInsertArray()
generateSphereInsertArrayKernel()
testSphereInGridCell()
testSphereInGridCellKernel()
countUsedGridCells AndAddSphereOffset()
countUsedGridCells AndAddSphereOffsetKernel()
countUsedGridCells AndAddSphereOffset()
countUsedGridCells AndAddSphereOffsetKernel()
computeGridCellSphereIndex()
computeGridCell SphereIndexKernel()

Figure 2.8: The procedure to insert the biggest prototypes as spheres for the packing. The horizontal boxes contain the class names which call the methods on the vertical lines. The vertical boxes show when a method starts and responds.

At first, each prototype is checked if the difference between two neighbor cells is close to 0; in this case it is discarded. The remaining prototypes are now sorted by their radii. Each prototype which is still valid is checked against all other prototypes and if two prototypes intersect, the smaller one is discarded. At this point all remaining prototypes could be inserted, but since the second step also produces small prototypes for several reasons, which are covered in 4.2.4, there is an additional condition. The volume of each prototype is compared against the biggest prototype for the current iteration. All prototypes which are too small are discarded, the biggest one is guaranteed to become inserted. Too small is defined as a volume threshold compared to the biggest prototype. The remaining prototypes are inserted, also all grid cells are updated, which intersect with the new spheres. If this is the case, it is added to the intersection sphere list for future iterations.

2.11.6 Loop

The initialization 2.11.3, optimization 2.11.4 and conversions steps of the prototypes 2.11.5 are looping in the GridManager class until the amount of added spheres exceeds the configured amount. For previous theses, the amount of spheres usually was set to 200,000.

Chapter 3

Measuring Packing Quality

In this chapter, several visualizations and metrics are introduced to measure the quality of the sphere packing because any improvement needs a measurement to be confirmed. First, 3-d rendered sphere packings will be discussed in 3.1. Additionally, 2-d representations for various aspects are presented: sphere concentration 3.4.1, voxel coverage 3.4.2 and density of the covered volume 3.4.3. Aside from the 2-d representations, multiple graphs are applied to the packing: filling curve 3.4.4, amount of theoretical required spheres 3.4.6, distribution of the sphere sizes 3.4.7 and entropy 3.4.5. The results of the distribution of the sphere sizes can also be used to classify the models by using the Kullback-Leibler divergence and cosine similarity, 3.4.8. The results of the measurements are used to improve ProtoSphere in 4.

3.1 3-d Rendering of Sphere Packings

Any 3-d sphere packing can be rendered as a 3-d visualization, but since there is no way for a human to estimate and sum up the volume of thousands of spheres, this representation contains no information of the covered volume. Additionally, any representation is displayed on a 2-d screen which uses a perspective viewport to simulate 3-d effects; even in Virtual Reality, only the front side is visible and therefore leads to the same issues.

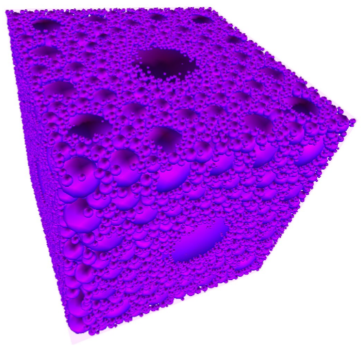

Figure 3.1: A 3-d perspective of a cube with the majority of the volume covered with a 3-d sphere packing which was created with ProtoSphere. [WZ10b]

Aside from the simulated 3-d effects for a cube with a length of 1, the smallest spheres of a 200,000 packing have a radius of 0,0008 which would require a resolution of at least 1250x1250 pixel for the smallest spheres to cover at least a pixel. In addition, visible spheres block the view of non-visible spheres and even if the spheres were slightly transparent, multiple layers of pixel sized spheres would produce visual disturbances. Therefore, assuming the outer layer of the object is packed and blocks the view of all inside spheres there is no way to tell if the inside is packed at all. Furthermore there is also almost no visual difference between 80% and 90% volume coverage. This gets even more difficult for the difference of 90% to 91%. A 3-d visualization of the sphere packing just shows a visualization of visible data, while 2-d representations can extract one aspect like sphere concentration or the thickness while all other aspects are ignored.

3.2 Quality criteria

To rate the packing quality, criteria must be set. For general physical applications, common criteria are computation time and precision. Sphere packings

as well have several meaningful criteria which depend on the application, like the covered volume. For this thesis, the goal is to enable the ProtoSphere configuration to target the following criteria: 1) cover the most volume with as few spheres as possible 2) cover as much volume as possible in a short time 3) reduce the computation time as much as possible even if less volume is covered 4) cover as much volume as possible regardless of the number of spheres.
Since optimization requires more computation time than greedy approaches, there is no configuration which targets high volume coverage and a fast packing. Therefore, reasonable configurations are considered, which are discussed in 6.1.

3.3 Used Models

For this thesis, nine different 3-d models are used: Ateneam Fig: 3.9, Armadillo Fig: 3.6, Bunny Fig: 3.5, Cow Fig: 3.7, Cube Fig: 3.8, Cylinder Fig: 3.3, Dog Fig: 3.2, Dragon Fig: 3.4, Pig Fig: 3.10.
The Armadillo, Cow, Dog and Pig were chosen as humanoid models with a torso, 4 to 5 limbs and a head. Regular models like the Cube and the Cylinder were chosen as geometrical primitive models, which have regular shapes. The Ateneam, Bunny and chinese Dragon were chosen as irregular models. Each model has different characteristics which have different effects on the sphere packing. The major characteristics are:
Majority torso: Ateneam, Bunny, Cow, Pig
Majority limbs: Armadillo, Dragon, Dog
Big limbs: Armadillo, Pig, (Dragon)
No small connections: Ateneam, Bunny, Cube, Cylinder

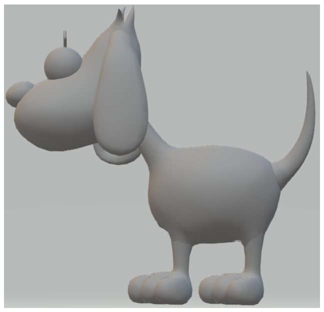
Figure 3.2: Dog

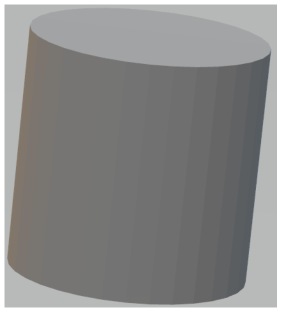
Figure 3.3: Cylinder

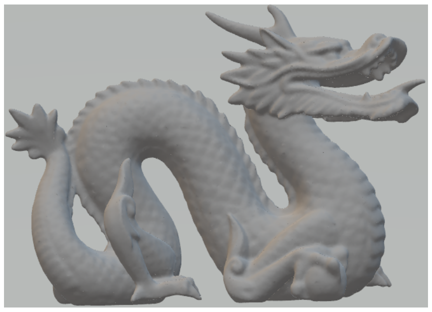
Figure 3.4: Dragon

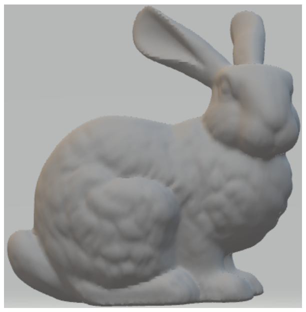
Figure 3.5: Bunny

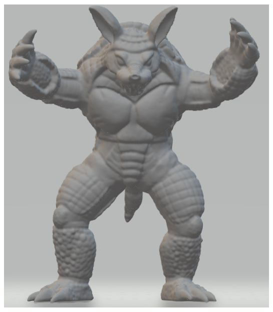
Figure 3.6: Armadillo

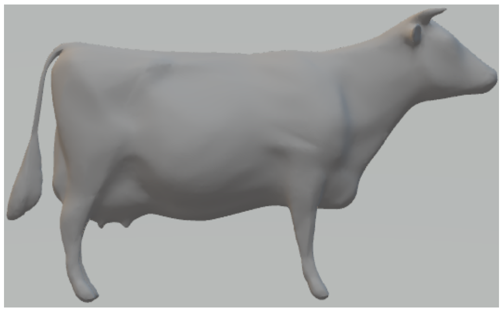
Figure 3.7: Cow

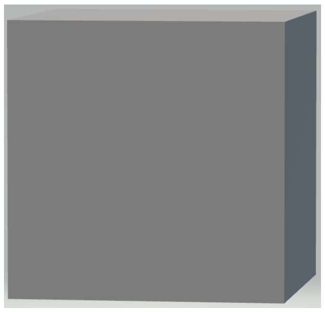
Figure 3.8: Cube

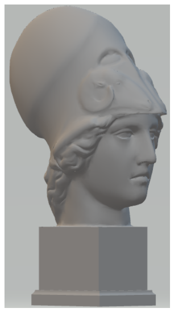
Figure 3.9: Ateneam

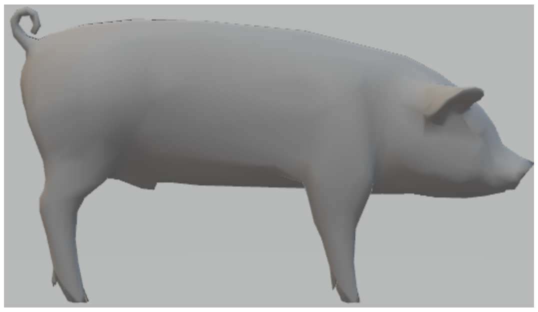
Figure 3.10: Pig

3.4 2-d Representations

3-d representations have several issues for quality analysis 3.1. Therefore, other representations are required which target the issues and visualize information in a format which is more human readable. To achieve that, 3-d representations are simplified to 2-d representations by calculating first the 3-d result and then summing up all layers of one axis to a single layer which is shown. Additionally, there are also possibilities to quantify the packed spheres by filling curves, histograms and simple numbers.
To visualize the 3-d sphere packing, three aspects are covered in this thesis: sphere concentration 3.4.1, voxel coverage 3.4.2 and density by combinding the results of the sphere concentration and the voxel coverage 3.4.3.
For the sphere concentration visualization, the idea of thresholding was used [SCS13] in which the background is separated from the x-ray image by applying thresholds to the sphere center matrix.
To calculate the voxel coverage, the basic idea of the algorithm of [BHMv14] was applied to the packing. The algorithm was applied to point clouds which marks cells. Since ProtoSphere does not work with point clouds, some adjustments were necessary to calculate a voxel map for the packing.
As a final step a density map is created by combining the sphere concentration and voxel map. The voxel map is divided for each entry by the concentration map to calculate a density map.
In this chapter there is one diagram per aspect which shows the XY view.

3.4.1 Sphere concentration

The map shown in Fig: 3.11 represents the distribution of sphere center concentration. To calculate this map, a matrix with a cubic size is computed. Each cell is filled with the number of centers inside but ignores the radius. The basic idea for this representation resembles the threshold algorithm in [SCS13].

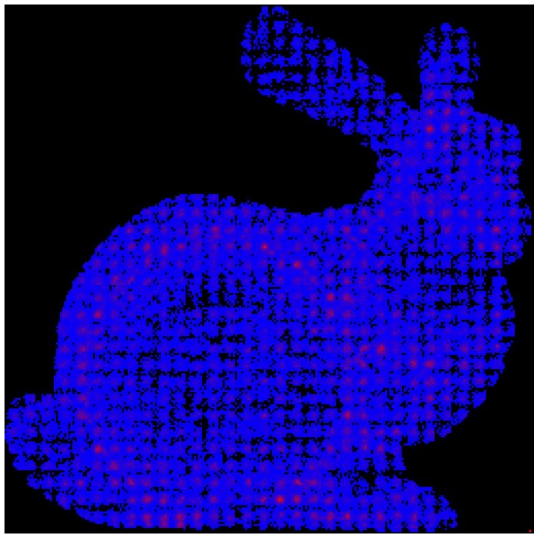

Figure 3.11: Shown is the distribution of the sphere centers over the area of the Bunny model Fig: 3.5. Areas without spheres centers are printed black, red areas contain the most sphere centers, blue areas contain relative small amounts of sphere centers compared to the red areas.

The map is generated by generating a 3-dimensional matrix, which counts the number of sphere centers for each cell. For the visualization all layers are summed up in a 2-d matrix which is used to generate the Fig: 3.11. The regular rectangular patterns are covered in 4.2.2.

3.4.2 Voxel coverage

In Fig: 3.12 the voxel coverage for the packing is shown. To calculate this map, a matrix with a cubic size is computed again. The basic idea is to create an a adjusted version by generating voxel coverage of a point cloud [BHMv14]. Each cell is treated as a voxel and checked for a random position to be inside of a sphere; in case of yes, the cell is marked as a covered voxel, otherwise its marked as not covered at all. That method has a minor error but on average this error is compensated by the law of big numbers.

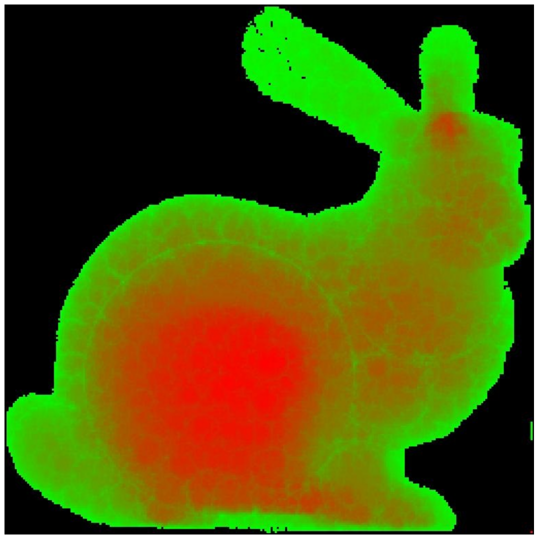

Figure 3.12: Shown is the distribution of the covered voxels of the Bunny model Fig: 3.5. Areas without any voxels are printed black, red areas contain multiple layers of covered voxels, green areas contain few layers of covered voxels compared to the red areas.

This representation can be validated against the silhouette of the model if there is any area covered outside of the silhouette, obviously invalid spheres were inserted. On the other hand, there also is a visual check if thick areas appear red, compared to thin areas in green. For the Bunny the expectations hold, the thick torso appears red, while the slightly thinner upper body appears in light red and thin ears appear green.

3.4.3 Density

The Fig: 3.13 shows the density differences over the area which should not be confused with the global packing density 2.10. To calculate the density the results of the sphere concentration and voxel coverage are combined by the division of the voxel coverage map by the sphere concentration map for each cell. For the case that the concentration map does not cover an area which is covered by voxels, that area in the concentration map is interpreted as low concentration, instead of no concentration, for a meaningful representation of the density map. This effect usually occurs near to the center of the biggest spheres of the entire packing. For a good packing, this coloration of the model is colored similarly over the entire area.

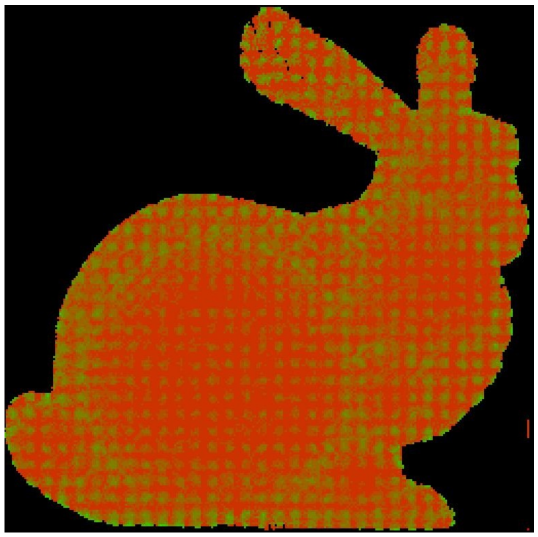

Figure 3.13: Shown is the distribution of the density which is calculated by dividing the voxel coverage and sphere concentration of the Bunny model Fig: 3.5. Areas without any density are printed black, green areas indicate a large number of spheres centers compared to the covered voxels, red areas indicate a high voxel coverage compared the the number of sphere centers.

The regular patterns which already appeared in the sphere concentration map 3.4.1 are visible again. This issue is covered in 4.2.2.

3.4.4 Filling curve

In Fig: 3.14, the covered volume with a growing of number spheres is shown. The major purpose of that curve is to get an idea of how much volume is covered with a growing number of spheres.

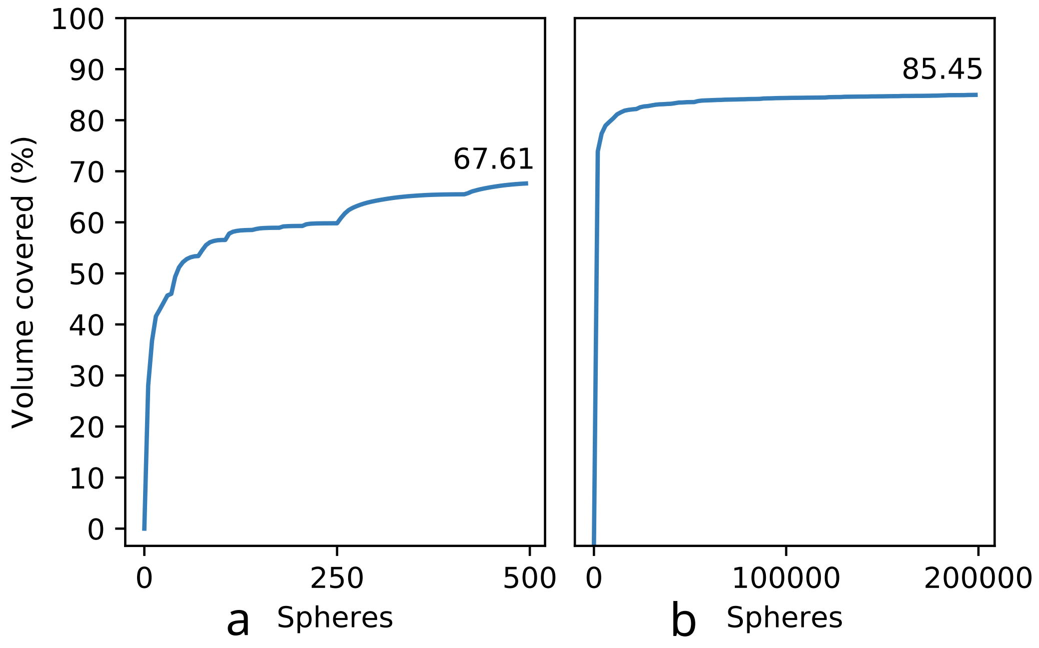

Figure 3.14: Shown is the filling curve of the covered volume in % for an increasing number of spheres. On the x-axis the number of spheres, on the y-axis the percentage of covered volume a) Curve over the first 500 spheres b) Curve over all spheres

Since the majority of the volume is covered with the first 100 to 500 spheres, the curves are distributed in a) the first 500 spheres of the packing and b) the entire packing. Interestingly in a) the curve jumps; that effect occurs because depending on the spawn position of the prototype, some empty areas are missed and covered in later stages when the prototype spawned on a better position 4.2.4.

3.4.5 Entropy

The entropy indicates how much independent information a set of data contains [Sha48]. To calculate the entropy the formula $E = \sum_i^N p_i \cdot \log p_i$ is applied on any p_i with $\sum p_i = 1$. The interpretation of the entropy is how fast the volume of a model is covered with spheres; a low entropy value indicates the spheres contain on average very little information, a big entropy value indicates on average more information2.5. As example: the Cube requires a small amount of big spheres to cover more than 80% of the volume, with the first sphere covering already more than 50% of the volume, while an humanoid object like the Pig requires more spheres to cover the same amount of volume, therefore the entropy of the Cube is lower than the entropy of the Pig since the Pig needs more spheres to cover the same percentage of volume. To apply the entropy on a packing, the packing is split into 100 parts, while the first part is the first sphere, which should be the biggest sphere. On the other 99 parts all other spheres are distributed evenly. To calculate the p, the volume of every part is divided by the total covered volume 3.1.

	Armdl.	Atnm.	Bunny	Cow	Cube	Cyl.	Drgn.	Pig	Dog	$\bar{e}$
entropy	0.226	0.187	0.216	0.173	0.208	0.154	0.207	0.162	0.214	0.194

Table 3.1: Entropy of the packing for the parallized state. The bigger the entropy, the less volume is covered by early spheres while late spheres cover more volume.

3.4.6 Theoretical number of required spheres

With the curve shown in Fig: 3.15, the theoretical number of required spheres is shown. To calculate those values, the linear regression $y = m \cdot x + b$ is computed for every 500 spheres. Therefore, the linear regression aligns every step more to the almost linear curve for later stages of b) in Fig : 3.14. To calculate the theoretical amount of required spheres, the slope is assumed constant and the number of spheres which are required to cover 100% of the volume is computed by $x = \frac{100 - b}{m}$. The result x gives a prediction of how many sphere would be required to cover 100% of the volume.

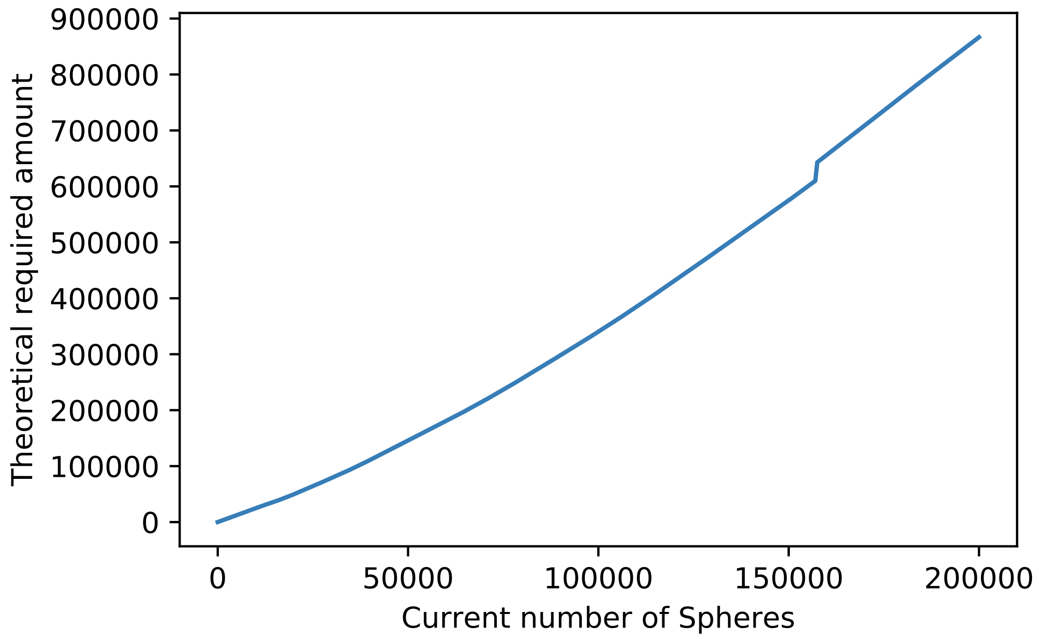

Figure 3.15: Shown is the theoretical number of required spheres. On the x-axis the current number of spheres is shown. The Y-axis shows the theoretical number of spheres with the assumption that the slope does not decrease anymore.

However, since the slope in the curve 3.15 decreases over time, any regression is an optimistic assumption which never holds. This becomes obvious by inspecting some values. For 29,000 spheres about 60,000 spheres would be required with the current volume growth slope, for 60,000 actual spheres about 140,000 spheres would be required, for 140,000 spheres 310,000 spheres would be required and so on.
The factor of current spheres and theoretically required spheres increases, this effect is verified by computing $n = \frac{y}{x}$ with bigger x-values, the n value increases slightly in the range of $2.0 \leq n \leq 2.4$ for the observed data. Therefore, this evaluation gives the idea that the packing most likely will never hit 100% which is also supported in [SDSRH12]. Depending on the model, the slope has different values; still, the slope decreases almost to $m = 0$ with a

growing number of inserted spheres.

3.4.7 Radii of the spheres

In Fig: 3.16 a distribution for the radii of the spheres is shown. To calculate the histogram, a exponential function with the formula $f(r) = a * e^{-k*r}$ is calculated to fit for the biggest to smallest sphere. The function is split into 50 parts, on which the spheres are distributed depending on the size.

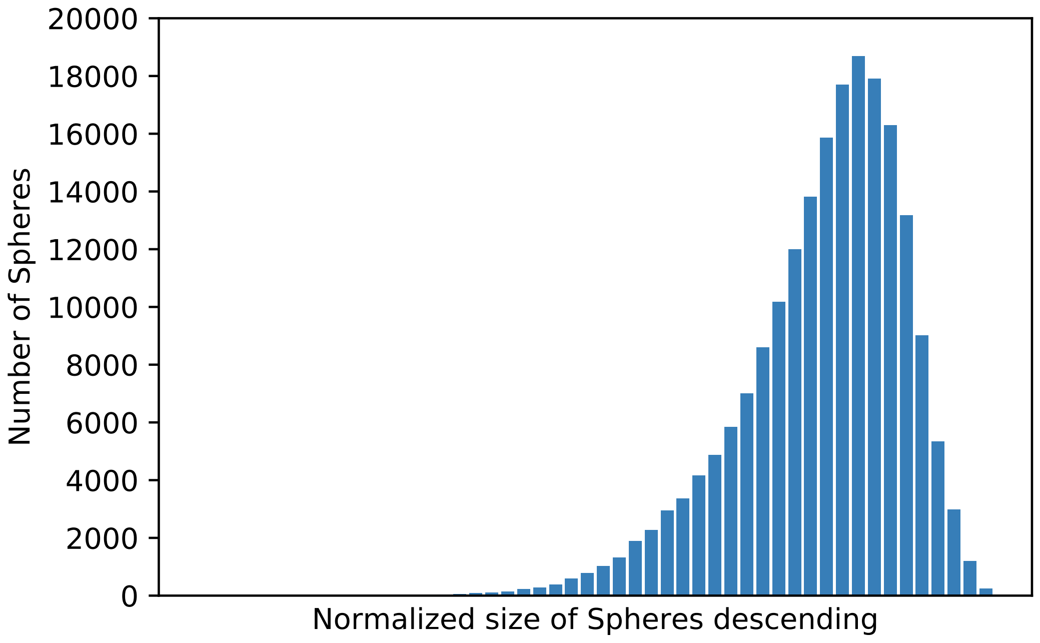

Figure 3.16: In this histogram, a distribution of the sphere radii is shown. The histogram is ordered from left to right by the descending radius of the spheres. The distribution has an early peak because of the great tolerance of the inserted spheres for each step 4.2.1.

The distribution in Fig: 3.16 is verified by investigation, sphere radii between two steps (sphere 193824 to sphere 193827, radii: 0.000666606; 0.000665818; 0.00927337; 0.00832588). There is a major difference between the radii of those spheres, even if the slope of the filling curve is close to 0. Fig: 3.15

3.4.8 Similarity

The distribution of the histogram 3.16 for the sphere radii can also be used for other applications like cosine similarity.

$$\cos\alpha = \frac{d_1 \cdot d_2}{|d_i| \cdot |d_2|}$$

The results can be used to compare the sphere packings of different models. The cosine similarity is a simple method to compare two sets with each other by calculating the resulting vector for each set and comparing the angles. In contrast two equal sets produce parallel vectors, which have an angle of 0° between each other, which both are identical. Two sets which have no values in common produce orthogonal vectors with a angle of 90° which leads to no similarity. Applied on the initial state on ProtoSphere, the similarity is the following.

	Ateneam	Armadillo	Dog	Cow	Cylinder	Cube	Dragon	Bunny	Pig
Ateneam	1.000	0.998	0.968	0.952	0.996	0.994	0.931	0.976	0.930
Armadillo	0.998	1.000	0.976	0.961	0.999	0.988	0.940	0.964	0.940
Dog	0.968	0.976	1.000	0.997	0.978	0.949	0.989	0.900	0.990
Cow	0.952	0.961	0.997	1.000	0.964	0.933	0.997	0.877	0.997
Cylinder	0.996	0.999	0.978	0.964	1.000	0.984	0.943	0.957	0.943
Cube	0.994	0.988	0.949	0.933	0.984	1.000	0.910	0.991	0.911
Dragon	0.931	0.940	0.989	0.997	0.943	0.910	1.000	0.850	0.999
Bunny	0.976	0.964	0.900	0.877	0.957	0.991	0.850	1.000	0.850
Pig	0.930	0.940	0.990	0.997	0.943	0.911	0.999	0.850	1.000

Table 3.2: Shown is the similarity of the initial sphere packing. Each model is compared to other model, 1.0 for an almost equal sphere distribution, decreasing with less equal distributions to 0.0 for no similarity at all.

All similarities of 3.2 must be multiplied by 100 to get the percentage similarity, each model has as expected a 100% similarity to itself. The lowest similarity is about 85% and the Cube and Cylinder have a 99% packing similarity and more to some models, which have no similarity to mathematical shapes like an Armadillo. For the initial state, the comparing of the packing models does not give reliable results, which correlate to the classification of the characteristics in 3.3. A Pig has 99.9% similarity with a chinese Dragon while a Cow and an Armadillo have about 96.1% similarity. On average most packings are over 90% similar, while correlations between shapes are hard to find or do not make any sense. Therefore the sphere packings which are generated by the parallelized state of ProtoSphere can not be used to classify models. Another technique also exists to compare sphere packings [Alb17].

$$S = 1 - \frac{\sum_i |n_i - n_i^{\text{ref}}|}{2(N-1)}$$

This formula has several issues which makes it hard to apply to a packing of an arbitrary model. First, both packings must contain exactly the same amount of spheres. Second, if two models are scaled differently, this formula would spot a difference even if both packings are equal except for the scaling. Therefore this formula will not be used, since even if the same model is used, no reliable results are produced.
On the other hand, the difference between the packings can be calculated with the Kullback-Leibler divergence [BDA11].

$$D(g||B) = \int |g(x) \log \frac{g(x)}{b(x)}|$$

Compared to the similarity, the divergence indicates which models do not have similar characteristics. Therefore, small values indicate that models have a similar packing, while similar bigger values cannot be used to indicate a similarity, since there is no way to determine if the divergence was caused by the same values.

	Ateneam	Armadillo	Cylinder	Cow	Cube	Dragon	Bunny	Dog	Pig
Ateneam	0	0.26	0.32	0.52	0.34	0.56	0.26	0.09	0.37
Armadillo	0.21	0	0.52	0.24	0.50	0.26	0.44	0.14	0.13
Cylinder	0.38	0.74	0	1.05	0.14	1.15	0.08	0.50	0.83
Cow	0.40	0.22	0.70	0	0.67	0.05	0.60	0.35	0.10
Cube	0.50	0.90	0.17	1.20	0	1.33	0.16	0.63	0.94
Dragon	0.41	0.23	0.71	0.05	0.68	0	0.61	0.36	0.11
Bunny	0.32	0.66	0.09	0.95	0.14	1.05	0	0.43	0.74
Dog	0.09	0.16	0.40	0.42	0.40	0.45	0.33	0	0.28
Pig	0.32	0.13	0.62	0.11	0.59	0.13	0.53	0.26	0

Table 3.3: Shown is the divergence of the initial sphere packing. Each model is compared to any other model, 0.0 for an almost equal sphere distribution, increasing with less equal distributions.

In 3.3 the divergence is shown. Again each model has no divergence to itself. The difference of the models ranged from $0.05 \leq D \leq 1.33$ with a Cow and Dragon having the least divergence while the cube and dragon have the biggest divergence. Compared to the cosine similarity, the Kullback-Leibler divergence indicates that reliable results might exist.

3.5 Hypothesis

Since the visualizations give the idea that the packing algorithm does not work properly, a further investigation for the hypotheses makes no sense at this stage. Once the cause for the regular patterns is found, the hypotheses are challenged again in 5.5.

3.6 Summary

In this chapter several human readable aspects to measure sphere packing have been introduced, while the disadvantages of a 3-d visualization were discussed. The different aspects are: sphere concentration, voxel coverage, density, filling curve, entropy, the theoretical amount of required spheres and the distribution of the radii. In addition the cosine similarity on the distribution map was computed to measure the similarity of different sphere packing.

Chapter 4

Improving ProtoSphere

After defining metrics to measure the current quality of the 3-d sphere packings, in this chapter all issues, fixes, mitigations and possible enhancements are investigated. Any issue which was found either on decisions for the implementation of ProtoSphere or general issues which leads to lower packing quality is investigated for the cause and effect. If possible, a fix or mitigation for the issue was implemented. In this case, other data structures are necessary and possible enhancements are described. Additionally all configurations to adjust and improve the packing for several aspects are explained.

4.1 Initital State

Since the initial version developed in [Wel12], there were adjustments made resulting in a loss of about 10% of the volume coverage. One important adjustment was the parallelization of the algorithm [Teu13].
In the Figures 3.11 and 3.13 regular patterns are visible which may indicate the reason for the loss of the covered volume. Additionally the difference of the volume coverage for the models has a range of about 12% as shown in 4.1, while no volume coverage has a value close to 95.01% 2.2.4. Except for the covered volume, computation time and 3-d representation, no metric

was measured.

	Armdl.	Atnm.	Bunny	Cow	Cube	Cyl.	Drgn.	Pig	Dog	$\bar{v}$
volume	83.14	84.73	85.45	83.28	89.47	91.18	80.94	84.03	79.23	84.60

Table 4.1: The table shows a comparison of the covered volume for the sphere packings generated by the initial state of ProtoSphere.

4.2 Issues

4.2.1 Radii of inserted spheres

The first issue was an assumption on the inserted prototypes. The non-parallelized implementation just inserted a single sphere for each step. When the algorithm was parallelized, a huge amount of potential prototypes were generated for each cell. To reduce the amount of small prototypes, an additional condition to be met by each prototype was set. To filter the prototypes, every prototype is compared to the biggest prototype which is inserted without fail each time. For all other prototypes, two conditions must be met. The first condition is that the prototype must have a bigger radius than the lowest configured radius. The second condition is to have at least $\frac{1}{10}$ of the radius of the biggest prototype, this condition was set during the implementation in [Teu13]. Any prototype which meets both conditions is inserted if it is not overlapping with another prototype. The tolerance led to the big distribution depicted in Fig: 3.16
The factor of a tenth for the radius was chosen randomly and leads to a factor of 1000 for the volume difference of the biggest and smallest sphere for each step. Especially during early iterations, on which lots of spheres optimize their position inside the grid cells which have no intersecting sphere yet, lots of non-Voronoi points are found. By inserting the non-Voronoi prototypes as actual spheres, the empty area requires more spheres to cover the same amount as the Voronoi prototype would have covered as a single sphere.

4.2.2 Coverage over certain cells

As already mentioned in 3.4.1 and 3.4.3, some cells are overfilled, while some are ignored. This issue causes an uneven distribution of volume coverage, which leads to areas with high coverage and areas with low coverage. For applications like remote operations, this could be fatal since the surgeon cannot rely on the haptic feedback.
The cause for this problem is a wrong assumption on the decision-making algorithm for the selected implicit cell. Cells can have four different states: Empty, Inside, Border, Outside. Inside cells are completely inside a single sphere, border cells, are either partly covered by spheres or hit the surface. Outside spheres have no intersection with the surface. The initial algorithm always chooses the first border cell.
After some iterations, eventually almost every cell inside of the models is marked as a border cell, which leads to the pattern. The decision making always has a set of cells which is chosen for the current iteration. The cell-picking process follows the following line of priority: first empty cells over border cells, in case any cell is marked as a border cell, the first border cell is picked. After at most eight steps, any cell was chosen as empty cell and is marked as border cell, therefore after at most eight steps just one cell was chosen for the packing all the time. The surrounding cells are covered slightly by a sphere which intersects, but eventually all positions of intersecting spheres are used and finally no more volume can be covered inside of the cells which can not be chosen anymore.

4.2.3 Prototype-polygon intersection

Prototypes are spawned in the implicit cells before those are checked to have an intersection with other polygons or spheres. There are several possibilities for the spawning position: inside of the model without any intersection, outside of the model, inside of a sphere or direct in a polygon. For the last case, floating point precision can produce the wrong direction if the prototype is

interpreted from the outside.
Another effect occurs if a prototype spawns in the middle of either two polygons or spheres whose normals almost point in the opposite direction as in Fig: 4.1. The prototype has a limited amount of steps and cannot leave the current position. Therefore, the prototype will not come close to its Voronoi point. In this case, computation time is consumed for bad results.

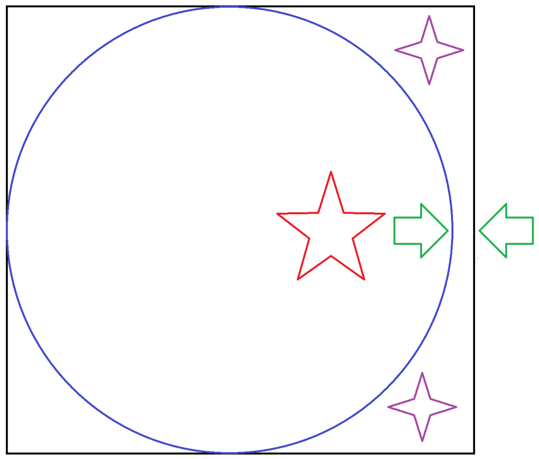
Figure 4.1: A prototype (red star) inside a sphere (blue circle), which cannot converge towards a Voronoi point (purple stars), because the converging vectors point in opposite directions (green arrows.)

4.2.4 Non-Voronoi prototypes

In each iteration valid prototypes are computed over the entire grid. The prototypes are randomly distributed over the implicit grid and optimize the positions over a limited number of steps. During each step the distance the prototype can travel is reduced to approximate the Voronoi point. For prototypes which converge to the Voronoi points, the volume covered is optimal. During each step, some prototypes are spawned on positions for which the number of steps and the cooling function is too low to come close to the Voronoi points and stop midway; the effect is similar to the prototypes which cannot leave its area 4.2.3.

However, prototypes which are not converged to Voronoi points are still inserted if the volume is not too low compared to the biggest sphere for the current step. Since Voronoi points always cover the most volume, any inserted prototype which is not close to the Voronoi point decreases the volume because more spheres are required to cover the same volume.

4.2.5 Low coverage at surface

In Fig: 3.13 a density map is shown, which has a low volume coverage at the surface. This distribution causes an uneven volume coverage for the inner parts and the outer parts, which leads to a lower precision.
There are several causes for this effect. Depending on the shape of the model, this effect is worse or almost does not matter. For the shape of the model, three different cases are investigated.

4.2.5.1 Convex models

For convex shapes like Fig: 4.2 ProtoSphere has the best performance, the spheres align to the surface and cover a lot volume with a few spheres.

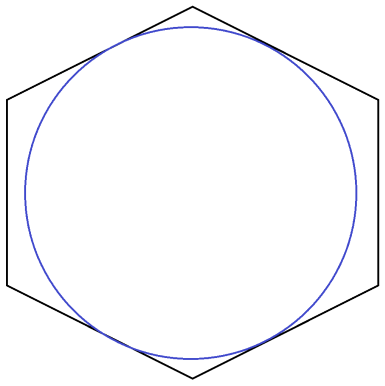

Figure 4.2: A sphere inside a convex model, which covers the majority of the inside volume. Inside the red box a shape which has a recursive pattern is highlighted.

4.2.5.2 Concave models

For concave shapes like Fig: 4.3 ProtoSphere produces recursive patterns during the packing process. Inserted spheres leave a lot of uncovered volume; new spheres reduce that uncovered volume, however spheres cannot align to the surface if there some uncovered space is left. This effect is also the long-term effect when a huge amount of spheres are inserted, and might occur recursively. The remaining void is concave for new prototypes.

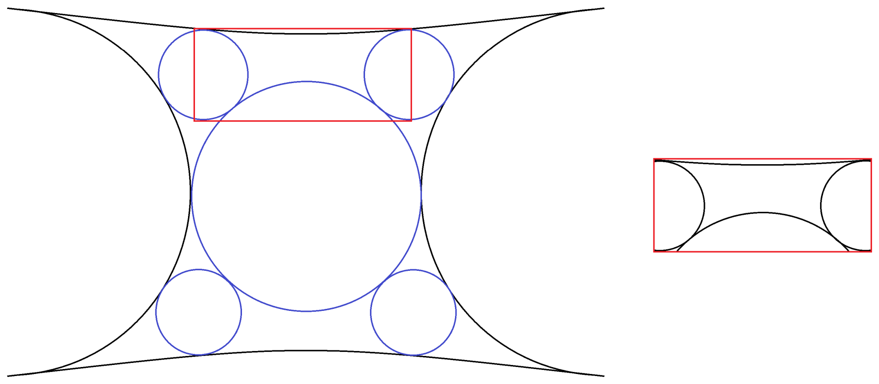

Figure 4.3: Multiple sphere inside a concave model, which cover about half of the volume. Inside the red area a recursive pattern appeared.

4.2.5.3 Flat surfaces

Flat surfaces like in Fig: 4.4 behave similarly to concave models. Inserted spheres leave uncovered volume and cannot align to the surface.

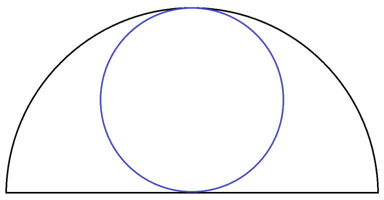

Figure 4.4: A sphere inside a model with a flat surface.

4.2.6 Big gridcells and tiny parts

Grid cells perform well on any model, but there are some cases when the layout of the cell leads to minor issues. If the cells are completely inside of the model, it performs well. However, for partly covered cells the algorithm has a minor issue, a chance to spawn an inside prototype is the same as the amount of covered volume of the implicit cell. In case of an edge which can also be covered by spheres, that spawned in other cells, the effect is small. In the case of a small part, like an antenna or finger, the covered volume is decreased, if e.g. 7% of the cell is covered by the thin part, on average every fourteen steps a valid prototype is spawned, while the prototype is discarded otherwise.

4.2.7 Different diameter

The tolerance factor mentioned in 4.2.1 defines the lower bound over the entire model. If the model has parts with different diameters, like humanoid objects, this factor will discard any sphere in the limbs, unless the prototype in the torso are similar in size to the diameter of the limbs, like in Fig: 4.5. Except for the connection areas, no prototype on each area can intersect with each other, accordingly the global tolerance factor increases the overall computation time since good prototypes are discarded until they are inside the tolerance bounds.

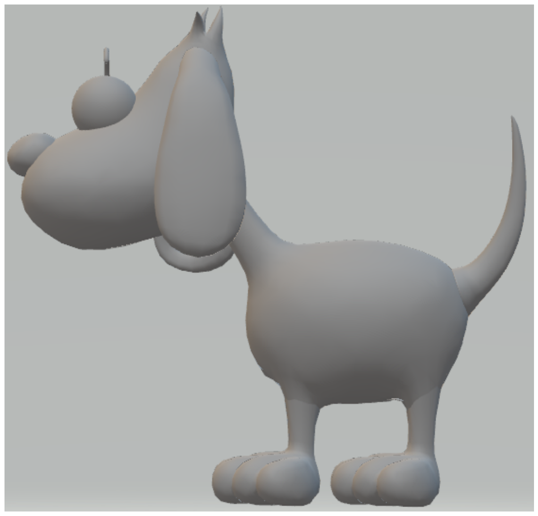

Figure 4.5: A dog, on which the paws, the head and torso can be packed independently.

4.2.7.1 Outside spheres

If parameters *eig* and *sg* are configured with the default values, all packings are valid. Adjusting these configuration enhances the packing speed. However for some configurations, invalid sphere packings are generated which contain outside spheres, like in Fig: 4.6 in which a sphere on the front legs appeared outside of the model, the same effect occurs on the tail. The configuration was added in prior versions and should always be used with caution.

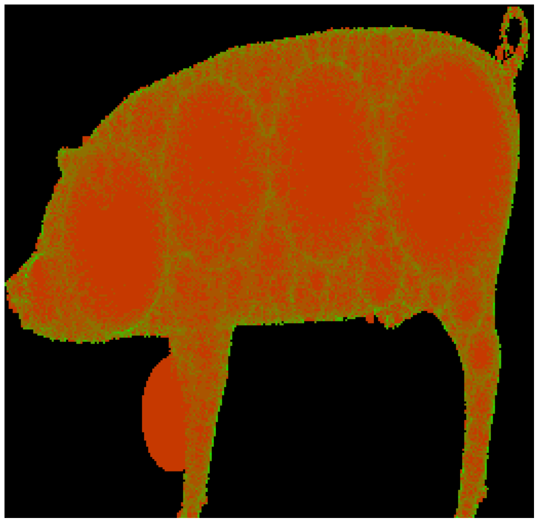

Figure 4.6: A pig packed with spheres, on which several outside spheres appeared because of a bad configuration.

4.3 Mitigations

In this section the mitigations which were found for discovered issues in 4.1 are introduced. The mitigations either reduce the amount of trivial operations or enable ProtoSphere to enhance the distribution of the prototypes to cover more volume. If the issue is caused by a configuration which leads to a trade-off, for example precision and computation time, additional configurations are added to target either aspect.

4.3.1 Mitigation: radii of inserted spheres

In the parallized version, a tolerance factor of 10 for the biggest and smallest radius during each step was implemented. This tolerance factor was required as simple heuristic to not insert any sphere. However this factor of 10 for the radius lead to a volume tolerance of 1000, since sphere volume is computed with r^3. To mitigate that issue, the tolerance factor of 10 was replaced by two new parameters **minsd** and **maxsd**, which configure the minimal and maximal tolerance for the volume of the prototypes.

4.3.2 Mitigation: coverage over certain cells

To prevent ProtoSphere from generating uneven distributions depending on the cell state, the algorithm which always chose the first border cell was adjusted by choosing randomly between the implicit cells. This leads to the following graphs.

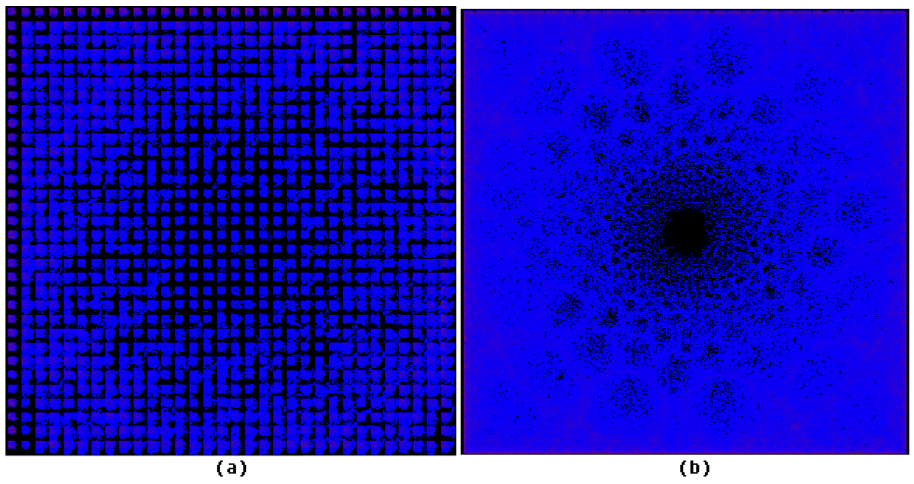

Figure 4.7: Shown is the enhancement on the static starting cell for prototypes. a) Before adjustment one of eight cells (3-d 2x2x2) was selected to insert prototypes every time b) After adjustment, every cell is selected evenly distributed for prototype insertion

In 4.7 the parallized version and the adjusted version are compared a) the parallelized version distribution is shown, for b) the distribution after the mitigation by picking random cells.
The distribution now is more even. There are also recursive patterns for each side, which is expected since the distribution should be almost equal. The covered volume for all models is also increased by mitigating the volume

tolerance factor for inserted sphere from 1000 to a configurable value, which is explained in 4.5.2. In general, the covered volume is raised from an average of 82-87% to 89-94%. The results will be discussed in 6.

4.3.3 Mitigation: prototype-polygon intersection

Prototypes which come too close to the surface are discarded for the run. The threshold is below a distance of 10^{-7}, which is the maximum precision for floats. In comparison, if a prototype has a lower radius than 0.0287941 it again leads to floating point errors since the volume of a sphere with a radius of $r \leq 0.0287941$, is less than the floating point precision. So any prototype which is discarded could not provide any relevant volume.

4.4 Possible Enhancements

In this section possible enhancements for issues which were not fixed or mitigated are explained.

4.4.1 Mitigation: Big gridcells and tiny parts

To solve that issue, the bounding box for the currently relevant cell was computed. But since every cell has information about relevant polygons for that area and even may contain multiple elements like multiple parallel fingers, some negative effects on calculated bounding boxes can occur. In the best case, the bounding box is valid, but there are also cases where the bounding boxes covered empty space between two polygons which led to an even worse performance since instead of a chance of 7%, all prototypes are invalid initialization points. There was no significant difference for valid boxes, for invalid boxes the coverage became worse. Therefore, this change was discarded.

4.4.2 Mitigation: Different diameter

The effect which was explained in 4.2.7 can be mitigated. A cluster algorithm which separates parts of the models which cannot have intersecting could determine which reference prototype in the cluster is used. By clustering the clusters are filled independently, and less suiting prototypes are discarded. The clusters can also be used to calculate finer grids for thin areas, and less trivial checks like intersection tests of prototypes of two non-intersecting clusters.

4.5 Improvements

In this section general improvements are introduced. The improvements enable ProtoSphere to adjust the packing depending on the application. Therefore, any introduced parameter is dynamic. The effect of the different configuration possibilities is discussed in 6.1.

4.5.1 Volume difference per step

The effect explained in 4.2.1 happens because the initial tolerance for inserted spheres during each step is too big. For mitigation a dynamic tolerance factor was implemented. To adjust the tolerance factor a lower and upper bound for the packed volume percentage for each step can be configured. If the covered volume for the current step is bigger than the upper bound, the tolerance factor is reduced, while it is increased if the covered volume is lower than the lower bound. The tolerance factor for inserted prototypes is initialized with the lower bound, explained in 4.5.2.

4.5.2 Volume tolerance of prototypes

As explained in 4.2.1, the tolerance factor of the volume of inserted spheres during each step is 1000. This factor was replaced by a lower and upper bound for the tolerance factor which is set by the packing configuration. Since the criteria, like computation time or covered volume, depends on these values, there is no optimal value to set. In general, a low tolerance leads to a slow packing with bigger spheres, while a high tolerance leads to fast packings with smaller spheres. The covered volume for the same amount of spheres is always higher for the slow configuration. Therefore, the lower bound of the configuration sets the initial tolerance when ProtoSphere starts, the implementation of 4.5.1 adjusts the dynamic tolerance depending on the current packing speed. To prevent the effect of 4.2.1, the upper bound clamps the dynamic factor. As a nice side effect, the upper bound also enables ProtoSphere just to consider the biggest sphere for the packing, which is the slowest configuration but produces the most covered volume per sphere.

4.5.3 Additional break criteria

ProtoSphere only allows a configuration to set the number of spheres, which must be packed until it stops. This feature is required, since there must be at least one criteria to stop the filling, but there are several other options to stop filling the model with more spheres.

4.5.3.1 Break on percentage covered

An easy criteria to implement is a configuration for the desired amount of covered volume percentage. The advantage for this criteria is that the covered volume is the same and does not depend on the model like the initial version 4.1, all models have a different number of spheres. A disadvantage of the approach is when the configured volume is higher than the maximal volume percentage which can be covered, the packing will never stop. To mitigate that case, the specified amount of spheres still stops the loop.

4.5.4 Break on a dynamic criteria

In addition to the static break criteria, a dynamic break criteria was implemented to stop the packing when the slope stops growing.

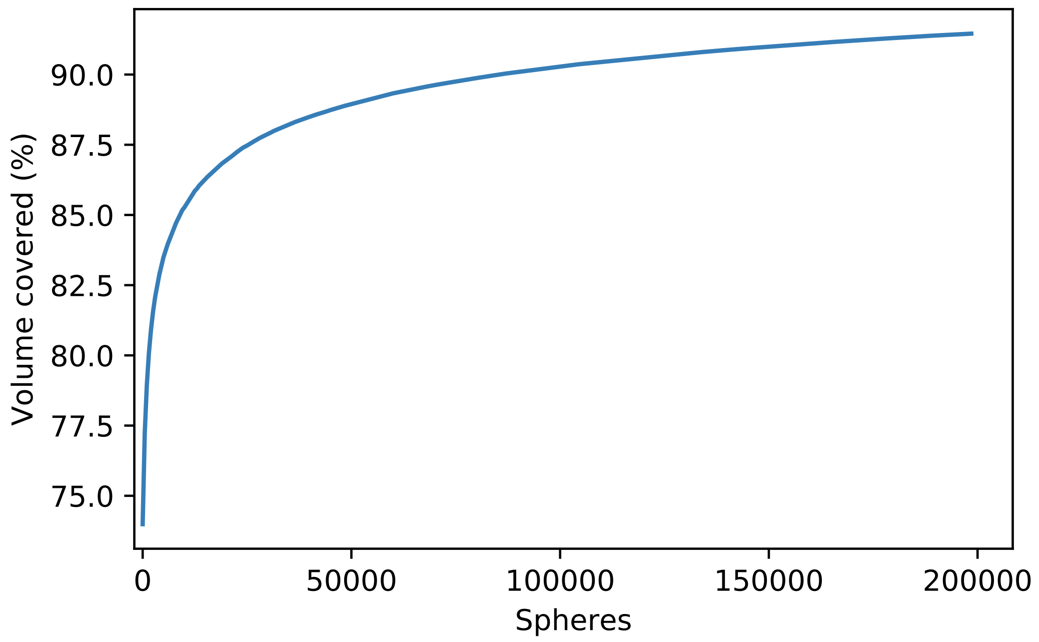

Figure 4.8: Shown is the filling curve of a sphere packing with a low slope at the end.

To achieve that, the first approach was to find a function which predicts the filling curve like in Fig: 4.8 for several models. This approach was not successful since no such function was found.

4.5.4.1 Approximation of the slope function

The final result references to characteristic all filling curves share; after about 100,000 spheres are inserted, all graphs have a low positive slightly decreasing slope. This characteristic is used to calculate a linear function with the method of least squares $y = m \cdot x + b$ which aligns to the curve after 100,000 spheres. To calculate this function, the packing is split into segments of 500 spheres. On the data set a linear regression is calculated [IM]. The calculation is repeated every 500 spheres, each iteration aligns better to the long-running function and therefore gives more reliable results.

4.5.4.2 Calculating a dynamic break point

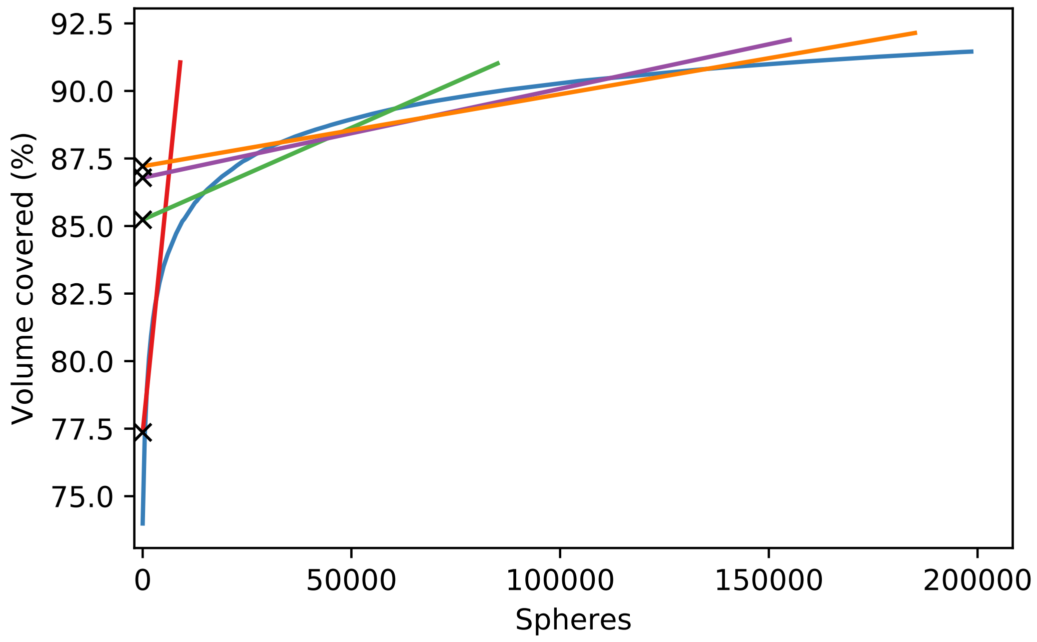

Figure 4.9: Shown is the filling curve of a sphere packing with multiple linear regression lines. Red: regression over first 4,000 spheres, Green: regression over first 80,000 spheres, Magenta: regression over first 150,0000 spheres, Orange: regression over first 180,0000 spheres with X-markers at the b offset values.

In the example 4.9 four regressions are calculated for 4,000 spheres; 80,000 spheres; 150,000 spheres and 180,000 spheres.

With $y = m \cdot x + b$ calculated for every 500 spheres which are inserted, there are two values, which can be used to find the best point to stop packing the model. The slope m is close to the limit of floating point precision, therefore the usage of this value is done with caution [Gol91]. Each function has different m and b values, which can be compared to its previous value to find a good criteria to stop the packing.

First m as the approximation of the slope for later stages of the curve. To find a good value to stop the packing, the first approach was the difference of two following m values. All values are positive since the slope decreases monotone. For the following comparison the floating point precision is ignored. The major problem of the difference is the difference between the slopes for each model. However models with a low entropy, on which the volume is covered fast with very few spheres, have significant lower slope values than models with a high entropy 3.1. In additionn the slope is also close to the limits of floating point precision [Gol91]. Therefore, this approach is dropped.
The next approach is to calculate the quotient for relative results. That approach produced unreliable results because of the floating point precision [Gol91], therefore no further investigation on this approach was done.
Since m cannot be used to calculate reliable results b is investigated which is, after around 3,000 spheres, almost always in the range of $0.5 \leq b \leq 1.0$ so floating operation can be performed with a reasonable precision.
First the difference is calculated again. This approach leads to similar problems like the difference with m, the difference produces no stable results. Since the models have a different upper bound for the covered volume, like the Cylinder which has a total volume coverage with more than 95% compared to the Dog which hardly covers more than 90% of the volume. However, the covered volume differs by about 5% relative and absolute, this approach requires data which is accessible after the packing is done. Therefore, the difference of b can not be applied for a dynamic break criteria.
The last remaining approach is the division of b. By calculating the quotient,

which has no issues with the floating point precision and also provides stable result, after about 7,000 spheres are inserted with values in the range of of $0.95 \leq b \leq 1.0$. Over time, the b values grow to $0.999; 0.9999$ similarity of two following b values until the precision of the float is at its limit and the value is rounded to 1[Gol91]. However, this effect occurs when more than one million inserted spheres are exceeded and the result of the quotient increased slowly to 0.9999999 before it is rounded, therefore that issue is ignored. Any value beyond 0.999 can be used to stop the packing as dynamic break criteria to produce stable results. For the nine models which were used for this thesis the value 0.99995 turned out to be a good which is investigated in detail in 6.1. Therefore the quotient of b is used to configure the dynamic break criteria.

4.6 Classification

The initial state provides no sphere packing which could be used to calculate the similarity between two packings, since almost all packings are distributed similar to 3.4.8. To classify models depending on their shape, sphere packings must depend on the characteristics of the shape of the model. However, a tolerance factor of 10 for inserted spheres 4.2.1 leads to lots of spheres which produce a dominating noise. To apply a classification method on the packing, the noise must either be removed or comparable.

4.7 Summary

In this chapter, all issues which were found during the thesis are documented. If possible the issues are mitigated or possible enhancements are suggested. Additional general improvements which enable ProtoSphere to adjust the packing to several use-cases are documented.

Chapter 5

Results

ProtoSphere received several adjustments on the packing procedure and options for the packing configuration. In this chapter several results for the adjustments are shown. Additionally the effects for the major configurations are covered. Every packing was computed with a NVIDIA GeForce GTX 1070 Graphic Card and an Intel Core i7-6700K CPU and 16 GB RAM. For comparison a computation with the configuration target 200,000 spheres, volume per step: $0.02 \leq V \leq 0.25$, volume tolerance per sphere: $999 \leq T \leq 1000$ was used, since that configuration has the most similarities with the parallelized state.

5.1 Volume Covered

In this section the differences for the covered volume are shown. At first the parallelized state is compared to the most comparable configuration for the adjusted state, which is a tolerance for the sphere radius per step of 10, while each cell is used for the entire sphere packing, while the initial version just used 1 of 8 cells after 9 steps 4.2.2.

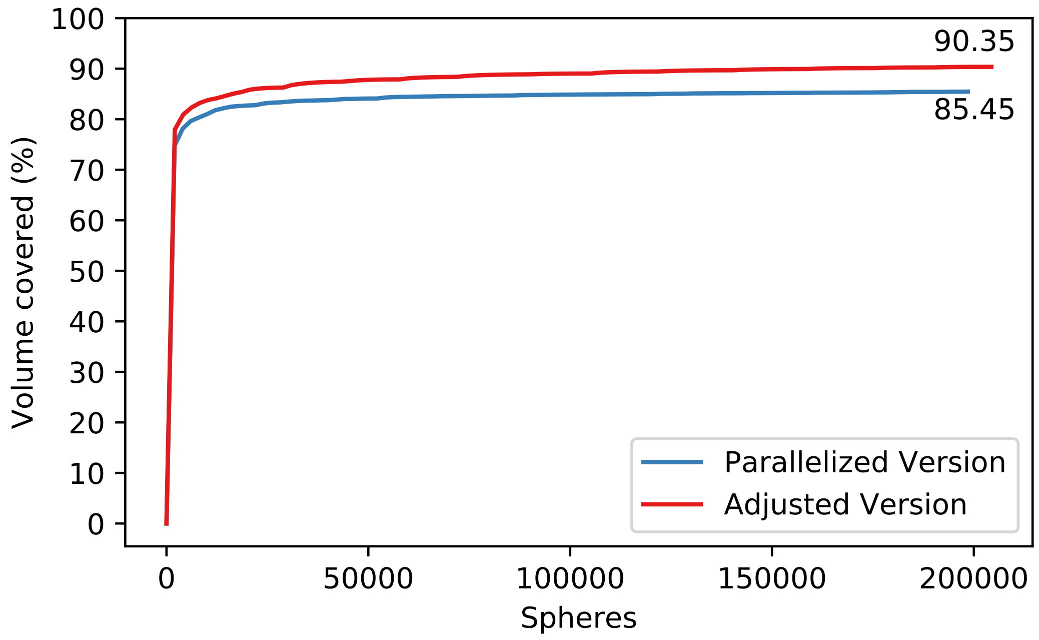

Figure 5.1: Comparison of filling curve before and after the adjustments. Blue: initial state, Red: with adjustments.

The Fig: 5.1 *a* represents the packing progress for the initial state which converges at about 85.45%. For the adjusted packing, the curve *b* converges at about 90.35% as it has a slightly bigger slope. This enhancement reduces the error from 14.55% to 9.64% which is about 34% less missed volume than the initial state of ProtoSphere.
The initial version of ProtoSphere, which did not use grids, uses an optimized packing on which does not use a tolerance for the inserted spheres per step. And there exists no absolute value as reference except for Fig: 5.2.

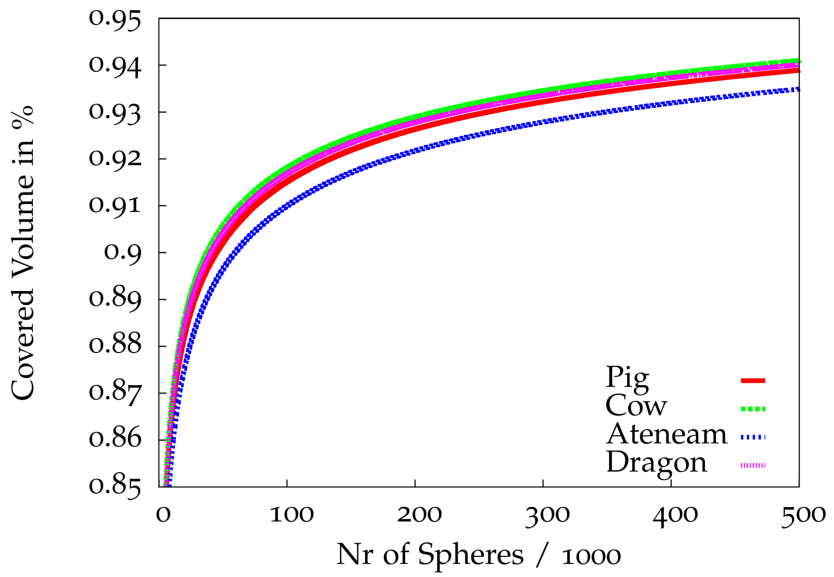

Figure 5.2: Covered volume for the version implemented in [Wel12]

No model covers for 100,000 spheres more than 92% of the volume, which is used as reference for comparison. To have a similar configuration, the adjusted version uses a maximum tolerance of 20% volume difference which is a radius tolerance of about 1.063 with a break criteria at either 100,000 spheres or 92% covered volume. The results are following: Ateneam: 90.80%, Cow 92% with less than 100,000 spheres, Pig 91.35% and Dragon 91.82%. Since the range of the radii is the same, the adjusted version is comparable to the non-parallelized version again.

5.2 Packing Quality

In the initial state, ProtoSphere distributed prototypes over the same cells and ignored a majority. From the distributed valid prototypes, almost all were inserted since the tolerance from the biggest to the smallest accepted a volume difference factor of 1,000. The following graphics compare the initial packing procedure to the adjusted procedure which distributes the prototypes

over all cells and accepts a maximum volume tolerance factor of 4 which is adjusted depending on the covered volume per step.
The major purpose of these representation is to verify if the packing is correct and to identify areas for which the covered volume could still be enhanced.

5.2.1 Sphere concentration

In this section the differences of the distribution of the sphere centers after the mitigation 4.2.2 is covered.

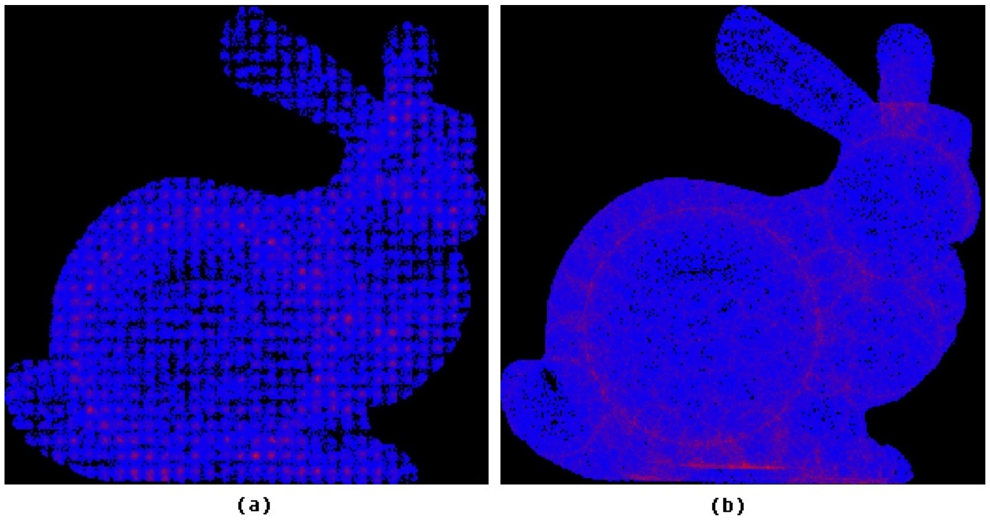

Figure 5.3: Comparison of the sphere concentration for 200,000 spheres before and after the adjustments. a) initial state b) with adjustments

In Fig: 5.3 for a) the parallized state is shown, for b) the adjusted state. The regular patterns which were caused by using the same cells over and over again disappeared. The silhouette of the model also became slightly bigger on the right side, since cells on the surface which were ignored could not be covered with spheres at all.
For the adjustments some new noticeable red areas appeared, red areas indicate flat surfaces. Since the model of the Bunny has flat areas on the bottom

and the top of the head, spheres from the XYZ map aligns in the Z direction. Big spheres also have surfaces with low bend factors which lead to similar effects. Depending on the size of the sphere, the surface is more or less visible in the concentration map. The enhancements are easily visible on any model.

5.2.2 Voxel covered

In this section the covered voxel map of the parallelized state and the adjusted state are compared.

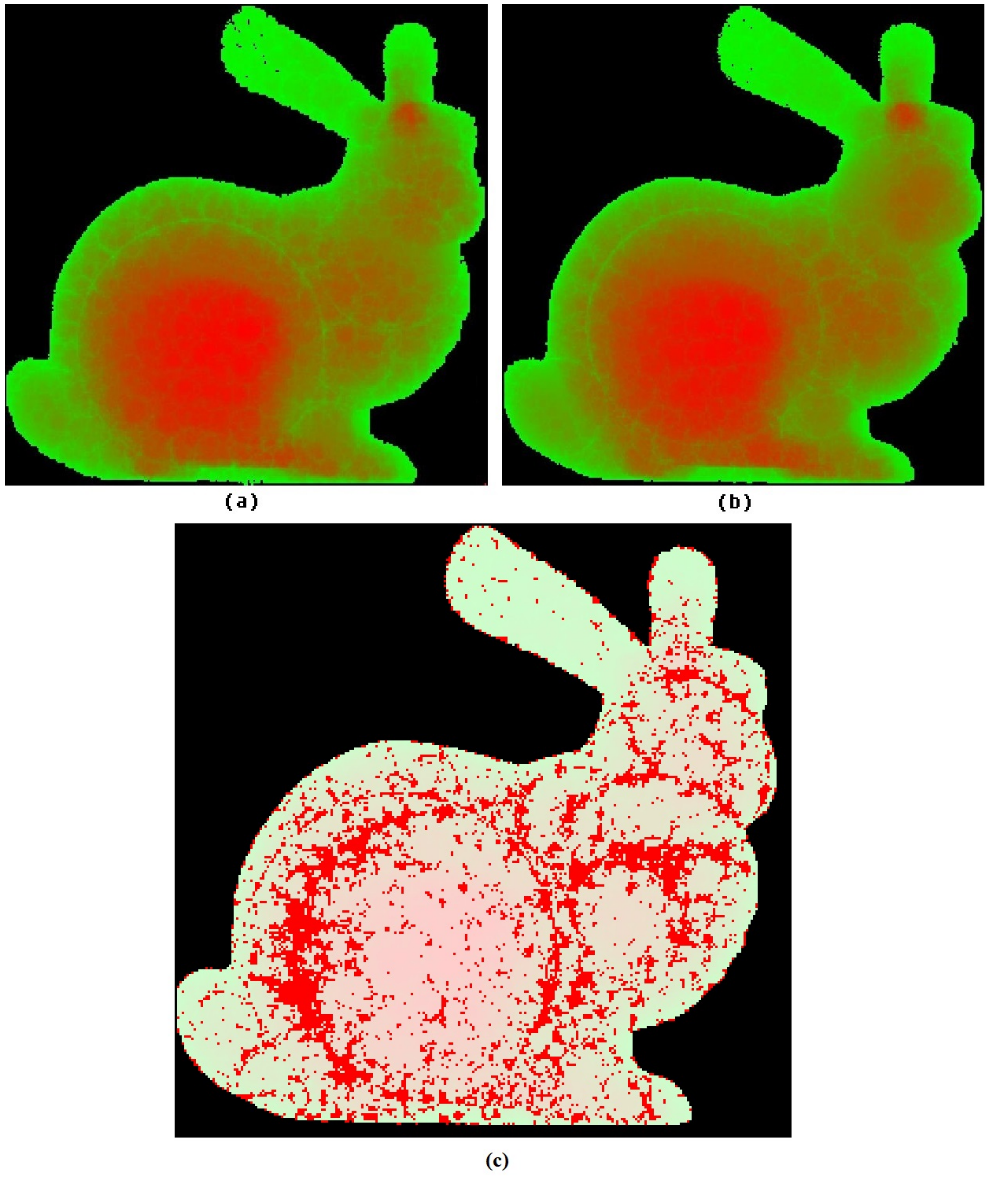

Figure 5.4: Comparison of the voxel coverage for 200,000 spheres before and after the adjustments. a) initial state b) with adjustments c) more than 6% difference between a and b is marked red.

The representation Fig: 5.4 does not differ much, since it mostly depends on big spheres. Still there are some improvements visible like less noise on the surface and inside of the ears. On the adjusted version the chest, throat and head appear slightly more red, since the covered volume increased. To compare both maps a difference of 6% was computed. If the same difference is applied on the $200kF$ and $critL$ which are explained in 6.1, which show the difference between a fast and optimized packing, less differences are visible Fig: 5.5.

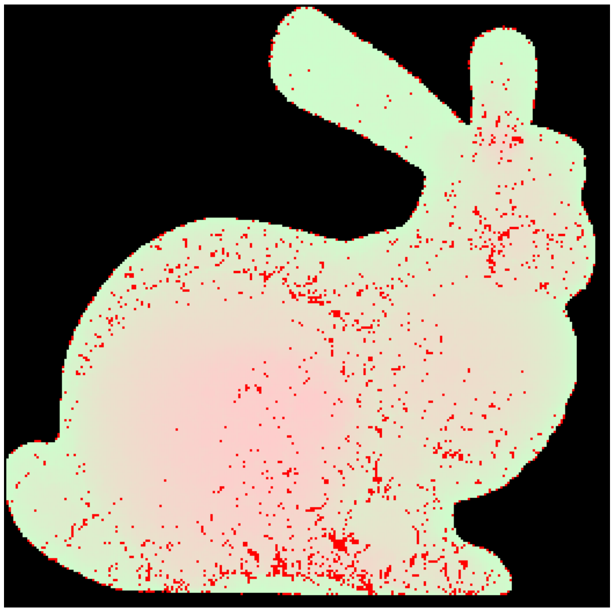

Figure 5.5: Difference between a non-optimized fast packing and a more optimized packing. Red pixels have more than 6% difference.

5.2.3 Density

In this section the density map of the parallelized state and the adjusted state are compared.

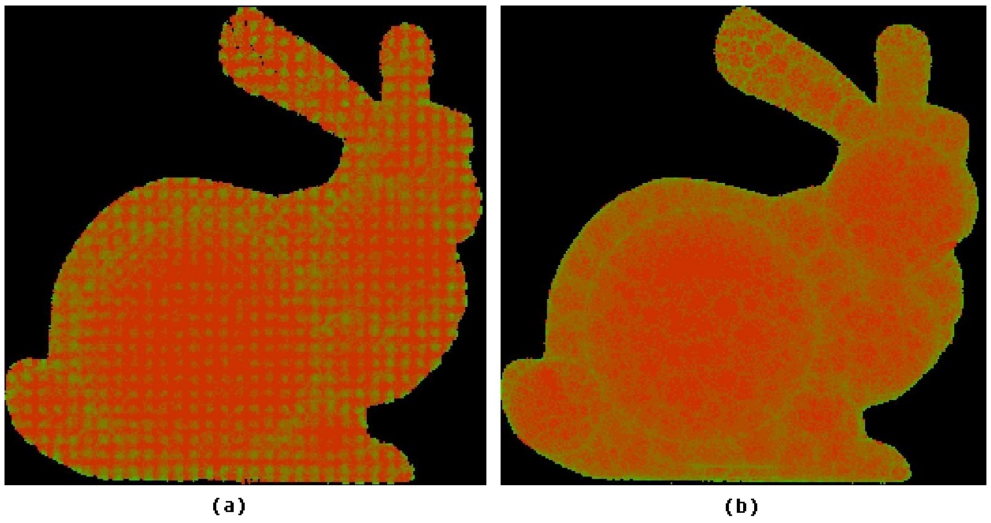

Figure 5.6: Comparison of the density for 200,000 spheres before and after the adjustments. a) initial state b) with adjustments

In Fig: 5.6 from the left side to the right side the regular patterns of the initial state disappeared, which results from the enhancements on the sphere concentration. For the right side the optimal result would be an image which would have even colors for major areas. Green areas indicate a low density, which is in general the surface of any model.

5.3 Distribution of radii

In this section the distribution of the radii of the spheres are compared, first the results for the parallelized state compared to the adjusted state. For a better comparison, the adjusted state is distributed for the same scaling as the parallelized state to visualize an actual comparison.

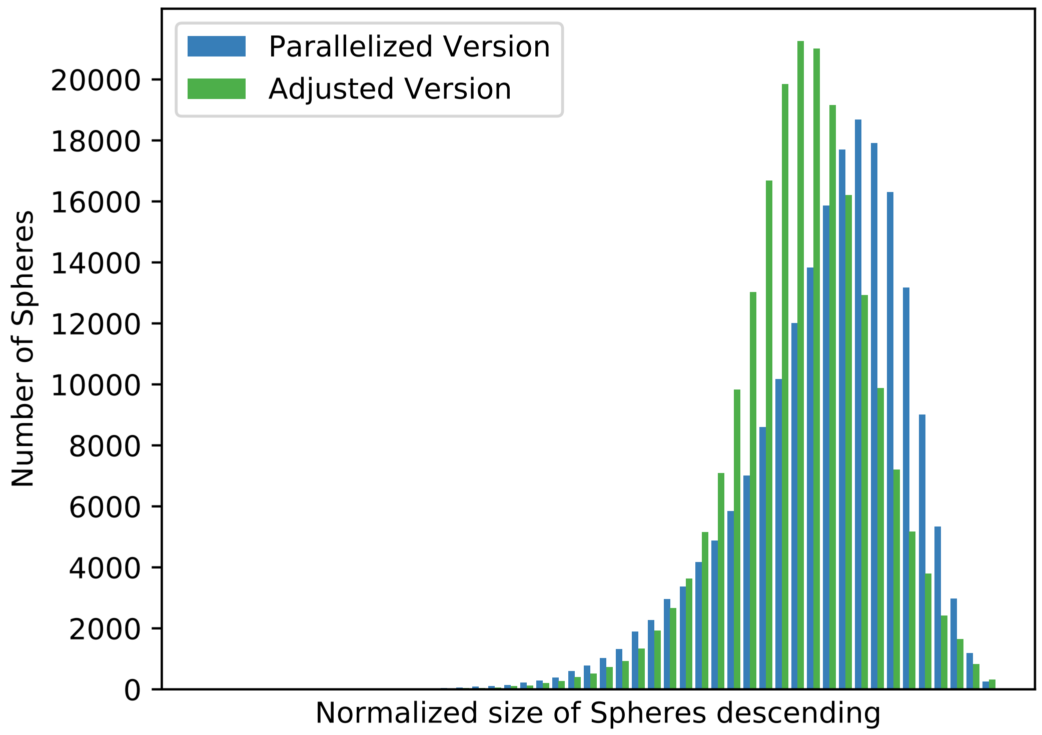

Figure 5.7: Comparing the distribution of the sphere radii for 200,000 spheres of the parallelized version and the adjusted version.

In Fig: 5.7 the distribution of the sphere radii for the parallized and adjusted version are compared. The distribution for the adjusted version is shifted slightly to the left while the amount of bigger spheres also increased from about 19,000 to 21,000, which means that more bigger spheres are inserted. For the biggest version, the adjusted version approximated a bigger Voronoi point than the parallelized version, while the smallest sphere is pretty much equal in size. Since the sphere sizes are bigger on average, the volume increased from 85.46% to 90.35% for the same amount of spheres.

5.4 Enhanced performance

To measure the enhanced performance, the most comparable way is to use 200,000 spheres as break criteria for both packing. Additionally, the new configuration for the adjusted packing was: $1 \leq spherSizeTolerance \leq 8$; $0.02 \leq tolerancePerStep \leq 0.25$. The results are shown in 5.1.

	Armdl.	Atnm.	Bunny	Cow	Cube	Cyl.	Drgn.	Pig	Dog	$\bar{x}$
i_vol	83.14	84.73	85.45	83.28	89.47	91.18	80.94	84.03	79.23	84.61
a_vol	88.22	90.36	89.33	90.11	91.07	93.43	83.31	89.39	86.065	89.36
%missing Volume Reduced	30.13	28.03	26.67	40.85	15.19	25.51	12.43	33.56	32.91	32.68
i_etr	0.226	0.187	0.216	0.173	0.208	0.154	0.207	0.162	0.214	0.194
a_etr	0.28	0.26	0.27	0.25	0.23	0.19	0.3	0.25	0.3	0.26
%Entropy Increased	25.83	40.11	22.69	45.09	11.06	22.73	45.41	52.47	38.79	33.90

Table 5.1: Comparison of the initial state and adjusted state. Prefix: i_ for initial state, a_ for adjusted state

With the adjustments and the fastest packing configuration, the density increased by more than 4% in total, which decreased the missing covered volume by more than 30%. By reducing the error the results also became more stable, for the initial state the standard deviation for the covered volume is 4.92% which was reduced to 3.95%, which means that the 95% confidence interval was reduced from $75.61 \leq x \leq 94.89$ to $80.52 \leq x \leq 96.01$, so the adjusted version produces more stable values for the covered volume.
In 5.3 the distribution of the spheres sizes shows, that the adjusted version still can cover more volume since it is still possible to insert smaller spheres which can still cover volume. In the filling curve the slope for the progress of the packing also indicates that more volume can be packed by spheres. This also goes for the initial state, but since that version cannot cover the volume over all grid cells, it obviously has a lower upper bound.
In the 2-d representations the coverage obviously became better since all cells are used for the volume coverage and as a result the patterns disappeared.

Finally the distribution graph shows that the adjustment to drop small spheres, which cover less than a fourth of the volume than the biggest sphere for the current iteration, leads to more big spheres which cover more volume.

5.5 Analyzing the Hypothesis

In this section the hypothesis 1.4 is analyzed. A common method to prove hypothesis by finding falsifiable criteria and applying those to the results.

5.5.1 Falsifiable criteria

For the following graphs the bunny was used. The adjusted version is tested, since the parallelized version generates packings with an uneven coverage.

5.5.2 Upper bound

To test if the upper bound for the covered volume comes close to 100%, the filling curve 5.8 is computed for the adjusted version.

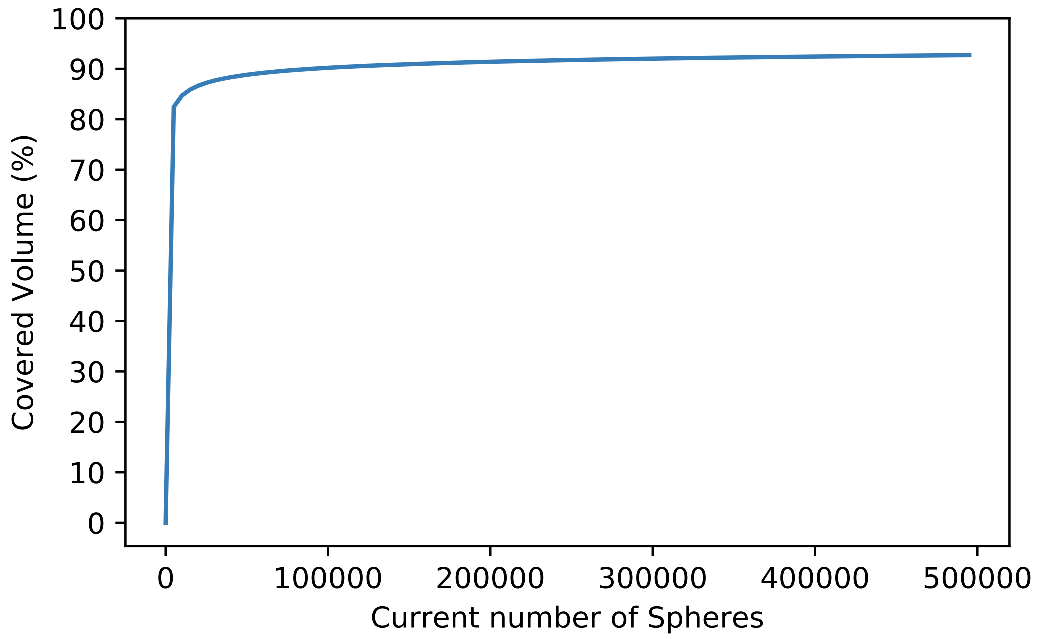

Figure 5.8: Filling curve of the bunny for the adjusted version.

In the curve 5.8 the covered volume slope is close to 0, which indicates that a huge amount of spheres is required to cover any more volume. For the first 60,000 spheres, the covered volume increases to about 90%, however the covered volume increases for the next 300,000 spheres by just about 3% with a decreasing slope. Therefore, the assumption is that either a ridiculous amount of spheres is needed to cover almost 99.9 percent of the volume, or there might be an upper bound at about 95 percent, which is suggested in [SDSRH12].

5.5.3 Disproval of the hypotheses

In this section both hypotheses 1 and 2 are challenged and proven to be wrong. As evidence the measurements are performed on the adjusted version.

Hypothesis 1 : The more spheres become inserted, the remaining voids become smaller, more similar in shape and size, therefore new spheres add no more information.

Hypothesis 2 : With more spheres the upper bound for the volume approximates 100%.

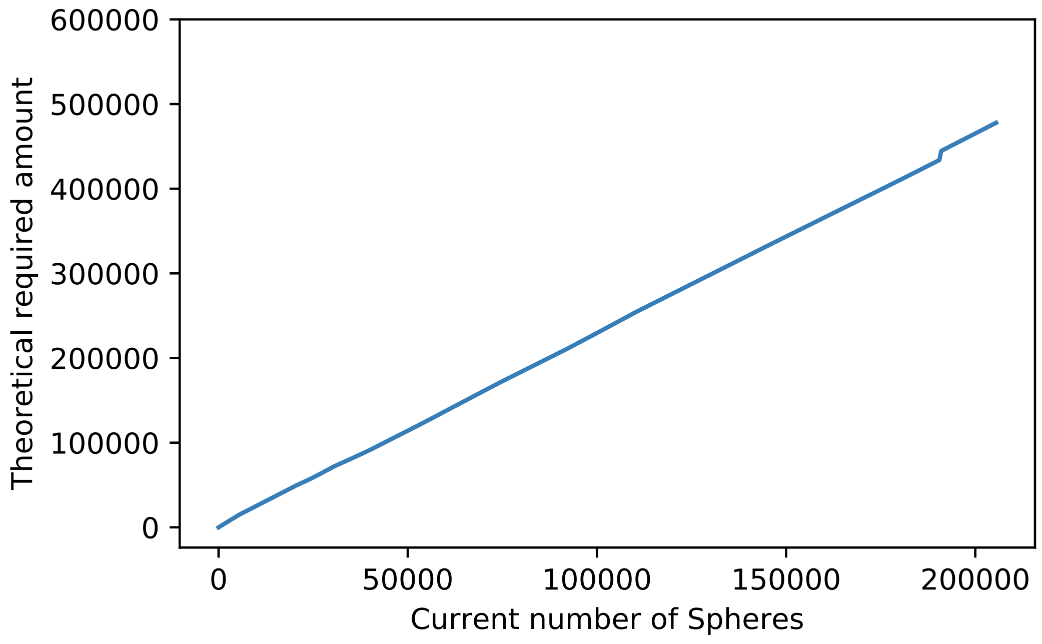

Figure 5.9: Amount of theoretical spheres required for a certain amount of spheres, with the assumption that the slope does not decrease anymore.

For these hypotheses 3.4.7 and Fig: 5.9 are used to prove that both are wrong. It is shown in 3.4.7 that the radii of spheres does not converge, during later stages the number if jumping in some ranges which cannot converge at all. Fig: 5.9 shows, that the number of theoretical required spheres increases at a faster rate than the actual covered volume. Because of that, the filling rate cannot reach 100% for the observed space. This is confirmed by the curve 5.8, which converges at about 90.4%. Therefore, the **hypothesis 1 and 2** are discarded.

5.5.4 Confirmation of the hypothesis

In this section, **hypothesis 3** is proven right.
Hypothesis 3 : A dynamic break criteria to stop the packing when the spheres become too small exists.
Two criteria where investigated in 4.5.3, first a break criteria when the percentage volume coverage exceeded a certain value, 4.5.3.1. This criteria has an issue, since the percentage volume coverage depends on the model and is in the percentage range of usually $89 \leq p \leq 94$, if the value 94% is configured, but the volume cannot exceed 93% the packing process never ends. In 4.5.4.2, a criteria which depends on the slope of the function was introduced. This criteria produces stable results, which covers a decent amount of the volume, depending on the configuration. Since all values are normalized by the quotient, the packing always stops when the configuration was meaningful. Therefore, the **hypothesis 3** is confirmed.

5.6 Summary

In this chapter, all major results of the enhancements were covered. The patterns of the initial state disappeared, which led to a significant reduction for the remaining non-covered volume. Additionally the initial hypotheses were rejected. The hypotheses were an assumption that in later stages of the packing all sphere radii are the same and over time 100% of the volume is approximated.

Chapter 6

Discussion

In this chapter, all adjustments are compared to each other to investigate the effect on the packing quality 6.1 to rank any configuration for several criteria 6.3. Furthermore, the default values for a configuration which satisfies a compromise on any conflicting packing criteria are discussed and tested 6.2. Aside from the packing quality, the similarity of the models for different packing strategies is investigated 6.4.

6.1 Packing Configuration

All adjustments have several advantages and disadvantages. Depending of the goal, different configurations help to receive the optimal packing. Different adjustments are compared in this chapter. Unmentioned configurations are set to unreachable values.
The splitting configuration is static for every packing with $n = 2$, $s = 6$, $m3 = 3$, $eig = 2$, $sq = 1$, $i = 30$.
For the packing the setups are the following:
critH quotient for b: 0.99995, volume per step: $0.02 \leq V \leq 0.25$, volume tolerance per sphere: $1 \leq T \leq 8$
critL quotient for b: 0.9995, volume per step: $0.02 \leq V \leq 0.25$, volume

tolerance per sphere: $1 \leq T \leq 8$

200kS target 200,000 spheres, volume per step: $0.02 \leq V \leq 0.25$, volume tolerance per sphere: $1 \leq T \leq 8$

200kF target 200,000 spheres, volume per step: $0.02 \leq V \leq 0.25$, volume tolerance per sphere: $999 \leq T \leq 1000$

tlrzL quotient for b: 0.99995, volume per step: $0.01 \leq V \leq 0.2$, volume tolerance per sphere: $1 \leq T \leq 8$

tlrzH quotient for b: 0.99995, volume per step: $0.5 \leq V \leq 1.0$, volume tolerance per sphere: $1 \leq T \leq 8$

tlrzS quotient for b: 0.99995, volume per step: $0.01 \leq V \leq 0.25$, volume tolerance per sphere: $2 \leq T \leq 8$

tVol target vol about 1% more than the *critH* results, volume per step: $0.02 \leq V \leq 0.25$, volume tolerance per sphere: $1 \leq T \leq 8$

dflt target vol 92.5%: $0.02 \leq V \leq 0.25$,quotient for b: 0.99995, target 200,000 spheres, volume per step: $0.2 \leq V \leq 0.5$, volume tolerance per sphere: $1.26 \leq T \leq 8$

opt target vol 92.5%: $0.02 \leq V \leq 0.25$, target 200,000 spheres, volume tolerance per sphere: $1 \leq T \leq 1.05$

200kS stands for 200,000 slow packing and **200kF** for 200,000 fast packing. These packings compare the computation time for the *minsd* and *maxsd* adjustments. **critL** and **critH** stands for the dynamic break criteria which stops the packing when the slope has almost no more slope, the effect of the difference of 0.9995 and 0.99995 is compared. **tVol** stands for target volume, the upper bounds depend on the *critH* results with a slightly increased volume for every model. **Opt** stands for optimal packing, where a minimal amount of spheres are inserted to generate a packing where any sphere is as close as possible to its Voronoi point. **tlrzL** stands for low tolerance, this packing forces the covered volume per step into a certain amount by adjusting the sphere tolerance. **tlrzH** stands for high tolerance, which adjusts the sphere tolerance if exceptionally, low or high values are exceeded. Finally, **tlrzS** is used to compare the results for a minimum volume tolerance per step of 2.

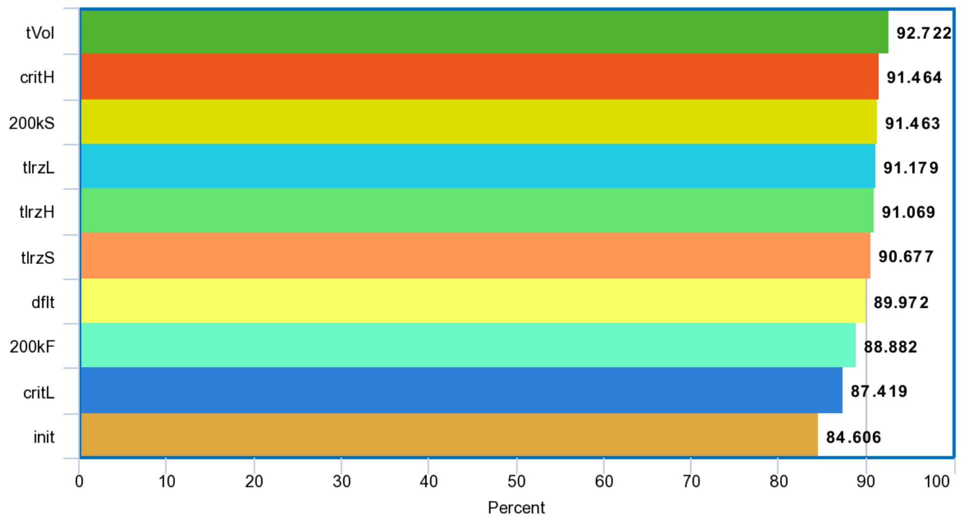

Figure 6.1: Result: Comparison of the covered volume for all configurations, the higher the better.

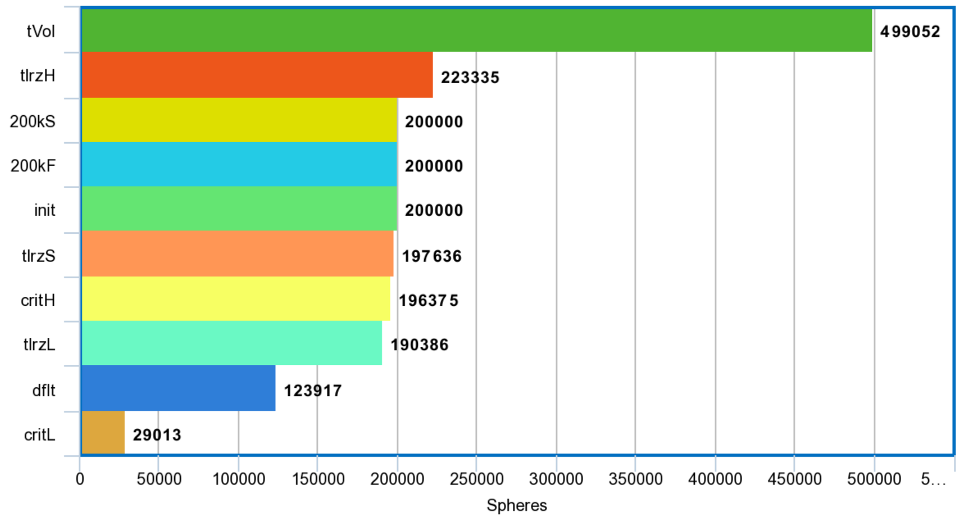

Figure 6.2: Result: Comparison of the required spheres for all configurations

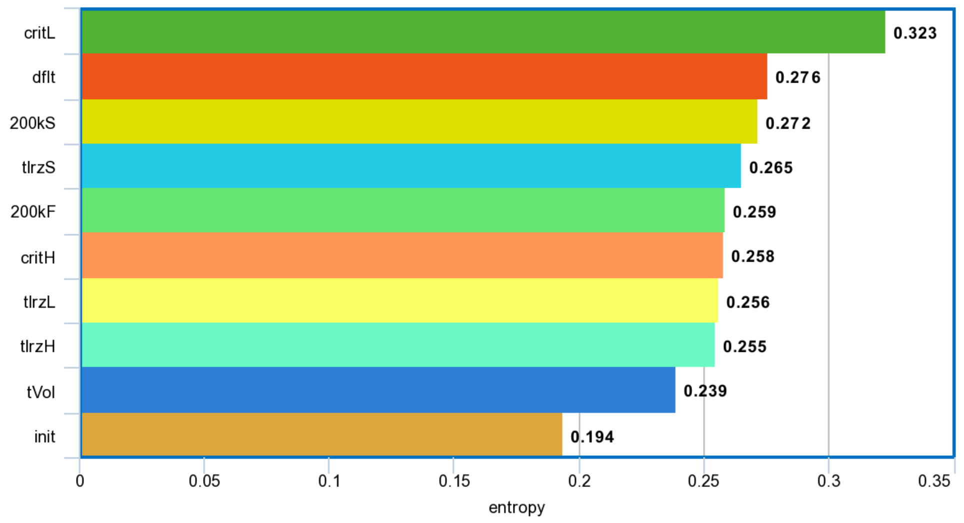

Figure 6.3: Result: Comparison of the entropy for all configurations, the higher the better.

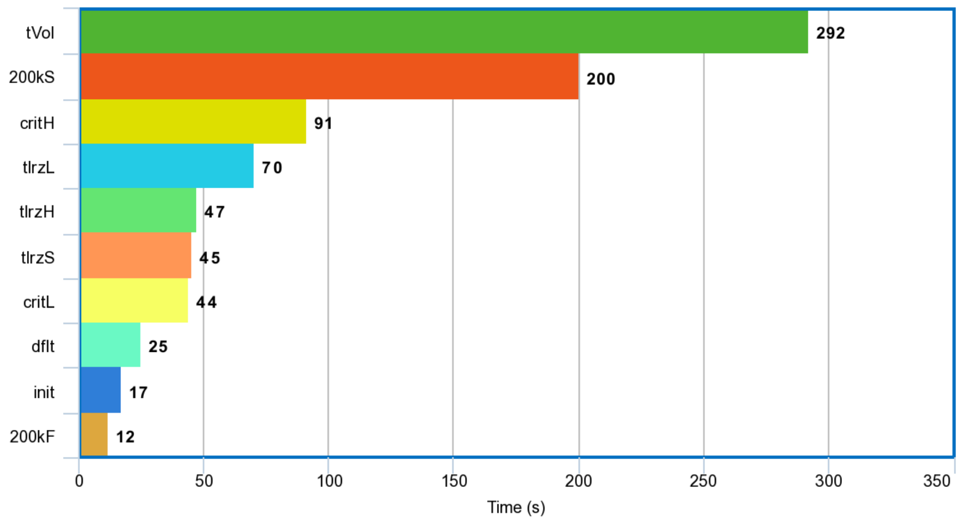

Figure 6.4: Result: Comparison calculation time for all configurations, the lower the better.

6.2 Default Packing Configuration

To find a good default configuration, all major configuration possibilities are compared to each other in Fig: 6.1, Fig: 6.2, Fig: 6.3 and Fig: 6.4 . For an easy comparison, the default results are already shown.

All configurations produced different results, so obviously the configuration has a huge effect on the result. First cH and cL compare the break criteria thresholds. A lower threshold packs the models way faster with a lower amount spheres but misses about 4% volume; since the packing is still quite good, the entropy for the cL configuration is better because the spheres on average cover more volume, therefore the cH configuration performs better. Next there are the $2S$ and $2F$ configurations, which compares a low tolerance to a big tolerance. Again the less strict configuration is faster but misses some volume for the same amount of spheres. Since the less tolerant setup misses some volume, the entropy for the packing is slightly lower; depending on the goal, both configurations are valid which would set a good amount of spheres to 200,000. The next three setups compare different tolerance settings for the covered volume per step. Interestingly the final packing has a difference of less than 0.5% for the final packing, with similar amounts of spheres and computation time. Since th performs worse for either the covered volume or the computation time, a mix of the tL and tS configuration might appear as a good solution to benefit from the advantages of both setups. Finally, the volume covered configuration is investigated, which produces the best results regarding the covered volume, but the worst in terms of time and average amount of spheres required to hit that volume. Still, this result shows that objects which are densely packed with a low number of spheres could also need this break criteria.

As a result, the final configuration should have an upper bound of 200,000 spheres, 93.5% covered volume and regression quotient of 0.9999. If either is hit the packing is stopped. During run time the adaptive filling should cover about 0.2% to 0.5% of the remaining volume, while starting at 1.26 (cube root of 2) initial tolerance which scales up to 8 (power 3 of 2). This leads to the configuration

df target vol: 93.5%, quotient for b: 0.9999, target 200,000 spheres, volume per step: $0.2 \leq V \leq 0.5$, volume tolerance per sphere: $1.26 \leq T \leq 8$
By comparing all results, following tabular 6.1 shows the results.

6.3 Ranking Packing Configuration

aspect	1	2	3	4	5	6	7	8	9	10
volume	critH	200kS	tlrzL	tlrzH	tlrzS	dflt	200kF	critL	init	(tVol)
entropy	critL	dflt	200kS	tlrzS	200kF	critH	tlrzL	tlrzH	tVol	init
spheres	critL	dflt	tlrzL	critH	tlrzS	tlrzH	tVol	(200kS)	(200kF)	(init)
time	200kF	init	dflt	critL	tlrzS	tlrzH	tlrzL	critH	200kS	tVol

Table 6.1: Ranking all configurations. Rank 1 is the best and 10 the worst. Break criteria rankings are added as last entries and are not considered for the final rank.

By ignoring the setups which hit its break criteria, the configurations are ranked in the following order.

	dflt	critL	200kF	200kS	tlrzS	critH	tlrzL	tlrzH	init	tVol
$\bar{x}$	3.25	3.5	4.33	4.66	4.75	4.75	6	6	7	8.66

Table 6.2: Final Ranking, average placement of the packing configuration for volume, entropy, number of spheres, calculation time.

By comparing the ranks in 6.2 without weighting, the default configuration produces on average the best results. The only aspect on which the default configurations seem to perform badly, is the final packing density. By inspecting this rank in detail there is a range for the final packing from $85.868 \leq D \leq 91.464$, the default configuration still has a density of 89.97% which is about 1.5% less than the best configuration. Compared to the other fast packing configurations *init* with 85.87 cL with 87.14% and $2F$ with 88.89% which have also no big outlier, the default setting covers more than 1% volume for good computation times. Considering that, the default configurations performs well on any aspect.

Comparing again the initial state to the default and best volume configurations this leads to the following tabular 6.3.

	Armdl.	Atnm.	Bunny	Cow	Cube	Cyl.	Drgn.	Pig	Dog	$\bar{x}$
i_vol	83.14	84.73	85.45	83.28	89.47	91.18	80.94	84.03	79.23	84.606
i_etr	0.226	0.187	0.216	0.173	0.208	0.154	0.207	0.162	0.214	0.194
d_volume	89.02	89.70	89.92	90.43	93.50	93.24	87.82	89.92	86.20	89.972
d_entropy	0.304	0.28	0.289	0.274	0.232	0.209	0.311	0.268	0.313	0.276
%missing Volume Reduced	34.88	32.55	30.72	42.76	38.27	23.36	36.1	36.88	33.56	34.34
%Entropy Increased	34.69	49.73	33.8	58.38	11.54	35.71	50.24	65.43	46.26	42.866
c_volume	90.8	91.45	91.74	91.97	93.46	94.55	89.87	91.65	87.69	91.46
c_entropyc	0.281	0.258	0.266	0.256	0.241	0.201	0.286	0.247	0.288	0.258
%missing Volume Reduced	45.43	44.01	43.23	51.97	37.89	38.21	46.85	47.71	40.73	44.01
%Entropy Increased	24.5	37.97	23.15	47.98	15.87	30.52	38.16	52.47	34.58	33.910

Table 6.3: Packing comparison major configurations, Prefix i_ : initial configuration, d_ : default configuration, c_ : criteria low quotient divergence

The standard deviation for the default configuration is 2.32 which leads to a 95% confidence interval of $85.43 \leq x \leq 94.52$, the standard deviation for the criteria configuration is 1.97 which leads to a 95% confidence interval of $87.60 \leq x \leq 95.33$. Overall the missing volume was reduced by more than 33% for the default setup and almost 45% for the highest packing density setup which has a reasonable amount of spheres and computation time. Considering the results that the upper bound for the packing density is at about 95.01% [SDSRH12], the best packing configuration reduced the possible remaining error by more than 50%. The overall distribution of the volume coverage for the major configuration is shown in the following tabular 6.4.

	std dev	$\bar{x}$	95% lb	95% ub
initial	3.77	84.60	77.22	91.99
fast 200 k	2.39	88.89	84.20	93.57
default	2.32	89.97	85.43	94.52
highest density	1.97	91.46	87.60	95.33

Table 6.4: Comparing standard deviation, average covered volume and 95% confidence interval for the major configurations

Every new packing, regardless of how simple the configuration is, provides better and more stable results. The $default$ configuration is the best tradeoff for all aspects and performs slightly worse than the highest density configuration, which lowest bound on the 95% confidence interval even surpasses the initial setup, ProtoSphere performs way better.

6.4 Packing Similarity and Classification

As mentioned in 3.4.8, sphere packing can potentially be used to classify models. Initially the packing had no similarity 3.2 which could be used to classify models. In the following, the similarity of the models will be applied on three major setups: 200k fast packing, highest density criteria packing, default packing.

	Ateneam	Armadillo	Dog	Cow	Cylinder	Cube	Dragon	Bunny	Pig
Ateneam	1.00	0.97	1.00	1.00	0.95	0.90	1.00	0.95	1.00
Armadillo	0.97	1.00	0.98	0.97	1.00	0.98	0.97	1.00	0.99
Dog	1.00	0.98	1.00	1.00	0.97	0.93	1.00	0.97	1.00
Cow	1.00	0.97	1.00	1.00	0.95	0.90	1.00	0.95	0.99
Cylinder	0.95	1.00	0.97	0.95	1.00	0.99	0.95	1.00	0.97
Cube	0.90	0.98	0.93	0.90	0.99	1.00	0.90	0.99	0.94
Dragon	1.00	0.97	1.00	1.00	0.95	0.90	1.00	0.95	0.99
Bunny	0.95	1.00	0.97	0.95	1.00	0.99	0.95	1.00	0.97
Pig	1.00	0.99	1.00	0.99	0.97	0.94	0.99	0.97	1.00

Table 6.5: Similarity for the default configuration $dflt$. 1.00 indicates a 100% equality which decreases with less similar packings.

For the default packing 6.5, the issues from the initial packing became even worse. The lowest similarity increased to 90% from 85%, which makes classifications even more impossible. On the other hand, this result can also be interpreted as a verification that the default configuration works equally at least for the models which were used for this thesis.

	Ateneam	Armadillo	Dog	Cow	Cylinder	Cube	Dragon	Bunny	Pig
Ateneam	1.00	1.00	0.99	0.98	0.97	0.95	0.97	1.00	0.99
Armadillo	1.00	1.00	0.99	0.98	0.98	0.95	0.97	1.00	0.99
Dog	0.99	0.99	1.00	1.00	0.95	0.91	0.99	0.98	1.00
Cow	0.98	0.98	1.00	1.00	0.92	0.87	1.00	0.97	1.00
Cylinder	0.97	0.98	0.95	0.92	1.00	0.99	0.90	0.99	0.94
Cube	0.95	0.95	0.91	0.87	0.99	1.00	0.85	0.97	0.90
Dragon	0.97	0.97	0.99	1.00	0.90	0.85	1.00	0.95	0.99
Bunny	1.00	1.00	0.98	0.97	0.99	0.97	0.95	1.00	0.98
Pig	0.99	0.99	1.00	1.00	0.94	0.90	0.99	0.98	1.00

Table 6.6: Similarity for the fastest packing configuration $200kF$. 1.00 indicates a 100% equality which decreases with less similar packings.

Like the initial state packing and the default packing, this result 6.6 has at worst a 85% similarity which shouldn't be used to classify models.

	Ateneam	Armadillo	Dog	Cow	Cylinder	Cube	Dragon	Bunny	Pig
Ateneam	1.00	0.97	0.89	0.99	0.86	0.80	0.86	0.98	0.95
Armadillo	0.97	1.00	0.97	0.99	0.76	0.70	0.95	0.91	1.00
Dog	0.89	0.97	1.00	0.94	0.62	0.58	1.00	0.80	0.98
Cow	0.99	0.99	0.94	1.00	0.79	0.73	0.91	0.94	0.99
Cylinder	0.86	0.76	0.62	0.79	1.00	0.98	0.59	0.93	0.71
Cube	0.80	0.70	0.58	0.73	0.98	1.00	0.55	0.87	0.65
Dragon	0.86	0.95	1.00	0.91	0.59	0.55	1.00	0.76	0.96
Bunny	0.98	0.91	0.80	0.94	0.93	0.87	0.76	1.00	0.88
Pig	0.95	1.00	0.98	0.99	0.71	0.65	0.96	0.88	1.00

Table 6.7: Similarity of the more sensitive criteria packing $critH$. 1.00 indicates a 100% equality which decreases with less similar packings.

Finally, the highest density packing which was computed with the more sensitive dynamic break criteria provides promising results 6.7. The similarity ranges from 55% to 100%. In 3.3 the models were classified as:
Majority torso: Ateneam, Bunny, Cow, Pig
Majority limbs: Armadillo, Dragon, Dog

Big limbs: Armadillo, Pig, (Dragon)
No small connections: Ateneam, Bunny, Cube, Cylinder
For majority torso, the classification would fit for all object, which were chosen for this aspect, the Armadillo with a similarity of more than 95%. The same goes for the majority limbs and small limbs classification, however more models would fit for those aspects too. Additionally the no small connections classification holds again if the low similarity of the Cube and Cylinder for all other models is considered.
On the other hand, if objects with no similar common classifications are compared like the Bunny and Dragon, the similarity drops to less than 80%. The Cube and Cylinder have the worst results for similarity, both models are pretty similar to each other. For other objects the similarity rises, e.g. the Cube and Ateneam which consists of a cube like object and a head and the Cube and Bunny. The sample size is small and a bigger sample size would be too much to cover for this thesis. Still, these results show that further investigations might show that the cosine similarity can be used as classification criteria.

However, these results indicate, that a classification needs a small or similar noise of small spheres to produce reliable results. The definition for noise in this case are small spheres which dominate in numbers while contributing almost no covered volume.

6.5 Packing Divergence

In 3.4.8 the Kullback–Leibler divergence had some promising results since the divergence had a big range from $0.05 \leq D \leq 1.33$. However, the divergence was computed on a configuration which ignored the majority of cells. Since the new packing considers all cells which cover more volume, the results are evaluated again.

	Ateneam	Armadillo	Cylinder	Cow	Cube	Dragon	Bunny	Dog	Pig
Ateneam	0.00	0.08	0.19	0.06	0.12	0.45	0.30	0.08	0.10
Armadillo	0.09	0.00	0.14	0.10	0.22	0.54	0.25	0.14	0.14
Cylinder	0.20	0.14	0.00	0.24	0.34	0.71	0.15	0.27	0.28
Cow	0.06	0.09	0.23	0.00	0.13	0.40	0.34	0.05	0.05
Cube	0.10	0.17	0.27	0.11	0.00	0.31	0.38	0.09	0.13
Dragon	0.36	0.39	0.54	0.31	0.29	0.00	0.60	0.29	0.27
Bunny	0.37	0.28	0.16	0.40	0.54	0.91	0.00	0.44	0.42
Dog	0.08	0.12	0.25	0.05	0.10	0.36	0.37	0.00	0.06
Pig	0.10	0.13	0.28	0.05	0.14	0.36	0.37	0.06	0.00

Table 6.8: Divergence of the more sensitive criteria packing $critH$. 0.0 indicate no divergence, bigger values indicate bigger divergences

In6.8 the range decreased from $0.05 \leq D \leq 1.33$ to $0.05 \leq D \leq 0.91$ similar objects like the Dog, Cow Armadillo and Pig again have low divergence values. However the Ateneam and Cube have divergences in the same range, while models like the Bunny and Dragon which are unique in their shapes, have big divergence results to almost any model. Therefore, like the packing similarity 6.4 further investigations might show that the divergence can be used as a classification criteria.

6.6 Summary

In this chapter, all configurations were compared to each other. By comparing all configurations, a default configuration which is a good tradeoff for all major aspects for sphere packing was found. If the cosine similarity or Kullback–Leibler divergence are applied to the packing, the sensitive break criteria provides results which could actually be used to classify objects by its packing.

Chapter 7

Conclusion

The missing packed volume with ProtoSphere was reduced by more than 50% if the upper bound 95.01% is considered as the maximum packing density. This enhancement was achieved by adjusting the prototype distribution algorithm to consider any cell and adjusting the sphere size tolerance per step by a configurable limit.
There are also new options to configure ProtoSphere to either pack models with a small amount of very good spheres, which is slow or pack a model quickly with any sphere which is inside the tolerance bounds. In addition to the packing configuration, there are new possibilities to stop the packing algorithm of ProtoSphere for either a certain amount of covered volume or a slope on the filling curve which is close to 0.
Two new tools were implemented for the analyses, which either analyze a sphere packing or compare sphere packings. The analysis tool, which is called *PackingAnalyzer*, generates several human readable 2-d graphics to verify if the packing is still valid for aspects like sphere concentration, voxel coverage or the density of the covered volume. Aside from the 2-d graphics, several graphs were added, which visualize the packing process with an increasing number of spheres and the theoretical amount of required spheres. Finally histogram visualizes the sphere size distribution. The distribution is also used to compute the entropy of the packing or the similarity between two pack-

ings which can be calculated by the second tool, the *PackingComparer*. For comparable results, the amount of small spheres which contribute almost no volume should be as minimal as possible or be comparable.
The tools were used to compare the different packing settings to each other, which target different aspects like a short computation time, the most covered volume or just insert the biggest spheres. Since each aspect is in conflict with each other, there is no optimal packing configuration, but depending on the results a default packing configuration was derived which performs well on average.
The **hypothesis 1 and 2** were discarded, since the results show that the upper bound for the covered volume is close to 95.01% than to 100%, additionally the tolerance factor for the inserted prototypes prevents the prototypes converging to the same radii for later stages of the filling process. In contrast to **hypothesis 1 and 2** the **hypothesis 3** was confirmed, since at least one stable dynamic break criteria exists.

Chapter 8

Future Work

ProtoSphere was improved on several aspects and performs better in any setup than the parallelized version. However, several issues were discovered during the thesis which still can be mitigated and enhance the performance of ProtoSphere even more.
One big improvement would be to cluster parts of the models where spheres have no chance to intersect and apply the inserting tolerance to each cluster instead of any inserted sphere for the single steps. This would fill any independent part of the mesh in parallel and increase the packing speed by a considerable amount while each part on the mesh is packed evenly. By doing this, there would also be a chance to create packing for deformable models, if clusters are defined between joints. Another enhancement is smarter bounding boxes for insertion; prototypes are dropped all the time if its cell is covered by a small part of the model, with smart bounding boxes this wasted effort would be reduced. This enhancement might also solve the problem for invalid spheres which are inserted between small parts of the model like fingers which are inside a single box.
There are also some minor issues during the packing process like prototypes which cannot converge towards Voronoi points. If those prototypes can be detected during computation time, they could either be dropped or other adjustments could be made which prevent ProtoSphere from inserting non-

Voronoi prototypes. By doing this, less spheres are required since potentially good spheres are not by bad prototypes anymore.
Aside from the packing, several aspects like the entropy, similarity or divergence were investigated, which might qualify to classify models depending on their characteristics. During this thesis only nine models were used as reference, which is a small sample size. The metrics can either be verified, discarded or enhanced if those are applied on a larger sample size. Another interesting observation is that if the entropy and covered volume percentage are multiplied, the result decreases over time until it converges, which might indicate another possible break criteria.
ProtoSphere was not enhanced for models which are not watertight. Additionally, thin areas or parts of the model which cover just a small part of grid cells are hardly covered since just prototypes which spawn inside of the model are considered for the optimization process. There are several possible solutions which can mitigate these issues and would enable ProtoSphere to pack even more models efficiently.

Bibliography

[Ake14] Akeb, Hakim: *A Look-Forward Heuristic for Packing Spheres into a Three-Dimensional Bin.* 2014

[Alb17] Albakush, Intisar H.: *Plagiarism Detection Between Theory And Practical Calculations.* Version: 2017. `https://pdfs.semanticscholar.org/72df/da79a3e4a62b387a05bfa601f47c7531d2fe.pdf`, Abruf: 08.08.2019. – 61–62 S.

[BDA11] Belov, Dmitry ; D Armstrong, Ronald: Distributions of the Kullback-Leibler divergence with applications. In: *The British journal of mathematical and statistical psychology* 64 (2011), 05, S. 291–309. `http://dx.doi.org/10.1348/000711010X522227`. – DOI 10.1348/000711010X522227

[BHMv14] Bienert, A. ; Hess, C. ; Maas, H.-G. ; von Oheimb, G.: A voxel-based technique to estimate the volume of trees from terrestrial laser scanner data. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2014), Juni, S. 101–106. `http://dx.doi.org/10.5194/isprsarchives-XL-5-101-2014`. – DOI 10.5194/isprsarchives–XL–5–101–2014

[BHS08] Benoit Hudson, Todd P. Gary L. Miller M. Gary L. Miller ; Sheehy, Don: *Size Complexity of Volume Meshes vs. Sur-*

face Meshes. Version: 2008. http://donsheehy.net/research/hudson09size.pdf, Abruf: 21.07.2019. – 1–7 S.

[Gol91] GOLDBERG, David: *What Every Computer Scientist Should Know About Floating-Point Arithmetic.* Version: 1991. https://www.itu.dk/~sestoft/bachelor/IEEE754_article.pdf, Abruf: 21.07.2019. – 8–8 S.

[HS11] HOPKINS, Adam B. ; STILLINGER, Frank H.: *Densest local sphere-packing diversity. II. Application to three dimensions.* 2011

[IM] INDUSTRIAL, Society for ; MATHEMATICS, Applied: *Linear Least SquaresAnalysis.* https://archive.siam.org/books/sa14/sa14_samplechapter.pdf, Abruf: 08.08.2019. – 205–209 S.

[MB95] MICHAEL BORKOVEC, Ronald P. Walter de Paris P. Walter de Paris: *The Fractal Dimension of the Apollonian Sphere Packing.* Version: 1995. https://www.researchgate.net/publication/228326605_The_Fractal_Dimension_of_the_Apollonian_Sphere_Packing?enrichId=rgreq-8675d717c66ef9562ed7f10afe24f956-XXX&enrichSource=Y292ZXJQYWdlOzIyODMyNjYwNTtBUzoxMDY1MDk1MjI0NDAxOTZAMTQwMj
3D%3D&el=1_x_3&_esc=publicationCoverPdf, Abruf: 05.08.2019. – 1–5 S.

[Mis16] MISTRY, Roshni: *Circle Packing, Sphere Packing, and Kepler's Conjecture.* 2016

[PAS99] PIERRE ALLIEZ, Henri S. Nathalie Laurent L. Nathalie Laurent ; SCHMITT, Francis: *Mesh Approximation using a Volume-Based Metric.* Version: 1999. https://ieeexplore.ieee.org/document/803373, Abruf: 21.07.2019. – 1–10 S.

[RWT13] Rene Weller, Stefan G. Gabriel Zachmann Z. Gabriel Zachmann ; Teuber, Jörn: *Fast Sphere Packings with Adaptive Grids on the GPU*. Version: 2013. `http://www.gcc.tu-darmstadt.de/media/gcc/papers/Teuber-2013-GI.pdf`, Abruf: 18.12.2015. – 3–9 S.

[SCS13] Stolojescu-Crisan, Cristina ; Stefan, Holban: A Comparison of X-Ray Image Segmentation Techniques. In: *Advances in Electrical and Computer Engineering* 3 (2013), 08. `http://dx.doi.org/10.4316/AECE.2013.03014`. – DOI 10.4316/AECE.2013.03014

[SDSRH12] S. D. S. Reis, J. S. Andrade j. N. A. M. Araujo A. N. A. M. Araujo ; Herrman, Hans J.: *How dense can one pack spheres of arbitrary size distribution.* 2012

[Sha48] Shannon, C. E.: *A Mathematical Theory of Communication.* Version: 1948. `http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf`, Abruf: 21.07.2019. – 10–12 S.

[Teu13] Teuber, Jörn: *Faster Sphere Packing with Adaptive Grids.* 2013

[Wel12] Weller, Rene: *NEW GEOMETRIC DATA STRUCTURES FOR COLLISION DETECTION.* Version: 2012. `http://elib.suub.uni-bremen.de/edocs/00102857-1.pdf`, Abruf: 19.04.2019. – 83–103 S.

[WZ10a] Weller, Rene ; Zachmann, Gabriel: *Inner Sphere Trees for Proximity and Penetration Queries.* Version: 2010. `http://www.roboticsproceedings.org/rss05/p10.pdf`, Abruf: 18.12.2015. – 3–6 S.

[WZ10b] Weller, Rene ; Zachmann, Gabriel: *ProtoSphere: A GPU-Assisted Prototype Guided Sphere Packing Algorithm for Arbitrary Objects.* Version: 2010. `http://cgvr.cs.uni-bremen.`

`de/papers/siggraph_asia2010/ProtoSphereSiggraph.pdf`, Abruf: 24.08.2019. – 1–2 S.

List of Figures

List of Tables

Glossary

cfg.	mdl.	armd	atnm	bny	Cow	Cube	cyl	drgn	Pig	smrf	$\bar{x}$
cH	vol	90.8	91.45	91.74	91.97	93.46	94.55	89.87	91.65	84.05	91.06
cH	ent	0.281	0.258	0.266	0.256	0.241	0.201	0.286	0.247	0.307	0.260
cH	s(k)	219	202	198	187	160	123	256	197	274	202
cH	time	156	105	131	131	383	168	128	135	363	189
cL	vol	86.70	87.63	88.06	88.16	90.28	89.65	84.89	87.72	79.25	86.92
cL	ent	0.361	0.343	0.354	0.330	0.298	0.237	0.289	0.329	0.410	0.328
cL	s(k)	32	30	29	27	23	17	39	29	42	30
cL	time	57	32	45	27	109	24	47	24	108	53
2S	vol	90.65	91.43	91.75	92.07	93.73	95.02	89.35	91.68	83.42	91.01
2S	ent	0.285	0.258	0.265	0.255	0.235	0.293	0.289	0.246	0.324	0.272
2S	s(k)	200	200	200	200	200	200	200	200	200	-/-
2S	time	149	102	113	127	430	225	113	104	440	200
2F	vol	88.22	89.01	89.33	90.11	91.07	93.43	83.31	89.39	80.53	88.26
2F	ent	0.284	0.262	0.265	0.251	0.231	0.189	0.301	0.247	0.324	0.262
2F	s(k)	200	200	200	200	200	200	200	200	200	-/-
2F	time	25	8	8	14	5	6	18	11	46	16
tL	vol	90.38	91.00	91.43	91.52	93.32	94.21	89.38	91.63	83.57	90.71
tL	ent	0.281	0.257	0.261	0.256	0.234	0.197	0.285	0.246	0.309	0.258
tL	s(k)	198	187	201	171	178	120	237	197	248	193
tL	time	105	60	71	60	71	43	94	57	252	90
tH	vol	90.24	90.85	91.41	91.41	93.42	93.89	89.79	91.13	83.47	90.62
tH	ent	0.280	0.256	0.264	0.256	0.223	0.200	0.285	0.246	0.308	0.258
tH	s(k)	225	291	196	196	268	120	275	180	296	227
tH	time	67	29	40	40	32	12	66	91	177	62
tS	vol	90.08	90.69	91.00	91.22	92.88	94.08	88.02	90.76	83.28	90.22
tS	ent	0.289	0.265	0.271	0.264	0.243	0.200	0.299	0.255	0.319	0.267
tS	s(k)	213	199	210	175	173	146	243	189	277	203
tS	time	74	36	35	39	36	23	71	38	180	59
vl	vol	93.00	93.00	93.00	93.00	93.00	93.00	93.00	93.00	83.50	-/-
vl	ent	0.252	0.225	0.233	0.231	0.250	0.226	0.260	0.213	0.322	0.246
vl	s(k)	688	616	506	402	113	32	735	501	208	422
vl	time	450	196	206	200	330	87	557	196	337	284
df	vol	89.02	89.70	89.92	90.43	93.50	93.24	87.82	89.92	82.05	89.51
df	ent	0.304	0.280	0.289	0.274	0.232	0.209	0.311	0.268	0.337	0.278
df	s(k)	122	117	104	108	200	67	150	113	262	138
df	time	43	19	17	22	24	10	42	19	98	33

Table 8.1: Comparison all configurations

Acknowledgements

At last I want to thank the people helped me to finish my master.
Thank you my brother Marian for all the discussions and support during my thesis, which helped my to sort my thoughts.
Thank you my mother Corinna and her partner Niels to help me to manage my everyday life.
Thank you my grandparents Helga and Heinz for any support, especially during my bachelor.
Thanks to my cat Amy for the moral support during my entire time on the university.
Thanks to my Coworkers, who always got along with all special-purpose solutions.
Thanks to my friends, who never doubted me.