INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Master's Thesis

Mesh based Scene Evaluation Metrics for LOD and Simplification

Daniel Kolb

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Master's Thesis

Mesh based Scene Evaluation Metrics for LOD and Simplification

Daniel Kolb

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	MNM-Team-Betreuer Dr. Christoph Anthes MNM-Team-Betreuer Markus Wiedemann
Abgabetermin:	25. Januar 2017

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 13. Januar 2017

. . . . . .
(Unterschrift des Kandidaten)

Abstract

I present seven metrics to quantify attributes of different meshes in a scene. Each metric represents a different geometrical or topological aspect of the mesh. The resulting rating values serve to convey the underlying complex data to the user. These allow the user to swiftly compare several features of multiple meshes. The metrics may thus guide users and programs during the process of mesh modification, i.e. optimization, simplification or smoothing, and scene modification as a whole.
I evaluate each metric individually by applying them to a sample scene. To examine the correctness and expressiveness of the metrics I compare the automatically calculated ratings to the raw base data. I find two of the metrics to be immediately useful and four of the ratings promising, but in need of adjustments. The remaining last metric, however, requires significant rework to generate useful data on par with the other six metrics.
This thesis first introduces the subject with a motivating example. It then presents important concepts and research on related topics. Afterwards it details the concept of the program and the mathematical considerations it is based on. It also lists my approach to solving the challenges which emerged during the implementation. Subsequently, the thesis focusses on the visualized output of the program and challenges said ouput. Finally, it contrasts the expectations and goals of each metric with the respective actual results.

Contents

1. Introduction

Even though performance of computers, CPUs and GPUs continues to increase [Rup14], so does the demand for more details, higher resolution and more dense meshes. As individual 3D models in video games surpass 100 000 polygons [Pir13], scene optimization algorithms have never been more beneficial. Another advantage of optimized 3D models lies in their reduced memory requirements. This holds especially true for applications running on mobile devices, as smart phones and other mobile gaming devices already provide little memory space due to size constraints. Furthermore, the memory capacities of mobile devices can not be enhanced as almost arbitrarily as desktop computers or laptops.
The memory space of Nintendo's 3DS, for example, employs SDHC memory cards, which can provide up to 32 GB [Ass16] for downloaded software, while its game cartridges carry a maximum capacity of 8 GB [Yeu10]. With the current 7th generation of Pokemon bringing the number of individual pocket monsters (including alternate forms) well above 800. This means that on average for every 10 MB on a 3DS cartridge of maximum memory space all data relevant to a single Pokemon need to be present; especially its mesh, its textures and its animation. Thus it is in the best interest of the game developers to optimize the memory usage of their 3D models in order to save the memory space needed for further required game data, like sounds, music, character models, game mechanics etc.
With the release of Pokemon GO the very same problem is present with smart phones. Many of today's smartphones come with a memory space of 32 GB in their basic models (iPhone 7 [Inc16], Samsung Galaxy S7 [Mar16]). However, these 32 GB of internal memory are only partially available for installed apps as the provided memory space is also used for pictures, videos, music, pre-installed apps as well as the operating system itself. Thus, in order to maximize the reachable audience, the memory requirements of an app need to be minimized. Otherwise, one would risk alienating users unwilling to invest non-unsubstantial amounts of money to upgrade their model's memory. Once again mesh optimization is a critical step during the development of the game. However, unguided reduction of the complexity of the game's models would lead to noticeable differences in the perceived quality of the individual models, unless the equality of the levels of detail of all models is ensured.
Consequentially optimization of individual meshes is needed to maximize the number of visual details per polygon count. Moreover the perceived fidelity of a rendered scene can be promoted by adapting level of details (LODs) to the user's needs [FS93]. Both applications, however, need ways to gauge a mesh's LOD, complexity or redundancy to enable easy comparison of the mesh to different versions of itself as well as to other models within the current scene.
Hence, I propose specialized, yet standardized metrics to rate 3D meshes based on intrinsic attributes. Specialization aims to preserve the informative value of a single rating, while standardization is needed to keep ratings expressive when comparing different metrics as well as identical ratings of different meshes. The calculation of these metrics and their subsequent rating values are conglomerated in the Scene Analyzer. It automatically analyzes and rates provided scenes.

During this paper I first detail the work previously achieved in the field of mesh analysis as well as concepts relevant to my research. I then present the structure and the mathematical keynotes of my scene analysis. Afterwards I explain the challenges I faced during the implementation of the intended algorithms and how I solved or circumvented them. I subsequently provide the empirical findings and the actual automated analysis of several scenes and multiple diverse meshes as well as the examination of the results. At the end I inspect the positive and negative results, discuss possible solutions for found shortcomings and explore possible future research topics and applications of the Scene Analyzer.

2. Fundamentals and Related Work

In this chapter I discuss notions relevant to my thesis, both the direct and fundamental kind as well as the indirect and related kind. I cover groundwork which introduces new concepts, which I then transfer to fit the problem adressed by this thesis, and other solutions to variations of said problem.

2.1. Fundamentals

This section contains mathematical approaches relevant to my thesis which are neither well-known common techniques nor trivial in their execution and relevance. I detail each method, their origin and provide an example to visualize their use.

2.1.1. Planar Grouping

Planar grouping is a segmentation algorithm which subdivides a mesh based on the similarity of attributes of adjacent triangles. Planar grouping originates in the gathering of polygons in *Near-Coplanar Sets* using a representative tree [HH93] as a means of *Polygonal Reduction* and *Unseeded Region Growing* [LJT01] from the field of image analysis.

Near-Coplanar Sets are created by adding polygons whose normals $\vec{n_i}$ differ from a set's representative normal $\vec{n_r}$ by at most 2ϵ:

$$cos^{-1}\left(\frac{\vec{n_i} \circ \vec{n_r}}{|\vec{n_i}||\vec{n_r}|}\right) \leq 2\epsilon.$$

These representative normals $\vec{n_r}$ can either be the mean average of the normals of all polygons contained within the set or chosen by the user. Both variants can lead to bad fits by barely including a polygon p_1 with its normal $\vec{n_1}$ in a set S and barely excluding an adjacent polygon p_2 with its normal $\vec{n_2}$ from S. Even though $\vec{n_1}$ is more similar to $\vec{n_2}$ than to the normals of the other polygons in S, p_1 and p_2 are not part of the same set. Fig. 2.1 presents one such case: p_1 and p_3 are part of the same *Near-Coplanar Set* S constructed around the representative normal with an angular criterion of $2\epsilon = 50°$. Due to p_2 not meeting the angular criterion, it is not part of S. This means that p_1 and p_3 are grouped together while their respective normals differ by 90°, whereas p_1 and p_2, which differ by only 18°, are not.

Region Growing is a technique which can be utilized to dissect and analyze an image based on the pixels' attributes. This method grows regions by adding pixels to a region, if the difference d in color values of an edging pixel px_1 and an adjacent regionless pixel px_2 doesn't exceed a given threshold t:

$$d(px_i, px_j) \leq t.$$

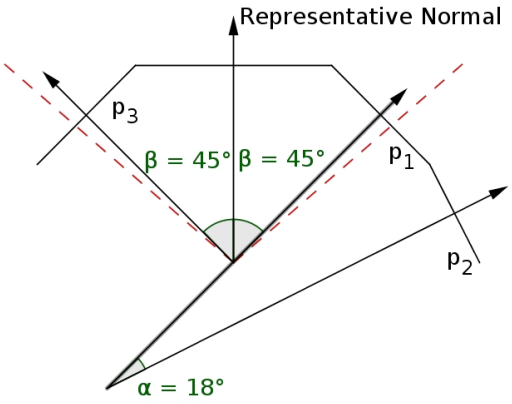


Figure 2.1.: Near-Coplanar Set S containing p_1 and p_3, but not p_2

Unseeded Region Growing eliminates the otherwise present need for the user to choose starting pixels around which the regions are grown by fusing regions. Thereby the resulting grouping becomes independent from the choice of the starting pixel. Fig. 2.2 shows a grayscale image of 6*6 pixels. The colors of the image are encoded as integers. For the sake of this example I assume ten different shades of gray, ranging from 0 to 9. Of these ten unique values, seven are present in the given picture. I apply *Unseeded Region Growing* with a threshold of $t = 1$ to the image, which led to the segmentation into two disjunct sets, S_1 and S_2. As the difference in color values along the border of S_1 and S_2 exceeds t, the two regions can't be joined. In this example S_1 represents the darker region, whereas S_2 contains the brighter pixels, and their border signifies a sudden disparity in the otherwise gradual variation of values.

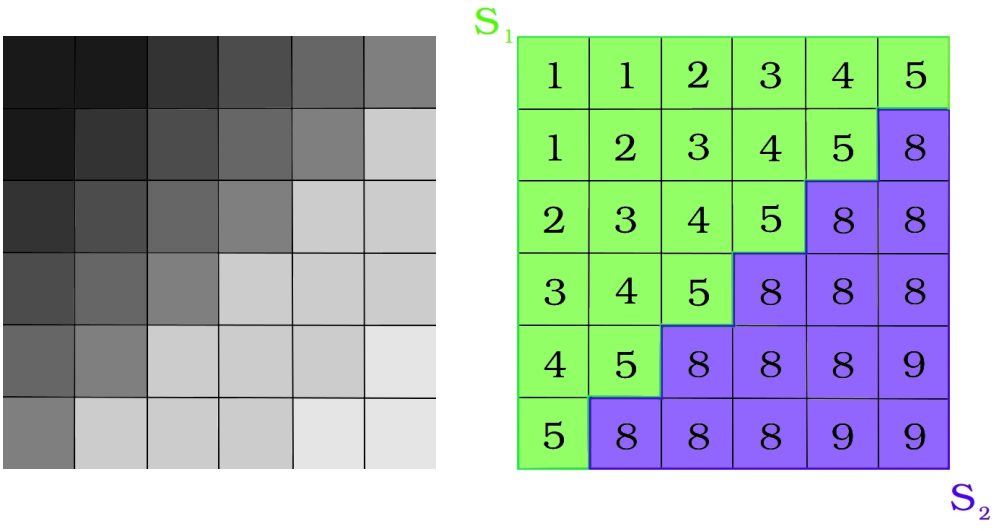


Figure 2.2.: A 6*6 px image and its segmentation by *Region Growing*

2.1.2. Gray Scale Co-Occurrence Matrix

Another utilized approach originating from *Image Analysis* is the Gray Scale Co-occurrence Matrix (GSCM) [HS+73]. It aggregates spatially dependent appearances of color values. Hereby a given image is analyzed along a specified direction $\vec{d}$ and distance Δ. This means that each pair of two pixels (p_i, p_j) counts as one co-occurrence if p_j can be reached from the position (x, y) of p_i by moving at most Δ pixels in direction d. Each co-occurrence then increments the co-occurrence value of the respective color values of p_i and p_j in the GSCM C at position $[color(p_i), color(p_j)]$ by one.

$$C[color_i, color_j] = \sum_{x=0}^{n} \sum_{y=0}^{m} \begin{cases} 1 & \text{if color(x,y)} = color_i \text{ and color(x+}\Delta_x\text{,y+}\Delta_y\text{)} = color_j, \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 2.3 shows the GSCM resulting from the example image previously introduced in Fig. 2.2: The underlying color scale has ten unique shades, which leads to a 10x10 Matrix. The analysis was carried out along the horizontal direction and $\Delta = 1$, although the approach is not limited to this kind of co-occurrence. Therefore the image was examined by column-wise counting directly neighboring pixels. The cells of the completed GSCM C contain the absolute quantitiy of co-occurrences: $C[6, 5]$, for example, equals 5. This means that there exist 5 instances within the image of a pixel with the color value of 4 having a pixel with the color value of 5 as immediate neighbor to its right. Due to the small size of the image, the total number of co-occurrences is, when compared to the size of the GSCM, low and the matrix itself is very sparse. Nevertheless the GSCM delivers viable results: All non-zero values lie either directly on the main diagonal of the matrix or close by. This signifies an image of low frequency and gradual changes as spatially close pixels share similar colors. A high frequency image, on the other hand, would result in most values converging towards the counter diagonal, which represents sharp edges. This is due to proximal pixels having contrasting colors.

1	1	2	3	4	5
1	2	3	4	5	8
2	3	4	5	8	8
3	4	5	8	8	8
4	5	8	8	8	9
5	8	8	8	9	9

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 2.3.: A 6*6 px image and its Co-occurrence Matrix

2.2. Related Work

This section deals with previous work containing ways to quantify geometrical and topological characteristics of a mesh. While my thesis does not make direct use of these algorithms and metrics, it does share some of the underlying ideas. This also serves to supply the inclined reader with further research relevant to the field of mesh analysis.
I find that metrics for rating geometrical and topological features to be mostly present in scientific papers dealing with one of two fields of research: Mesh segmentation, which requires a way to differentiate mesh sections based on a specific measure, and mesh optimization, which needs to evaluate the complexity of a mesh in order to guide the optimization process. The metrics here listed may also, after some neccessary adjustments towards standardization and normalization, be suited as additions to my own metrics.

2.2.1. Mesh Segmentation and Segment Classification

A very important contribution to my thesis comes from the research of Lee et al. [LVJ05]. Their paper addresses the quantification of the individual importance of a mesh's subregions to the user by transferring the previously established approach to Image Saliency by Itti et al. [IKN+98]. Lee et al. solved this problem by distinguishing between smooth consistent regions and curved regions. The authors further propose to take the surrounding regions into account when calculating the saliency of a vertex: a curvature of a small group vertices in an otherwise planar region is perceived as more important than a repeating pattern of curved groups of vertices. Hence, the importance of a vertex can be seen as the amount of information it carries by diverging from the pattern in its surroundings. Vertices, which are similar to their surroundings, on the other hand, represent partially redundant information, as they create a shape which the user expects. Those vertices are consequently of less importance.
When comparing the attributes of a vertex to those in its vicinity, Lee et al. opted to use a Gaussian filter to weight proximity instead of scaling distance linearly. Furthermore, the saliency of a vertex is the result of multiple saliency calculations which use different radii to determine the extent of the surroundings of the vertex. These individual results are weighted and aggregated into a single normalized value, which represents the final saliency value of that vertex in the mesh. The authors used statistical measures in the calculation of the curvature deviation of each vertex by comparing the bend of a vertex to the mean and standard deviation of the curvature of the surrounding vertices. The so generated saliency map of the mesh can be used to simplify a mesh. The eligibility for deletion of a vertex is inverse proportional to its saliency, which represents its importance and contribution to the structure and shape of the mesh. Fig. 2.4 shows the saliency of a 3D model of a male bust. In this visualization cold colors represent low saliency, whereas warmer hues highlight regions of high saliency. The wrinkles of the garment, prominent facial features and the ear get emphasized, while the smooth bald head and the neck are colored deep blue due to their low curvature variation.

Further interesting work was provided by Shilane and Funkhouser [SF07] by their proposal for segmenting meshes and measuring the importance of each segment for the mesh classification. The authors accomplished the partitioning of arbitrary meshes by constructing spheres around sample points of the 3D object. Every part of the object within the set radius around a sample point is then a component of that partition. They were then able

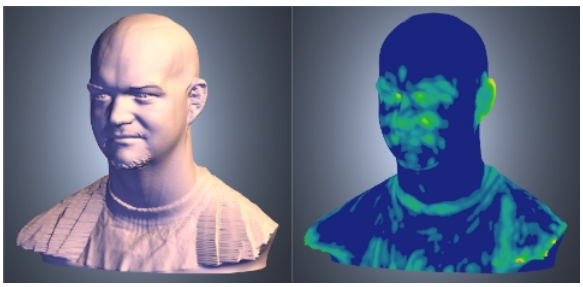


Figure 2.4.: Mesh Saliencies [LVJ05]. Warm colors represent locally deviating surface curvatures.

to describe these spherical regions with a Harmonic Shape Descriptor (HSD) [KFR03]. A HSD is a vector which contains the amplitudes of the harmonic decomposition for a shape describing function. Shilane and Funkhouser applied the process to a wide array of 3D objects, which were also divided into several groups matching the representated object. Thus they could identify the shape descriptor which were most similar to meshes of the same class and dissimilar to meshes of different classes, i.e. most distinctive. The partitions of a mesh corresponding to the distinctive shape descriptors are consequentially of higher significance to the recognizability of the object. The distinctiveness of each vertex of a mesh could therefore be generated by mapping the distinctiveness of the shape descriptors onto the mesh. A visualization of the distinctive regions of several examplary meshes can be seen in Fig. 2.5.

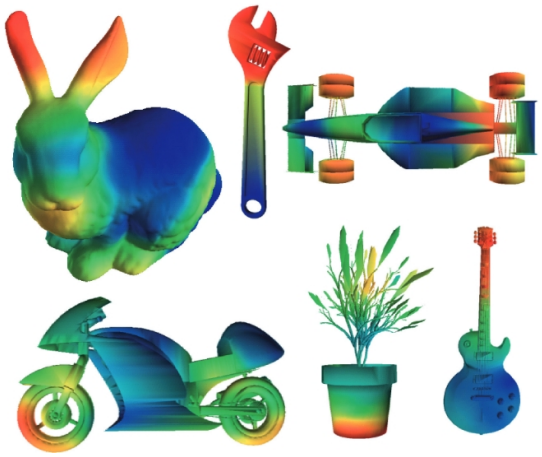


Figure 2.5.: Distinctive Regions of 3D surfaces [SF07]. Heatmaps show the mesh regions most relevant to object classification.

However, Distinctive Regions serve a different purpose as Mesh Saliency. Distinctive Regions represent a wholistic rating of vertex signficance, whereas Mesh Saliency measures local significance. It could therefore be used to quantify the similarity of different LODs of the same 3D object by rating how well the Distinctive Regions were preserved during mesh simplification. This in turn would help describing the concrete change of object quality as perceived by the user upon switching LODs. Similarly, Distinctive Regions can be used in guiding of mesh optimization, e.g. by displaying to a 3D artist which parts of a mesh can be reduced without hazarding the shape.

Podolak et al. [PSG+06] also developed an interesting approach to mesh analysis and mesh segmentation based on symmetry. The authors proposed a random sampling of pairs of points on the surface of a 3D object, constructing a plane of symmetry for each pair and then evaluating how well the plane reflects further surface points. The suitability of each such plane is calculated by mirroring surface points along the plane and aggregating the quadratic errors generated by the deviation of mirrored symmetry points to closest surface points. Due to the use and rating of several possible planes of symmetry the algorithm detects global symmetries as well as strong local symmetries. Regions of a mesh which lead to strong planes of symmetry can then be identified and highlighted. The application of the algorithm to the Stanford bunny (see Fig. 2.6) highlights parts of its ears, its face and its chest as the most symmetrical regions. The red points display the arbitrarily chosen surface points which were reflected through the plane. The green points are the closest matches to these reflected points.

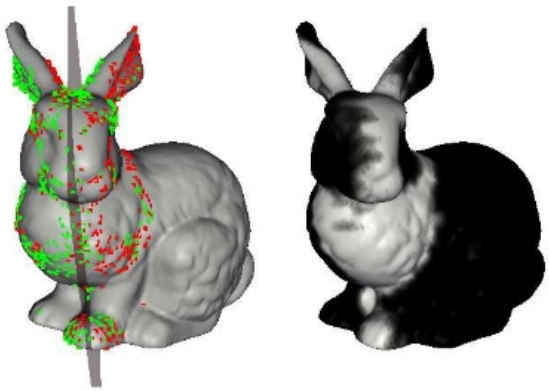


Figure 2.6.: A planar-reflective symmetry transform for 3D shapes [PSG+06]. Left shows the chosen points (red) and their reflections (green). Right highlights the most symmetric regions.

The results of this kind of mesh analysis can be used in several ways. First, meshes can be divided and segmented according to their local symmetry. These mesh segments could then be analyzed and rated individually, instead of the mesh as a whole. A mesh rating could then be aggregated from the individual segment ratings. Secondly, the data garnered from the mesh symmetries can be used to quantify the information gained when viewing the 3D object from a certain perspective. Presenting the symmetrical segments displays less information than a rotation showing the asymmetrical aspects. Thirdly, the mesh symmetry is another geometrical attribute which can be used to compare different LODs of a single 3D object and describe their (dis-)similarity. A loss of strong symmetries would equal a loss of important mesh features. Lastly, the presence of multiple strong symmetry planes sharing a single intersection means that the surface of the mesh is very even. The absence of strong symmetry planes or common intersections, on the other hand, represents an uneven, complex mesh. The planar reflective symmetry transform could thus also be used to examine mesh complexity. However, the runtime complexity of the algorithm when not relying on approximations is extremely high at $O(n^5 \log n)$.

Another way to segment a mesh, though not based on local features and instead using the occurrences of random cuts, was proposed by Golovinskiy and Funkhouser [GF08]. The authors determine edges of the mesh which were suitable as segment boundaries by applying a number of random cuts and then evaluating the likelihood of each edge to lie on a random cut. This algorithm thus emphasizes comparatively narrow parts of a mesh and part boundaries. Identifying these edges allows for a segmentation of the mesh in individual parts for further analysis. As this approach does not investigate fluctuation of specific local features its resulting segmentations can be used uniformly. Fig. 2.7 shows its application to the Stanford Bunny, a trophy, a skeletal hand and a chair. Each uniquely colored region represents a distinct segment of the mesh.

Figure 2.7.: Randomized Cuts for 3D Mesh Analysis [GF08]. Each distinct mesh segment is colored uniquely.

With no mesh-specific values being required for a succesful analysis, this segmentation algorithm has the benefit of functioning without prior knowledge or analysis. Additionally, the runtime complexity amounts to $O(VE^2)$ for a mesh with the vertex count V and an edge count of E. While this makes it unsuitable for real time analysis, it is on par with comparable tools for preparatory steps preceeding the real time analysis relying on solving max-flow min-cuts or all-pairs shortest paths problems. However, with the algorithm not using specific features to group polygons and its random nature, Randomized Cuts can clearly not be used for feature-based partitioning or analyses requiring consistent input values.

Semantical or feature-based segmentation of meshes also finds immediate use in locating optimal viewpoints. Takahashi et al. [TFTN05] solved this problem by partitioning a mesh into subvolumes based on their topological features. Each viewpoint, and thereby a specific orientation of the 3D object relative to the user, is rated based on how much of each subvolume is visible. The authors chose the value set [0;1] to quantify the relative quality of any given viewpoint in comparison to the best and worst viewpoints. Viewpoints which lead to a subvolume occluding other important subvolumes or a large portion of itself are thus rated low. The choice of such a normalized value set makes it possible to also evaluate a viewpoint in regard to a scene containing multiple 3D objects by aggregating their respective ratings. Takahashi et al. also included a weighting factor to account for partial opacity and transparency of a volume. Fig. 2.8 shows the ratings of diverse viewpoints for a 3D horse model.
This algorithm can inversely also be used to describe how well a 3D object can be seen by a user based on its given rotation and the user's position and viewing perspective. By evaluating how much information of the mesh is currently being presented to the user, a LOD switch may be pushed up or pushed back. Additionally, this metric already operates on a normalized value range which makes it highly compatible to other normalized metrics.

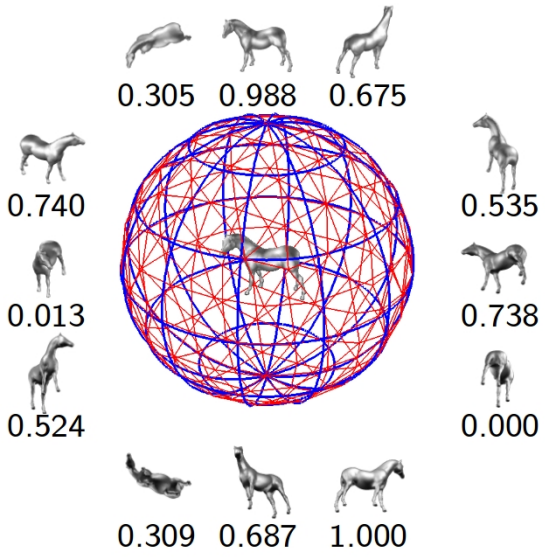


Figure 2.8.: A feature-driven approach to locating optimal viewpoints for volume visualization [TFTN05]. Each viewpoint is rated according to their relative quality.

2.2.2. Simplification Algorithms

Of further interest are algorithms which are able to detect regions of a mesh eligible for simplification. Optimally those simplifications would not even result in a decrease of detail or important features. A detection of large regions of this kind or a high detection rate thus imply that the analyzed 3D object has a high level of detail or an opportunity for optimization. These ratings could therefore be used for both LOD evaluation and guiding the user's mesh modification process.
An early example for a metric which quantifies the simplifiability of a mesh can be found in the Mesh Optimization algorithm by Hoppe et al. [HDD+93]. The core of this algorithm is detecting the overall shape of the mesh and the ensuing removal of vertices while minimizing the progressive deviation from the original shape. To guarantee a gradual increase of mesh simplicity while retaining topological features, the mesh is evaluated by an energy function E. The authors use the function E to measure the amount of vertices, the distance of the vertices and the deviation from the original point set. Each iteration of the algorithm applies a random simplification operation - edge collapse, edge split or edge swap - to a random edge of the mesh. If this results in a decrease of E, the simplification step is accepted and the iteration resumes. If the algorithm otherwise fails several times to reduce E, it terminates. Other non-random alorithms for selection of the vertex reduction and its application can be used as well. For example, instead of choosing a random edge of all available edges, the authors also suggest limiting the candidate set to include only edges which could lead to a decrease in E. This candidate set could even be sorted by its anticipated energy decrease. Fig. 2.9 shows five input meshes and the respective results of the Mesh Optimization.

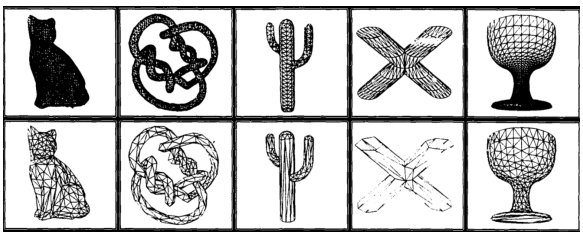


Figure 2.9.: Mesh Optimization [HDD+93]. Original meshes and their vertex reduced versions.

The algorithm aims to minimize the vertex density of a mesh while maintaining the mesh's overall structure. It is therefore a suitable method to either help create lower LODs for a given mesh or to rate the redundancy of the vertices used to generate a mesh regarding the mesh's shape. However, as the energy function E scales linearly with the vertex count, it is not normalized and therefore would need to be modified to enable comparison of diverse 3D object.

Another effective method for creating LOD libraries for a given mesh was designed and implemented by Schroeder [Sch97]. This approach, too, makes use of previously established simplification operators: edge collapse and vertex merge [HDD+93]. However, the algorithm also relies on the inverse operators, edge split and vertex split, to revert a previous simplification step and once again increase mesh complexity. This enables the algorithm to create and retrace a set of progressively simpler meshes as each simplification results in a decrease of vertex and/or edge count. The choice of which vertex (and its corresponding edges) is modified is made by examining the so introduced topological deviation from the original mesh. The operation on the vertex which leads to the least error gets executed. This also allows the user to limit the mesh modification by providing a maximum error. The algorithm terminates as soon as it can not go below this error threshold. Otherwise the algorithm continues to create progressively less complex LOD instances until there are no more edges left. The algorithm aims to create meshes with arbitrary levels of reduction. This ultimately enables it to change the topology of a mesh by changing and even closing holes. An example for this desired behavior can be seen in Fig. 2.10.

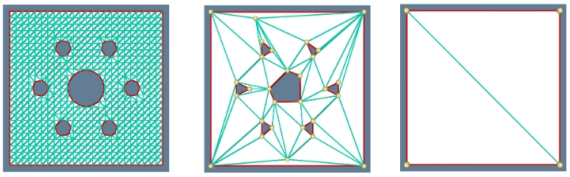


Figure 2.10.: A topology modifying progressive decimation algorithm [Sch97]. Left: Unmodified plane with holes. Center: Partially reduced plane with changes in the shape of the holes. Right: Maximal simplification with elimination of the holes.

This method provides the following two metrics for measuring mesh complexity and level of detail. First the pair of vertex count and edge count of a mesh, which directly relates to rendering time. Secondly, the error value generated by the progressive simplification.

Together, the metrics succeed in quantifying the decrease in required rendering time as well as the introduced loss of mesh quality; neccessary steps to evaluate a potential change in level of detail for a given 3D object during real time visualizations. However, these values do not explicitly account for the changes in the mesh topology in regards to the fidelity of the mesh. Furthermore, both metrics are only useful when comparing different LODs of the same mesh, since the rating values carry information relative to the initial, unmodified mesh. The error value also requires a full execution of the algorithm as its calculation is dependent on previous simplification operations.

Finally, Karni and Gotsman [KG00] show that treating a mesh, more precisely its vertices, as characteristics of trigonometric functions allows for the application of established frequency analysis tools; most importantly the Fourier transform [Tau95]. The algorithm uses the adjacency matrix of the mesh's vertices and their position vectors to describe topological fluctuation. The authors use the eigenvalues of the Laplacian Matrix to describe the frequency introduced by the corresponding vertex. A high eigenvalue and thus a high frequency means that the respective detail generated by the vertex is vastly different from the surrouding structure. A low frequency, on the other hand, implies a smooth surface feature. Therefore, a frequency analysis can be used to examine the unevenness of a mesh. Consequently, filtering of the basis functions by including only the n lowest frequencies and the subsequent reconstruction of the mesh using the corresponding n vertices serves as a means for mesh reduction while preserving the overall structure. An increase in the frequency threshold n allows the gradual inclusion of further details in order of prominence. An iteratively more detailed reconstruction of a 3D horse model can be seen in Fig. 2.11.

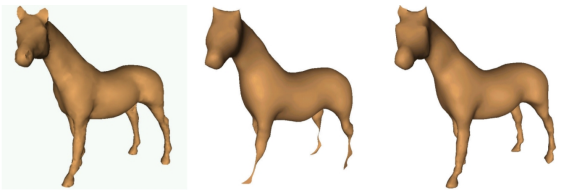


Figure 2.11.: Spectral Compression of Mesh Geometry [KG00]. Left: Unmodified model with 2978 vertices. Center: Reconstruction with 100 vertices. Right: Reconstruction with 200 vertices.

This motivates further research regarding the frequency analysis of additional features and attributes of the mesh to investigate mesh evenness under different aspects. Unlike the Co-occurrence Matrix the Fourier transform is able to detect changes on a global scale, whereas the analysis of the Co-occurrence Matrix is limited to neighbors of the nth degree. Unfortunately, the runtime complexity of the Fourier analysis is prohibitively large. The authors themselves had to segment the mesh beforehand and apply their algorithm to each segment separately.

3. Concept

This chapters deals with the design of the Scene Analyzer. It explains the individual steps, and their respective tasks, which encompass the program. This also includes the mathematical ideas behind the metrics and the visualization.
The ratings used to analyze the scene can be divided into two categories: Those, which use data generated during runtime, such as user position and orientation, and those, which remain unchanged during runtime. As the second, static category contains several computationally and memorywise expensive algorithms and as the results are invariant to subsequent executions of the Scene Analyzer, they are precalculated. Upon precalculation of a given scene, the results are written to files, which can then be read and reused indefinitely. This not only saves time and memory space during ensuing executions but also reduces the run time complexity of the Scene Analyzer to *O(n)*, given that the runtime exclusive metrics adhere to this limit as well. When supplied with the analysis files for the current scene, my application therefore enables the user to evaluate that scene.
The *Initializer* represents the first part of the scene analysis, encompassing all the steps relevant to the precalculations, which can be seen in Figure 3.1. It is structured as follows: The *User* supplies the program with one or several *3D Input Files* to be analyzed as a scene. The data provided by these files of possibly different file formats is then decoded and transformed by the *File Parser* into uniform geometrical data, such as vertex coordinates and normals. Additionally the user may opt to specify *Analysis Specifications*, for the Metrics. If the user provides no parameters, default values are used instead. Both inputs are then used to calculate the *Runtime Invariant Metrics*. Further external algorithms for the *Frequency Analysis* are also incorporated in the analysis. Upon execution of the analysis the resulting ratings as well as relevant meta-data are saved in binary *Scene Analysis Files* for each mesh.

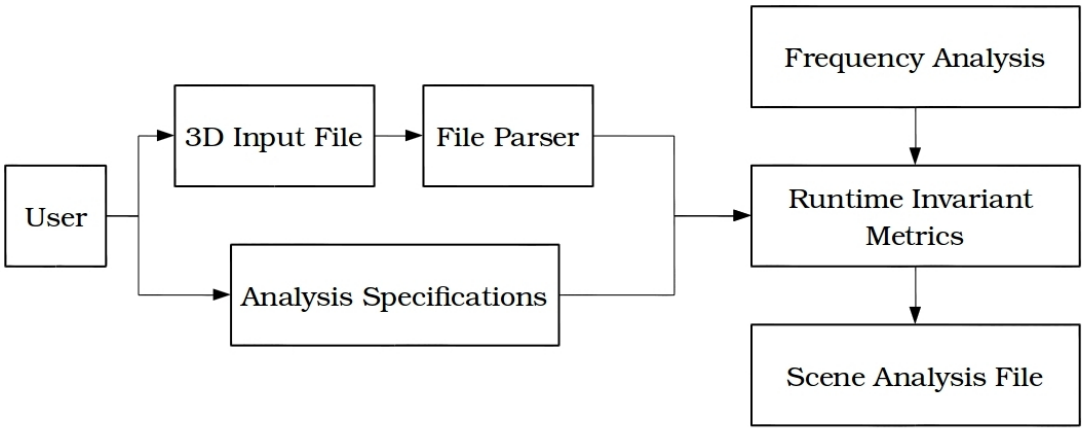


Figure 3.1.: Concept Initializer

Once execution of the *Initializer* has finished succesfully and a *Scene Analysis File* has been generated for each mesh of the scene, the realtime visualization by the *Scene Analyzer* (see Figure 3.2) can be executed. The first step involves the calculation of the *Runtime Exclusive Metrics*. This mirrors the procedure of the *Initializer* with the *User* providing the 3D Input File to the File Parser and combines the output with Analysis Specifications, e.g. the user's position and viewing direction, to determine the non-static and runtime dependent metrics. The processes of metric calculation differ only in the Initializer requiring a tool to facilitate Frequency Analysis (cf. Figure 3.1), whereas the Scene Analyzer does not. The results of these *Runtime Exclusive Metrics* are then combined with the *Runtime Invariant Metrics* provided by the *Scene Analysis File*, which was generated by previous execution of the *Initializer*. At that point all Metrics used in the analysis of the scene are finally present. The user has the option to supply each of these rating with *Rating Weights* suiting their needs to emphasize, diminish or even ignore different attributes of each mesh. The *Aggregation* of these data then results in single values representing the combined ratings for each mesh. Each aggregated rating is then mapped on a RGB-value and afterwards set for each mesh during *Coloration*. The eventual scene is comprised of the original *3D Input File*, imported by the *File Parser*, and the modifications to the color values of each mesh. Lastly, the *User* is able to *Navigate* and explore the *Displayed Rated Scene*.

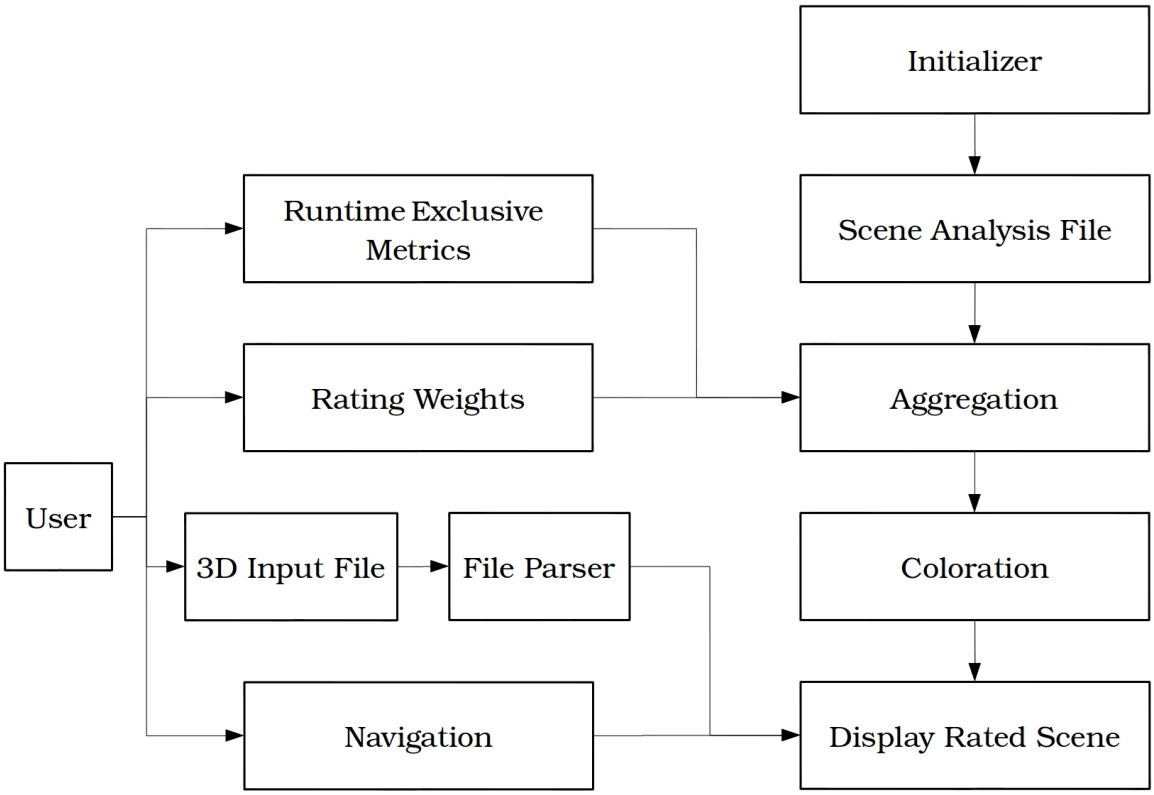


Figure 3.2.: Concept Scene Analyzer

3.1. Metrics

The following subsections detail each of the seven metrics I designed to analyze a scene by allowing easy comparison of the included meshes by rating their geometrical features. For each metric I specify the exact mathematical formula, the origin and computation of the variables used therein, its value set, its intended meaning as well as exceptional and fringe cases.

3.1.1. Polygons per Planar Group

Planar Grouping is used to partition all present polygons composing a mesh m. Each Planar Group $pg \in PG$ contains neighboring polygons. A new polygon p_1 is added only under the condition that its normal $\vec{n_1}$ differs no more than a certain amount of degrees α from the normal $\vec{n_2}$ of a previously added polygon p_2. p_1 and p_2 also need to share a common edge. In case no suitable pre-existing Planar Group is found, p_1 is assigned to a new and until then empty Planar Group. If p_1 fulfills the conditions of multiple Planar Groups, these join on p_1 and form a singular, bigger group. Once iteration over all polygons has finished, each polygon is member of exactly one Planar Group. A partitioning with $\alpha = 0$ can be employed to identify cases of plane surfaces using multiple planar triangles. The resulting quantity of Planar Groups $|PG|$, when compared to the number of polygons n, represents the proportion of "needed" polygons to used polygons. Thus, I define the metric as

$$R_{pppg}(m) = \frac{n - |PG|}{n - 1} \in [0; 1], |PG| \in [1; n].$$

If all triangles' normals differ from their neighbors' normals by at least α, all Planar Groups are singletons. Hence, $|PG| = n$, which leads to $R_{pppg} = 0$. Partitionings of the polygons in decrementally fewer and incrementally bigger Planar Groups results in higher R_{pppg} values. $R_{pppg} = 1$ means that all polygons are part of the same Planar Group, as $|PG| = 1$.
This metric uses only one mesh's geometrical data and can thus be precalculated. Its rating values are not influenced by other meshes or the user's movement.

3.1.2. Planar Group Variation

Another analysis utilizing the above established structure compares the sizes of the individual Planar Groups, which R_{pppg} completely disregards. By summing the squares of all normalized Planar Group sizes pg I aim to rate meshes with few proportionately big Planar Groups higher than those with multiples of small or medium size. This results in the following calculation:

$$R_{pgv}(m) = \frac{\sqrt{\sum_{pg \in PG, |pg|>1} |pg|^2}}{n} \in [0; 1], |PG| \in [1; n], |pg| \in [1; n].$$

Once again, a rating of 0 represents a mesh with absolutely no occurrences of co-planarity, as singleton Planar Groups, which represent the best case sceneario, are disregarded. Since the metric is heavily biased towards big Planar Groups, high ratings can only be achieved by meshes with a significant degree of co-planarity. The most extreme of these cases, a two-dimensional plane, results in $R_{pgv} = 1$.
As this metric relies on the exact same data as R_{pppg}, it can be pre-calculated as well.

3.1.3. Relative Density

A simple metric to determine complexity of 3D meshes lies in comparing their respective number of polygons n and provided surface S. It incorporates linear interpolation in three distinct intervals, I, II and III, determined by the range of densities of all meshes $m \in$ M of a scene.
First, I calculate the average amount of polygons per unit of area for each mesh. I define the resuling quotient as density d_m of each respective mesh m:

$$d_m = \frac{n_m}{S_m}.$$

I use the density values of all meshes in the scene to calculate the mean average density μ of the scene and its standard deviation σ. These values are defined as

$$\mu = \frac{\sum_{m \in M} d_m}{|M|}, \quad \sigma = \sqrt{\frac{\sum_{m \in M} (\mu - d_m)^2}{|M|}}.$$

Additionally, I identify the smallest density value $d_{min} = \min_{m \in M} d_m$ and the highest density value $d_{max} = \max_{m \in M} d_m$ within the scene. With these data I construct the three density intervals.
The central interval, II, encompasses all density values which are part of the mean deviation σ environment around the mean density μ. It produces linearly interpolated ratings ranging from -0.5 to 0.5. If there exist density values which undercut the lower bound of this interval then these values create another interval, I, with its smallest value d_{min} as its lower bound. Interval I creates ratings in the range [-1;-0.5[by linear interpolation, albeit possibly with a different gradient as the interpolation used in interval II. An analogous addition on the right side of interval II occurs if d_{max} exceeds $\mu + \sigma$. This interval III generates ratings ranging from 0.5 to 1. The compounded equation is as follows:

$$R_{rd}(m) = \left\{ \begin{array}{ll} -0.5 - 0.5 \cdot \frac{d_m - \mu + \sigma}{d_{min} - \mu + \sigma} \in [-1; -0.5[& \text{if } d_m \in [d_{min}; \mu - \sigma[\\ -0.5 + \frac{d_m - \mu + \sigma}{2\sigma} \in [-0.5; 0.5] & \text{if } d_m \in [\mu - \sigma; \mu + \sigma] \\ 0.5 + 0.5 \cdot \frac{d_m - \mu - \sigma}{d_{max} - \mu - \sigma} \in]0.5; 1] & \text{if } d_m \in]\mu + \sigma; d_{max}] \end{array} \right\} \in [-1; 1].$$

If all meshes have the same density, however, σ would be 0 and $d_{min} = d_{max} = \mu$. To avoid division by zero, R_{rd} is instead set to 0 for all meshes.
The relative density metric relies on the composition of the analyzed scene, particularly the meshes included. This information is only available once the execution of the Scene Analyzer has begun. Therefore, R_{rd} can not be precalculated. Due to the resource light implementation the calculation can easily be done in real time during the program's execution.

3.1.4. Distance

Level of detail environments can rely on the distance Δ of 3D objects to the virtual camera to classify the objects' importance. I rate each mesh by its distance to the user, relative to largest distance $\Delta_{max} = \max_{m \in M} \Delta_m$ and smallest distance $\Delta_{min} = \min_{m \in M} \Delta_m$ within

the scene. I use linear interpolation to calculate the individual distance ratings. This results in

$$R_d(m) = \frac{\Delta_m - \Delta_{min}}{\Delta_{max} - \Delta_{min}} \in [0;1].$$

Thus one can combine the attribute *distance* with several other metrics operating on the same domain when ranking the user's need for a higher level of detail of meshes. The time needed to calculate the distance to each object can be decreased by conservatively approximating the object; for instance via its own bounding box. In case of all meshes having equal distance to the user the distance rating is unable to deliver meaningful data and rates each mesh with a neutral 0.5.
Since this metric requires both all meshes of the scene to be analyzed as well as the user's position, it can not be precalculated and instead needs to be determined during runtime.

3.1.5. Contrast of Co-occurrence Matrix

Gray-level co-occurrence matrices are a powerful tool in the area of image analysis to determine spatial relationships of data points. Transferring the underlying idea from 2D color data consisting of a fixed range of values to 3D data of variable kinds and dynamic ranges requires a definition of spatiality on Meshes and a way to reinterpret the column and row indices of the matrix.
By drawing from the previously implemented neighborhood of polygons, I determine the spatial relation of two polygons p_1, p_2 as being neighbors of a certain degree. If p_1 and p_2 are direct neighbors, then they are neighbors of the first degree. If p_1 and p_2 require a common neighbor p_3 to reach each other, they're indirect, second degree neighbors. For the sake of simplicity, I limit the evaluation to direct, first degree neighbors.
Unlike pixels, polygons offer more attributes suitable for analysis than color alone, especially geometrical data like area, circumference or normals. Color values, however, are confined to a static limited value set - such as RGB, RBY, grayscale - irrespective of the occurrence of these colors. Therefore modelling a co-occurrence matrix M for each possible pair of neighboring area values would be infeasible given the infinity of possibilities. Thus I limit the range to those values, which are positively present within the mesh. A sorted set S with $dim = |S|$ of attribute values serves as a mapping of attribute values to matrix indices.
The co-occurrence matrix M is filled by iterating over all polygons p_i and their respective neighbors p_j. A lookup in S delivers the position of the attributes of these polygons in the ordered list, enabling us to increment $M(S(p_i), S(p_j))_{new} = M(S(p_i), S(p_j))_{old} + \frac{1}{|N_i|}$ and normalizing the increase by dividing by the number of polygons in p_i's neighborhood N_i. During the summation of the entries of M I supply each summand with a weight. These weights are the result of the squared respective difference in the chosen attribute: $weight = (attribute_i - attribute_j)^2$. Finally I normalize the sum by dividing the outcome by n, which equals the number of entries in M. Further division by the squared difference of the highest and the smallest attribute value in S accounts for the previously added weights: $normalization = n \cdot (\max(S) - \min(S))^2$. This prevents a scaling of the attribute values from influencing the rating. Thus, the complete equation is

$$R_{ccm}(m) = \frac{\sum_{i=1}^{dim}\sum_{j=1}^{dim} M(i,j) \cdot weight}{normalization} = \frac{\sum_{i=1}^{dim}\sum_{j=1}^{dim} M(i,j) \cdot (attribute_i - attribute_j)^2}{n \cdot (\max(S) - \min(S))^2} \in [0;1].$$

By design the metric simply returns 0 if all triangles are of the exact same size. Otherwise the result of R_{ccm} would lead to a division by zero due to $max(S) = min(S)$. I chose the value 0, as a mesh having $max(S) = min(S)$ features the maximum possible uniformity. The highest rating, on the other hand, would be the result of a mesh containing but two different kinds of triangles with each triangle being surrounded by only triangles of the respective other kind.
As the metric features both quadratic runtime and memory usage and uses only mesh-specific data, it is best suited to be precalculated to lessen the start up time of the analysis and visualization of the scene.

3.1.6. Fourier Transform

The discrete Fourier transform (DFT) is a useful resource to analyze the fluctuation of attributes of a series. I generate this series by picking a starting polygon p_{start} and add every remaining polygon p_i by order of distance $\overline{p_{start}\, p_i}$. The distance $\overline{p_i\, p_j}$ itself is defined by the length of the path from p_i to p_j by passing only through direct neighbors at a time, which in turn are reached only by going from the centroid of the polygon to the center of the common edge to the centroid of the neighboring polygon. Two examples for such a pathway can be seen in Fig. 3.3. A more detailed examination of the example can be found in chapter 4.2. I define p_{start} as the polygon with the highest average distance to every other polygon. This implies that there exists no polygon p_x without neighbors. Any such polygon p_x is removed from this analysis beforehand.

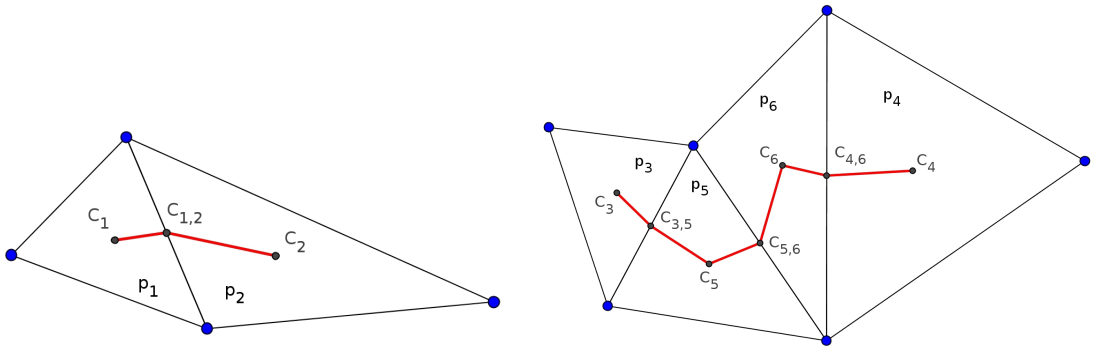


Figure 3.3.: Distances of polygons on the surface of meshes

Now I apply the DFT to the attributes of the thus generated one-dimensional series of polygons. As the imaginary part of the result is an odd function around $c = \lceil \frac{n-1}{2} \rceil + 1$ and as I am taking only the absolute values of the transformation into consideration, it suffices to limit the analysis to $X_k \in [X_0; X_c]$ with X_k being the amount of $\frac{2\pi k}{n}$ in the series. The metric consists of two aspects: One factor favors the magnitude of high frequencies by weighting each alternating component with its respective frequency. W normalizes the sum of all these weighted components by dividing by the sum of their magnitudes and the number of elements c:

$$W = \frac{\sum_{i=1}^{c} |X_i| \cdot i}{\sum_{i=1}^{c} |X_i| \cdot c}.$$

The second factor is the comparison of the strongest alternating component to the direct component. The combination of both factors thus results in

$$R_{ft} = \frac{\max_{1 \leq i \leq c} |X_i|}{|X_0|} \cdot W = \frac{\max_{1 \leq i \leq c} |X_i|}{|X_0|} \cdot \frac{\sum_{i=1}^{c} |X_i| \cdot i}{\sum_{i=1}^{c} |X_i| \cdot c} \in [0; 1].$$

In case the mesh contains only triangles of equal size, the Fourier transform of the resulting function would only have one non-zero spectral component: X_0. Thus the sum A of the magnitudes of all alternating frequencies would amount to 0 as well. In this case R_{ft} simply returns 0 to signify the most even distribution of the attribute in the mesh. This serves to avoid the division by zero during the calculation of W. The whole calculation, however, is based on the presumption that the absolute value of all spectral components $|X_k|$ are elements of $\mathbb{R}_0^+$. This can be accounted for by using an input signal containing only positive numbers or zero as values. Surface area values satisfy this condition. If the input signal does not meet this requirement, the resulting rating would not be element of [0;1].
R_{ft} is another metric with high runtime complexity. As it does not require data other than the analyzed mesh's geometrical data, it is calculated prior to the execution of the Scene Analyzer as well.

3.1.7. Rendering Time

Lastly, measuring each objects' rendI rate each mesh by its distance to the user, relative to largest distance $\Delta_{max} = \max_{m \in M} \Delta_m$ and smallest distance $\Delta_{min} = \min_{m \in M} \Delta_m$ within the scene. I use linear interpolation to calculate the individual distance ratings. This results inering time supplies an empirical rating of object complexity. Every object is rendered multiple times, each time with a different rotation applied to the mesh. The resulting measured rendering times t_i are aggregated and used to calculate a singular representative value

$$rt_m = \frac{\sum_{i \in I} t_{m,i}}{|I|}.$$

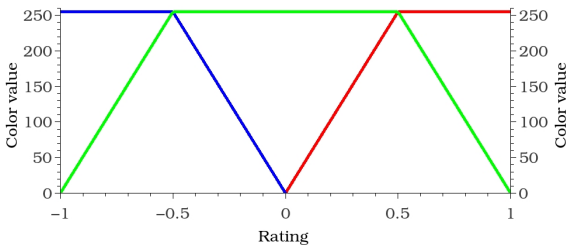
I rate each mesh by its rendering time, relative to largest rendering time $rt_{max} = \max_{m \in M} rt_m$ and smallest rendering time $rt_{min} = \min_{m \in M} rt_m$ within the scene. I use linear interpolation to calculate the individual rendering time ratings. This results in

$$R_{rt}(m) = \frac{rt_m - rt_{min}}{rt_{max} - rt_{min}} \in [0; 1].$$

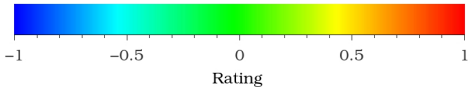
In case of each mesh of the scene having exactly equal rt_m, each is rated with a medium value of 0.5.
Since R_{rt} uses the rendering times rt_m of all meshes used during the execution of the Scene Analyzer, the calculation is split into two parts. The empirical measuring of each mesh's rendering time is sourced out to the precalculation. The measured data are then used to determine the actual relative rendering time rating during the runtime of the Scene Analyzer.

3.2. Visualization

In order to display the individual ratings of 3D objects, the respective meshes are dyed in different colors, thereby allowing the user to easily compare several objects within a single scene.
All previously listed ratings range from 0 to 1, with the exception of *relative density* which ranges from -1 to 1. With 0 representing simple, smooth or comparatively average meshes and 1 signifying complex or irregular meshes, as well as outliers, I choose a spectrum stretching from green for low ratings via lime, yellow and orange to red for higher ratings. For ratings encompassing negative values, which, for instance, can be used to visualize outliers of the lower extreme (with a rating of -1) differently than outliers of the higher extreme (with a rating of 1), I mirror the approach utilizing blue to represent low negative ratings (see Figure 3.4).



(a) RGB values



(b) Spectrum

Figure 3.4.: Assigned colors as a function of Rating

The hereby calculated RGB values are then applied to each mesh. If the mesh possesses original colors due to materials or textures, I combine the luminance of these hues with the mesh's ranking color in equal measure. I determine the luminance Y by weighting and combining the individual RGB values as in [Rec95]:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B.$$

This aims to ease identification of similar 3D objects having similar meshes, but different characteristics in terms of original color. The aggregation of multiple metrics into a single value occurs by multiplication of each rating $R_i(m)$ with its respective user-determined weight w_i. These weighted ratings are then totalled and normalized by the sum of all weights:

$$R(m) = \frac{1}{\sum_{i \in I} w_i} \cdot \sum_{i \in I} R_i(m) \cdot w_i, \quad I = \{pppg, pgv, rd, d, ccm, dft, rt\}.$$

In case metrics with differing value sets would be combined, the ratings are mapped on a common value set beforehand. The aggregation of R_{rd} and any of the other metrics would therefore require the use of $|R_{rd}|$ instead. Thereby each used metric would deliver results within [0;1].
To obviate difficulties in recognizing small differences in assigned colors, which would result from the scene having little variation in mesh ratings, I provide the option to normalize and scale the rating values. This means that the lowest and the highest rating present within the scene for each metric is set as to the respectively lowest and highest rating possible for that metric. The rating values of the remaining meshes is then linearly interpolated between these bounds:

$$R_{rel}(m) = \frac{1}{\sum_{i \in I} w_i} \cdot \sum_{i \in I} \frac{R_i(m) - \min_{m' \in M} R_i(m')}{\max_{m' \in M} R_i(m') - \min_{m' \in M} R_i(m')} \cdot w_i,$$

$$I = \{pppg, pgv, rd, d, ccm, dft, rt\}.$$

These relative ratings $R_{rel}(m)$ are then used for the visualization instead of the absolute ratings $R(m)$. The values of $R_{rt}(m)$ and $R_d(m)$, however, remain unchanged by the relativization, as these are relative metrics already. The mapping of ratings in a way that employs the whole range of the value set allows me to make use of the whole color spectrum corresponding to said value set. This makes it easier to recognize the meshes' individual rating colors and to discern differences which would otherwise be difficult to spot. In case that the rating values of a metric are identical, which would result in division by zero during the linear interpolation, the ratings are not mapped onto their whole range of possible values and are instead left unchanged.

4. Implementation

In this chapter I explain my approach to implementing the previously introduced concepts, the problems I encountered and their solutions. I split the program in two applications: One, the *Initializer*, calculates the mesh-specific data which is invariant to the actual scene composition. The other application, the *Scene Analyzer* generates the remaining runtime exclusive data and handles the visualization of the results. In the actual implementation the 3D Input File (see Figure 4.1) supplied by the user is decoded by the open source library ASSIMP [Tea16] and converted to a standardized representation of the geometrical data. ASSIMP is capable of parsing multiple different file formats, which are listed in the library's documentation. Additionally the user may provide parameters to the Initializer: A threshold angle α which dictates the definition of co-planarity during the generation of Planar Groups, a threshold size DIM which limits the size of the Co-occurrence Matrix in order to reduce running time and storage space and an amount of steps which represents the number of sample renderings during a full revolution of 360° around a single axis. The total number of measured renderings thus amounts to $steps^2$ different rotations as object is rotated around two axes. These data allow the calculation of R_{pppg}, R_{pgv}, R_{ccm} and the average rendering time. The final part of the Initializer, the calculation of the Fourier transform, is handled by the Fastest Fourier Transform in the West library [FJ16], which is provided by the Massachusetts Institute of Technology where it was developed by Matteo Frigo and Steven G. Johnson. The results of the Fourier transform are then used to calculate the R_{fft} of the mesh, which, along with the previously mentioned R_{pppg}, R_{pgv}, R_{ccm} and the average rendering time, are then written to a binary Scene Analysis File.

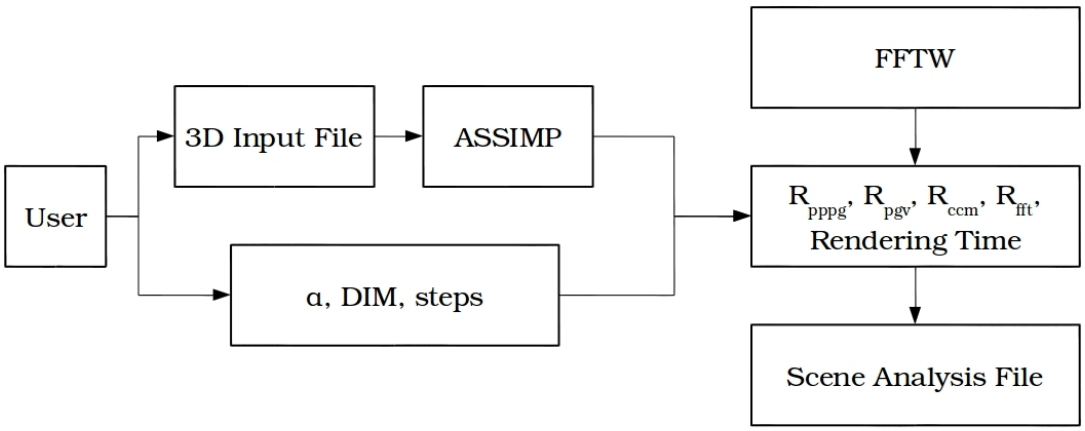


Figure 4.1.: Implementation Initializer

During the execution of the Scene Analyzer the user again provides one or multiple 3D Input Files detailing the scene to be analyzed, which is parsed with the help of ASSIMP.

The filepaths used to describe the location of the input files are also used to locate the respective Scene Analysis Files, which are stored in locations relative to the files containing the meshes. The average rendering times of each mesh within the scene are then gathered and used to determine each mesh's R_{rt}. Similarly, the calculation of R_{rd} and R_d takes place during runtime, since these metrics rely on runtime dependant values as well: composition of the scene and user position. With all seven ratings present, the user then assigns a weight w_i to each rating R_i, which is used to compute the weighted aggregated rating of each mesh. After that the final ratings are mapped on corresponding RGB values and assigned to each respective mesh, thereby completing the visualization of the scene analysis. During runtime the user is able to interact with the scene by using the arrow, ";" and "." keys to navigate the scene. Presses of "+" and "-" result in the enlargement and shrinking of the scene, respectively. "F1" through "F7" are used to de- or reactivate visualizations of particular metrics in order to explore different aspects of the scene. As an additional analysis tool "F10" can be pressed to change to the visualization of each mesh's Planar Groups and back. "F11" switches the rating mode between "Absolute" and "Relative". Pressing "p" writes all current rating values to file for examination of the exact values, whereas "F12" serves as a general helper key, which details all key functions and the current rating weights in the terminal output. Finally, the application can be exited by the use of "ESC".

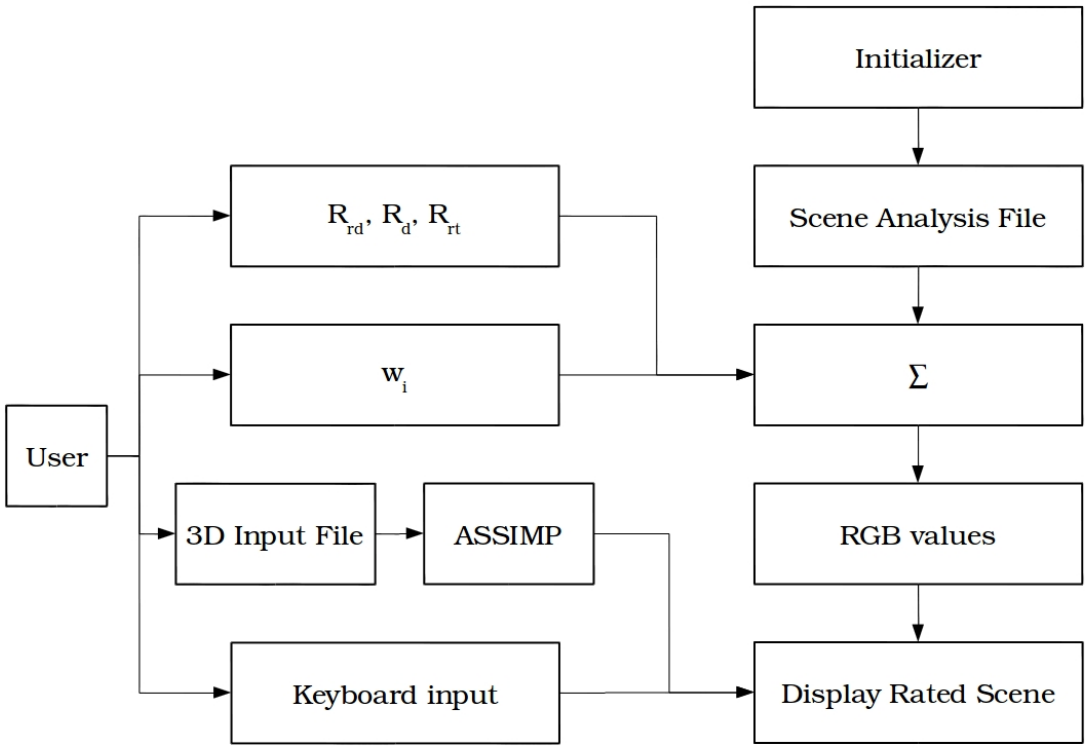


Figure 4.2.: Implementation Scene Analyzer

4.1. Planar Grouping

Planar Grouping is achieved by iterating over all polygons and their respective neighbors (see Algorithm 1). If their normals diverge by less than the required threshold (line 3), then they need to be assigned to the same *Planar Group*: If both polygons already belong to different groups, their respective groups are joined together (line 4f.); if only the currently tested neighbor is part of a group, then the previously unassigned polygon is added to the neighbor's Planar Group (line 6f.). Otherwise a new Planar Group containing only the polygon is created (line 11), even if a groupless neighbor fulfills the grouping condition. The neighbor will lateron be added to the polygon's group over the course of the encapsulating iteration over all polygons.

Algorithm 1 Planar Grouping

```
 1: for all p1 in Polygons do
 2:     for all p2 in p1.neighbors do
 3:         if angle(p1.normal,p2.normal) ≤ threshold then
 4:             if p1.group ≠ NULL && p2.group ≠ NULL && p1.group ≠ p2.group then
 5:                 join(p1.group,p2.group)
 6:             else if p1.group = NULL && p2.group ≠ NULL then
 7:                 p2.group.add(p1)
 8:             end if
 9:         end if
10:     end for
11:     if p1.group == NULL then
12:         p1.group = new planarGroup()
13:     end if
14: end for
```

4.2. Distance of Polygons on the Surface of a 3D Mesh

Since my frequency analsis requires a one-dimensional ordered list of the polygons of the mesh as input signal, I needbegin a sorting condition which fulfills the following two requirements: First the resulting list needs to be defined well enough to create an unambiguous and reproducable signal. This leads to consistent deterministic results over repeated executions. Secondly the sorting condition needs to be based on the spatial relationships of the polygons, as the goal of the frequency analysis is to highlight correlations between the polygons' attributes and their proximity.
Therefore, I choose to sort the polygons by distance. Comparing two-dimensional distances in three-dimensional space requires a way of dealing with the possible change of plane upon traversing the shared edge of two polygons. For this the distance between two neighboring polygons p_1 and p_2 is defined as the distance of the centers C_1, C_2 of both polygons to the center of their common edge $C_{1,2}$. The distance of two polygons p_3 and p_4 without common edge p_3, p_4 is recursively defined as the shortest of the paths which result from traversing adjacent polygons. Therefore the distance of p_3 and p_4 is the sum of the distances of p_3 and p_5, p_5 and p_6, and lastly p_6 and p_4 (see Figure 4.3).

To find the respectively shortest paths in this large, but sparse graph, I implement Dijkstra's solution [Dij59] to this All Pairs Shortest Paths Problem.

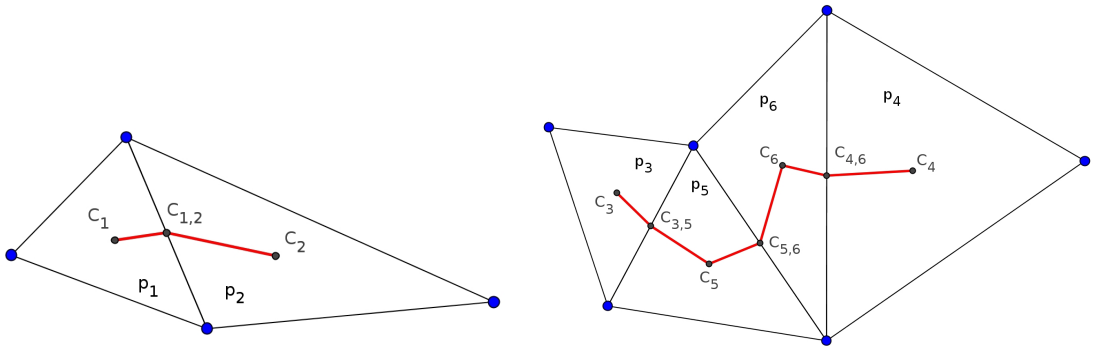


Figure 4.3.: Distances of polygons on the surface of meshes

4.3. Co-occurrence of Surface Areas of Polygons

For the implementation of the analysis of co-occurrences of surface areas of polygons two hardware-based hurdles need to be handled. The first and more obvious problem is the required memory space of the Co-occurrence Matrix M. At n^2 dimensions the matrix corresponding to a single mesh of 100 000 triangles would necessitate 20 GB memory if each individual entry was saved as 16 bit unsigned short int. With scenes containing multiple meshes of similar or even bigger polygon counts, such memory requirements are decidedly too large. Converting the matrix to triangular form by mapping $M'[i,j] = M[i,j] + M[j,i]$ would halve these requirements, but not mitigate the memory requirements in a sufficient way. The second problem results from the inherent discretization of values. As vertex coordinates are saved with a finite precision, a theoretically completely regular mesh, e.g. a geosphere, might contain triangles of diminutively diverging area sizes upon digitalization.
I solved both problems by modifying the construction of the initial sorted list S of the triangle sizes which are present within the mesh. Since a triangle's size is only added to S if it is unique, i.e. not already an element of S, a pairwise comparison of all triangles is necessary in any case. I expand the relational operator to treat triangles as equal sized even if their areas differ by up to the valuebegin of the machine epsilon of the used data type; 1.19209e-07 when using float. This does away with the second problem of same sized triangles not being recognized due to technical limitations. I approach the problem of too much memory being required by setting a hard limit DIM on the number of elements in S and thereby the size of M. For my implementation I set $DIM = 20\,000$, which, along with each entry being a double of 8 bytes, leads to a maximum of 3.2 GB memory per Co-occurrence Matrix. The eventual value of DIM, though, can be adjusted to fit individual requirements and hardware conditions. A maximum threshold of the number of elements, however, requires a non-arbitrary filtering to keep the resulting error as low as possible and correspondingly the analysis as meaningful as possible. The Planar Grouping Algorithm
To this end I group adjacent elements of the unreduced sorted list S and calculate the mean average of these groups (see Algorithm 2). I use this mean value, in turn, to represent all elements of these groups. Each representative mean average therefore reduces the amount of elements in S by the size of the group it represents minus one. To guarantee that the thus

aggregated error (lines 11, 14) is the smallest possible error, each turn one element is added to a group (line 20) only if it generates the least possible error during that iteration (line 16f.). To facilitate grouping and retain the values of the elements of each group, I initialize the algorithm with a new list of lists *S'* (line 1) containing all elements of S as singletons (line 2f.) and keeping the previously generated ascending order.

Algorithm 2 Limiting the size of the co-occurrence matrix

```
 1: T = new List(List)
 2: for all s in S do
 3:     T.add(new List(s))
 4: end for
 5: while sizeof(T) > DIM do
 6:     to_join1 = i, to_join2 = i+1, error = ∞
 7:     for i=0 to sizeof(T) do
 8:         m = mean(T[i],T[i+1])
 9:         e = 0
10:         for all t_1 in T[i] do
11:             e += |t_1 − mean|
12:         end for
13:         for all t_2 in T[i+1] do
14:             e += |t_2 − mean|
15:         end for
16:         if e < error then
17:             to_join1 = i, to_join2 = i+1, error = e
18:         end if
19:     end for
20:     join(to_join1, to_join2)
21: end while
```

5. Results and Discussion

In this chapter I examine the actual ratings and renderings which resulted from the automated analysis of 3D scene files. For each scene I present a flat rendering and the corresponding rated and dyed rendering. The flat rendering serves as a presentation of the raw scene data as seen by a potential user. The rated rendering, on the other hand, is automatically created by the Scene Analyzer by using the geometrical data of the 3D meshes. The thus generated ratings are then assigned to each mesh as color value in the successive rendering of the scene.
For each scene I compare the flat rendering and its underlying data to the rating assigned. On the one hand I investigate whether each rating suitably represents the attributes and structure of the corresponding mesh. In that case, I present the causes leading to the calculated values; which characteristics and features of the mesh warrant its rating. On the other hand, I explore the unrated scene and the raw data to reveal unbefitting ratings, be it by flaw of design or otherwise inability of the metric to pick up features it is intended to pick up. I discuss these cases of considerably improper ratings values, the reason for their impropriety, the causes of these metric shortcomings as well as possible solutions to these algorithmic deficiencies.

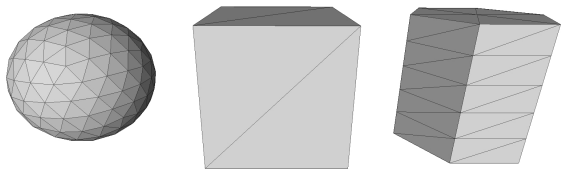
5.1. Exemplary Application of Mesh Rating Metrics

In this section I give seven example scenes, one for each metric, to elaborate representative cases of rating values and their causes. Each case is chosen to depict meshes of distinctly differing rating values in order to explore most of the value set of each metric. The scenes were designed to result in at least one low, one high and one medium rating. Therefore, not all of the meshes presented in this section are neccessarily representative of real life applications. Instead, the scenes serve to comprehensibly illustrate meshes of different rating values as well as the implications of each value for the mesh attributes and structures. Additionally, for each scene I provide a table containing all data relevant to the calculation of the respective metric.

5.1.1. Polygons per Planar Group of Geosphere, Cube and Hexagonal Prism

Fig. 5.1 shows two renderings of three specific 3D objects: a geosphere, a cube and a regular hexagonal prism. The geosphere consists of 320 triangles, each angled slightly different than its neighbors. The cube is built from 6 squares of 2 co-planar triangles each. Lastly, the regular hexagonal prism contains 5 layers, which leads to a polygon count of 72 triangles, 6 per top and bottom and 10 per side plane. I apply *Planar Grouping* with $\alpha = 0°$. This leads to the ratings of $R_{pppg}(geosphere) = \frac{0}{319}$, $R_{pppg}(cube) = \frac{6}{11}$ and $R_{pppg}(prism) = \frac{64}{71}$ (cf. Table 5.1). The geosphere receives the best possible rating, due to all *Planar Groups* being singletons, as no triangle within the mesh has the same normal as at least one of its neighbors. The cube receives a mediocre rating, because each of its triangles shares a

plane with one of its neighbors. The prism is rated worst out of the three meshes, as it uses 72 triangles to construct only 8 different planes. The same mesh could be created less redundantly by using less triangles per side.
The rating of the cube can be misleading, as it can not be simplified any further. The problem lies within the choice of the base primitives of the mesh. The cube is built from tetragons, whereas the metric operates on triangles. Generally a mesh built from a specific kind of n-polygons with $n > 3$ would volitionally contain many $n - 2$ co-planar triangles. Given prior knowledge of the components of the mesh the algorithm can be tweaked to group its n-polygons instead of its triangles.



(a) Flat lines rendering

(b) Rating rendering

Figure 5.1.: R_{pppg} evaluation of a scene containing a geosphere, a cube and a prism

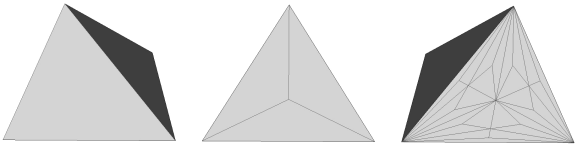
Mesh	n	$\|PG\|$	R_{pppg}
Geosphere	320	320	0
Cube	12	6	$0.\overline{54}$
Prism	72	8	0.901408451

Table 5.1.: Data relevant to the R_{pppg} evaluation of a scene containing a geosphere, a cube and a prism

5.1.2. Planar Group Variation of Tetrahedrons

Fig. 5.2 displays three tetrahedrons with differing polygon counts on the front facing side: The left tetrahedron has the most simple mesh of the three, featuring one triangle per side. The center tetrahedron is of increased complexity, as its front facing side is created by combining three triangles. Lastly, the tetrahedron to the right features a side containing 27 triangles, while the remaining three sides consist of singular triangles.
Once again I perform *Planar Grouping* on these three objects with $\alpha = 0°$, which results in a rating of $R_{pgv}(left) = \frac{0}{4}$, $R_{pgv}(center) = \frac{3}{6}$ and $R_{pgv}(right) = \frac{27}{30}$ (see Table 5.2). Since the left tetrahedron uses no redundant triangles to create its four sides, it receives the best possible rating. Next, the centered tetrahedron uses half of its total polygons to create one side, while the remainder is distributed over the other 3 sides. Consequently, it receives a

mediocre rating. Lastly, the tetrahedron to the right expends 27 of its 30 triangles on its front facing side, while the remaining three sides use but one triangle each. Therefore this 3D object receives the worst rating out of the three objects compared.



(a) Flat lines rendering

(b) Rating rendering

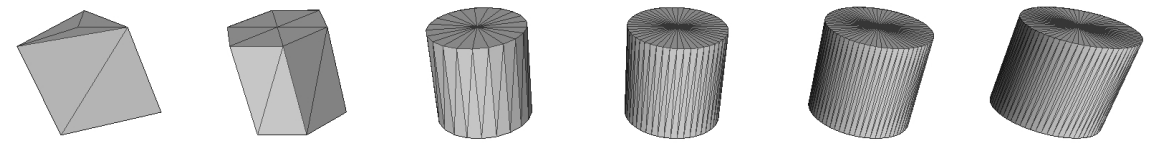
Figure 5.2.: R_{pgv} evaluation of a scene containing three tetrahedrons

Mesh	n	$\|pg\| \in PG$	R_{pgv}
Left	4	1,1,1,1	0
Center	6	1,1,1,3	0.5
Right	30	1,1,1,27	0.9

Table 5.2.: Data relevant to the R_{pgv} evaluation of a scene containing three tetrahedrons

5.1.3. Relative Density of Prisms

This example utilizes six prisms of equal height and iteratively bigger base areas (see Fig. 5.3). The base areas are approximations of the same disc and thereby share the same radius and differ only in the number of vertices. Consequently the six models are of comparable volume, but differ drastically in the number of polygons. Therefore, their respective densities of polygons per unit of area vary as well.
Statistical analysis of this scene delivers a mean density of $\mu = 0.29604$ and a standard deviation of $\sigma = 0.215309$ (see Table 5.3). The resulting categorization of the meshes shows that both leftmost prisms lie within interval I, due to their respective density falling below $\mu - \sigma = 0.080371$ (see Table 5.4). Hence both are dyed in shades of blue. Mirroring these properties, both righmost prisms exceed $\mu + \sigma = 0.511349$, which leads to assigning them to interval III and a coloration in shades of red. Finally, both prisms in center lie between $\mu - \sigma$ and $\mu + \sigma$ and thereby meet the requirements for interval II. As correspondingly meshes of average density, both are dyed in shades of green.



(a) Flat lines rendering

(b) Rating rendering

Figure 5.3.: R_{rd} evaluation of a scene containing six prisms

μ	σ	d_{min}	d_{max}
0.29604	0.215309	0.0369505	0.56065

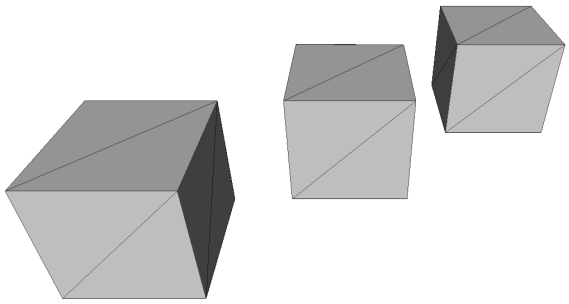
Table 5.3.: Mean density, standard deviation, minimum density and maximum denstiy of a scene containing six prisms

Mesh	d_m	Interval	R_{rd}
Leftmost	0.0369505	I	-1
Second from the left	0.0558266	I	-0.784423
Third from the left	0.179589	II	-0.270428
Third from the right	0.40802	II	0.260045
Second from the right	0.535204	III	0.741934
Rightmost	0.56065	III	1

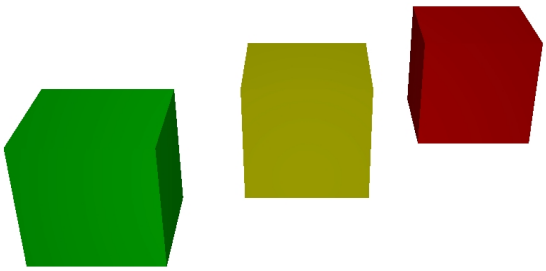
Table 5.4.: Data relevant to the R_{rd} evaluation of a scene containing six prisms

5.1.4. Distance of Cubes

To demonstrate the *relative distance rating* I use three equal cubes arranged within a scene at differing z-coordinates (see Fig. 5.4). Since I approximate the distance of meshes to the user by resorting to their respective bounding boxes, the meshes and their approximations are identical.
At the presented position and orientation of the user's viewpoint the distance of the meshes amount to, from left to right, 1.66838, 1.73882 and 1.81755 (see Table 5.5). Consequently the cube to the left, being the closest to the user, receives the lowest possible rating of 0. The cube on the right is treated analogously and rated with the highest possible value of 1 due to it having the furthest distance. The cube in the center features a distance which is virtually the average of both extreme distances of meshes within the scene. Therefore it is rated close to the average of 0.5 at 0.472204.



(a) Flat lines rendering



(b) Rating rendering

Figure 5.4.: R_d evaluation of a scene containing three cubes

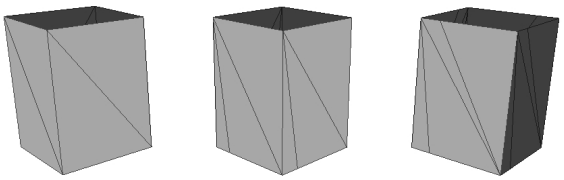
Mesh	Δ_m	R_d
Left	1.66838	0
Center	1.73882	0.472204
Right	1.81755	1

Table 5.5.: Data relevant to the R_d evaluation of a scene containing three cubes

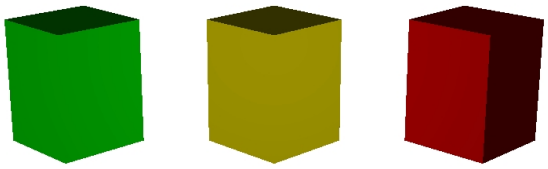
5.1.5. Co-Occurrence of Triangle Sizes in Tubes

To reduce the complexity of co-occurrences and thereby facilitate comprehensibility, I use four-sided tubes instead of prisms in this example. As tubes lack both tops and bottoms, each triangle has but two neighbors instead of three. Each tube features four equal sides, but the composition of these sides differs between the tubes (see Fig. 5.5).
The tube on the left is built from eight identical triangles of 35.35553 area units each. Hence, as the area sizes on the tube's surface don't change, it is considered entirely smooth. Consequently, the Co-occurrence Matrix M has only one element, which represents the number of times a triangle of 35.35553 area units borders on a triangle of the same size. As all of M's entries lie on its diagonal, the resulting contrast and rating amounts to 0 (see Table 5.6). The tube in the center, on the other hand, contains triangles of three different sizes, which results from the subdivision of every other triangle into a small triangle and a medium triangle of 7.6778 and 27.6776 area units, respectively. This leads to a fluctuation of triangle sizes on the tube's surface. With each of the three kinds of triangle sizes (small, medium and large) having only respectively different sizes of triangles as neighbors (small borders exclusively on medium and large, medium borders exclusively on small and large,

large borders exclusively on small and medium), the main diagonal of M contains only 0s. But since M's entries are not limited to its top right and bottom left corners, which results from the medium sized triangle working as a buffer by keeping small triangles and large triangles from touching 50% of the time, the contrast is lessened to 0.53303. The tube on the right applies the process of subdivision of each large triangle into a small and a medium sized triangle. With only two types of triangles present, the Co-occurrence Matrix M is a 2x2 matrix. As each small triangle borders exclusively on medium triangles and vice versa, M's only non-zero entries are located in its top right and bottom left corners. This signifies the maximum contrast and the highest frequency of area size fluctuation possible. Accordingly its rating equals 1.



(a) Flat lines rendering



(b) Rating rendering

Figure 5.5.: R_{ccm} evaluation of a scene containing three tubes

Mesh	S	M	R_{ccm}
Left	$\{35.3553\}$	$\begin{pmatrix}8\end{pmatrix}$	0
Center	$\{7.6778, 27.6776, 35.3553\}$	$\begin{pmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{pmatrix}$	0.53303
Right	$\{7.6778, 27.6776\}$	$\begin{pmatrix}0 & 8\\ 8 & 0\end{pmatrix}$	1

Table 5.6.: Data relevant to the R_{ccm} evaluation of a scene containing three tubes

5.1.6. Fourier Transform

For the illustration of the Fourier transform metric, I opt against the use of real 3D models as example. Instead I present four simple sequences of numbers S, which are easy to comprehensibly analyze, in Table 5.7. 3D models, which would generate these series through the ordering of the triangles as detailled in chapter 4.2 and by using the triangles' area sizes as signal, would be special cases and of degenerated shape. Therefore, I exclude 3D objects to keep this exemplary application simple and less obfuscating. The first exemplary series of triangle sizes is $\{100, 100, 100, 100, 100, 100\}$. As it contains only constant

values, the corresponding DFT delivers n = 6 components: a constant component X_0 of 600 and n - 1 = 5 frequency components of 0 each. I find $c = \lceil \frac{6-1}{2} \rceil = 3$ and the sum of all frequency components $[X_1; X_c]$ therefore amounting to A = 0. This means that the weighted sum of the frequency components is set to W = 0 as well. Lastly, R_{ft} amounts to $\frac{0}{600} * 0 = 0$. This rating fits a perfectly even and non-fluctuating series. The second series, $\{200, 100, 100, 100, 100, 100\}$, contains a single peak at twice the amplitude as the rest of the series. Accordingly, the DFT of the second sequence results in $\{700, 100, 100, 100, 100, 100\}$. Again, c is 3, which means that A = 100 + 100 + 100 = 300. As W then amounts to 2, the calculation of R_{ft} of the second series results in $\frac{100}{700} * \frac{2}{3} = 0.095238095$. This rating is still fairly low, which represents the mostly constant series and the fact that its lone peak is only twice as intense as the other five elements. As third example I choose the sequence $\{1000, 100, 100, 100\}$. This series contains a stronger peak than the previous series, which also reoccurs more often due to a period of n = 4. Correspondingly, c amounts to 4, which leads to A = 1800. With W = 1.5, the results of R_{ft} is $\frac{900}{1200} * \frac{1.5}{2} = 0.5625$. This medium rating value corresponds to a deep yellow coloration. It is significantly higher than the previous rating due to the higher frequency and the harsher amplitude difference at the peak. Lastly, I calculate the Fourier transform ratings of the sequence $\{100, 1\}$, which represents the maximum possible frequency and a strong variation in signal values. The corresponding DFT delivers $\{101, 99\}$. Since c amounts to 1, A is equal to the lone frequency component: 99. Thus, W amounts to 1 as well. Hence, R_{ft} delivers a rating of $\frac{99}{101} * \frac{1}{1} = 0.98019802$. This very high rating, which would lead to a deep red color, is the result of the extremely high frequency and variation present.

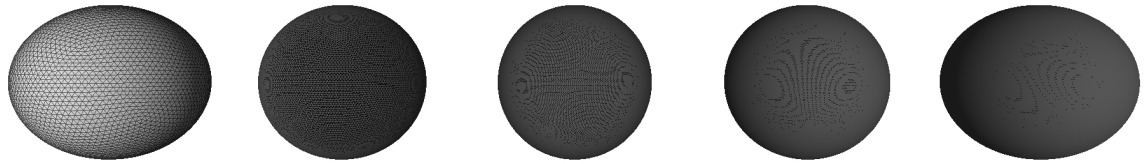
S	$\lvert DFT(S) \rvert$	c	A	W	R_{ft}
$\{100, 100, 100, 100, 100, 100\}$	$\{600, 0, 0, 0, 0, 0\}$	3	0	0	0
$\{200, 100, 100, 100, 100, 100\}$	$\{700, 100, 100, 100, 100, 100\}$	3	300	2	0.095238095
$\{1000, 100, 100, 100\}$	$\{1200, 900, 900, 900\}$	2	1800	1.5	0.5625
$\{100, 1\}$	$\{101, 99\}$	1	99	1	0.98019802

Table 5.7.: Exemplary sequences of triangle sizes and their R_{ft} values

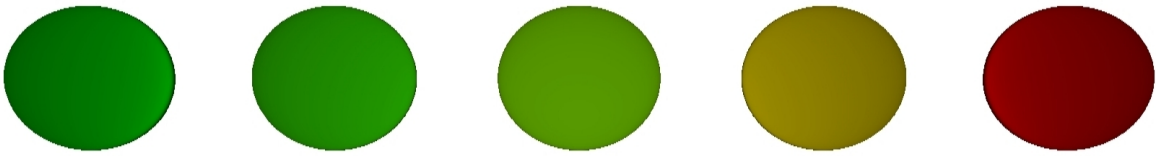
5.1.7. Average Rendering Time of Geospheres

Fig. 5.6 shows five geospheres of equal radius. These were constructed utilizing an increasing amount of polygons, ranging from 8 000 polygons used on the leftmost geosphere to 200 000 polygons used on the rightmost geosphere. Each rendering time rt represents the empricial mean rendering time of 10 000 individual renderings of each singular mesh using a Dell Latitude E6510's Nvidia Quadro NVS 3100M with 4GB of RAM and a Intel Core i7 620M processor.
Due on the linear correlation of rendering time and object complexity, i.e. polygon count, the mesh with the least amount of polygons features the shortest rendering time of the scene. The leftmost geosphere consisting of 8 000 polygons is therefore the least computationally expensive object at 0.21346 ms and hence rated with 0 (see Table 5.8). Analogously, the rightmost geosphere, which is built from 200 000 polygons, is the most complex object of the scene. Consequently, it requires the longest rendering time with 5.13932 ms, which results in a rating of 1. The geosphere to the left consisting of 128 000 polygons is approximately

halfway between the leftmost geosphere's and the rightmost geosphere's complexity, which is reflected in its rendering time and resulting rating of 0.50706. Both the geosphere in the center and the geosphere second from the left fall into line with rendering times at roughly one quarter and one tenth of the way between the leftmost geosphere's and the rightmost geosphere's rendering time. This is reflected in their respective ratings of 0.26822 and 0.10151.



(a) Flat lines rendering



(b) Rating rendering

Figure 5.6.: R_{rt} evaluation of a scene containing five geospheres

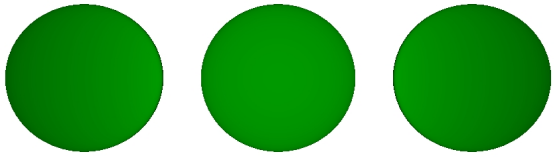
Mesh	n	rt in ms	R_{rt}
Leftmost	8000	0.21346	0
Second from the left	32000	0.73192	0.10151
Center	72000	1.53467	0.26822
Second from the right	128000	2.71117	0.50706
Rightmost	200000	5.13932	1

Table 5.8.: Data relevant to the R_{rt} evaluation of a scene containing five geospheres

5.1.8. Relative Rating of Geospheres

Fig. 5.7 (a) presents the virtually indistinguishable visualizations of R_{pppg} with $\alpha = 0°$ of three geospheres. These geospheres differ slightly in their number of polygons n and amount of planar groups $|PG|$. The geosphere on the left features 8 000 polygons and 8 000 planar groups, resulting in a perfect R_{pppg} of 0 (see Table 5.9. The geosphere in the center uses 8 006 polygons in its 8 002 planar groups. This small redundancy leads to a R_{pppg} of 0.0005. Lastly, the geosphere on the right contains 8 010 polygons in 8 005 planar groups. Its R_{pppg} therefore amounts to 0.00062. Even though the respective ratings are not equal, all meshes are assigned the same color value due to the differences in rating being miniscule. To display these small variations nonetheless, I utilized the relative variant of the metric. Thereby the values of R_{pppg}, which range from 0 to 0.00062 in the present example, are mapped onto [0;1].

As can be seen in Fig. 5.7 (b), the visualization of the new ratings of 0, 0.8004 and 1, respectively, permits the detection of the present differences and the succeding examination as to what constitutes the differences.



(a) Absolute rating

(b) Relative rating

Figure 5.7.: R_{pppg} evaluation of a scene containing three geospheres

Mesh	n	$\|PG\|$	R_{pppg}	R_{rel_pppg}
Left	8000	8000	0	0
Center	8006	8002	0.0005	0.8004
Right	8010	8005	0.00062	1

Table 5.9.: Data relevant to the R_{pppg} evaluation of a scene containing three geospheres

5.2. Scene

To further test and evaluate the metrics, I use the scene presented in Fig. 5.8. It employs 94 074 triangles in total to create a small room containing two couches with a cushion each, a table, upon which a controller has been placed, and a TV on a TV table (see Fig. 5.9). Both couches consist of a body of 1 430 polygons and four identical feet of 40 triangles each. The big couch is but an elongated version of the small couch as implied by their equal amount of triangles. The cushions lying on top of each couch are structurally the same as well. Each spends 40 612 triangles to create the desired surface. They differ only in a scaling factor, with the cushion on top of the big couch being slightly larger. The TV table consists of two boards, which contain 332 polygons each, and four feet of respectively 40 triangles. These furniture feet are essentially the same as those of the couches. They differ only in their height, since the feet of the TV table are taller, and their spatial orientation. The fourth and final piece of the furnishings is a television set situated on top of a TV stand. The feet of the TV stand are pairwise connected by an arc and modelled differently from the previously mentioned furniture feet. Each identical pair of legs of the TV stand consists of 256 polygons. Both boards of the TV stand are bundled together into the same mesh which amounts to 120 triangles. The TV itself consists of its display, which uses but 2 triangles, and its frame, which encapsulates the display with 360 triangles. At the top of

the backside of the TV frame are two TV buttons and their casing situated, which consist of 584 triangles and 36 triangles, respectively. An extension of the TV frame is attached to the bottom of the frontside of the TV frame. This bottom casing is divided into its front and its side meshes, which respectively amount to 22 polygons and 82 polygons. This casing houses the faux brand name of the TV which consists of 2 244 polygons. The rack of the TV stand, which uses 196 triangles, connects the TV to its base counting 252 triangles. Lastly, all of these pieces of furniture stand on top of the floor which contains 32 triangles. All of these mesh details and their exact corresponding rating values can be seen in Table 5.10 for further reference. The values of the R_d metric are omitted, as these vary during runtime based on the position of the user.
I forego the use of textures and materials for the design of the scene as my metrics don't use these data and the resulting visual clutter would impede the visual inspection of the individual triangles within the meshes. Additionally, I use but single metrics during the analysis to focus on the individual strengths and weaknesses of each metric. This means that I refrain from the aggregation of different metrics, even though combinations of, for example, R_{pppg} and R_{pgv} or R_{rd} and R_{rt} deliver promising results.

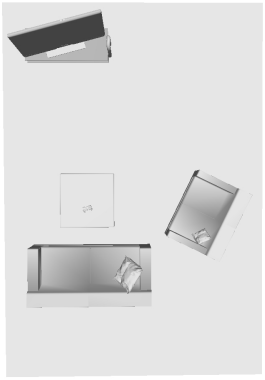


Figure 5.8.: Scene to be analyzed

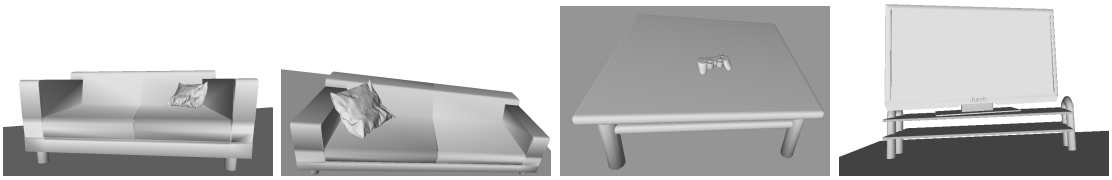


Figure 5.9.: Close up of the pieces of furniture: small couch, big couch, table, TV

5.2.1. Polygons per Planar Group

Inspection of the colorized scene, which can be seen in Fig. 5.10, which was rated by the R_{pppg} metric using $\alpha = 0°$, shows several prominent differences in the structure of the comprising meshes. First and foremost both the floor and the TV screen are dyed red. This is owed to the fact that both are strictly two-dimensional meshes using multiple triangles. Therefore,

all triangles of each mesh are part of the respectively same planar group. Next, the boards of the TV stand as well as the legs of the table and the couches feature a distinct shade of orange. As the legs of the furniture are modelled as prisms (approximating a perfect cylinder), they contain several co-planar triangles in their base and top areas. The boards of the TV stand, however, are of cuboid shape, which should result in a lower rating and thereby a more yellow color. Closer inspection of the mesh reveals a faulty attempt at rounding its edges and corners. This led to several triangles along each edge being co-planar instead of having slightly differing normals. Similar errors occured during the modelling of the second, bigger couch. Whereas the TV's frame and stand, the boards of the table and the small couch took on a shade of green yellow, the big couch's yellow color sets it apart from the rest. Conceptually, both couches are supposed to contain equal polygons and to differ only in width. Still, their R_{pppg} values are not equal. Highlighting each couch's planar group (see Fig. 5.13) reinforces the notion, as it shows that the big couch contains planar groups of larger cardinality than the planar groups of the small couch. Upon closer inspection of the meshes I once again found that the rounding of edges and corners was but partially succesful during the creation of the big couch, whereas the meshes of the small couch, the boards of the table and the TV's stand and frame feature noticeable differing orientations of the polygons along the edges. Peculiarly, the controller atop the table is colored almost the same as the table's boards. The controller has a rather uniform mesh aperture, which leads to the use of several small triangles to model planar sides (see Fig. 5.11). These sides could likewise be generated through the use of fewer, bigger triangles instead. Lastly, the smallest R_{pppg} values of the scene are awarded to the cushions and the legs of the TV stand. While small planar groups are present in the mesh of the cushions, they are dwarfed by the sheer number of polygons. The legs of the TV stand, as opposed to the legs of the couches and the table, were modelled in such a way that the number of triangles on their base areas is minimized, as seen in Fig. 5.12. Additionally, the equivalent problem at the top area is solved by replacing it with the curvature, which contains no co-planar triangles.

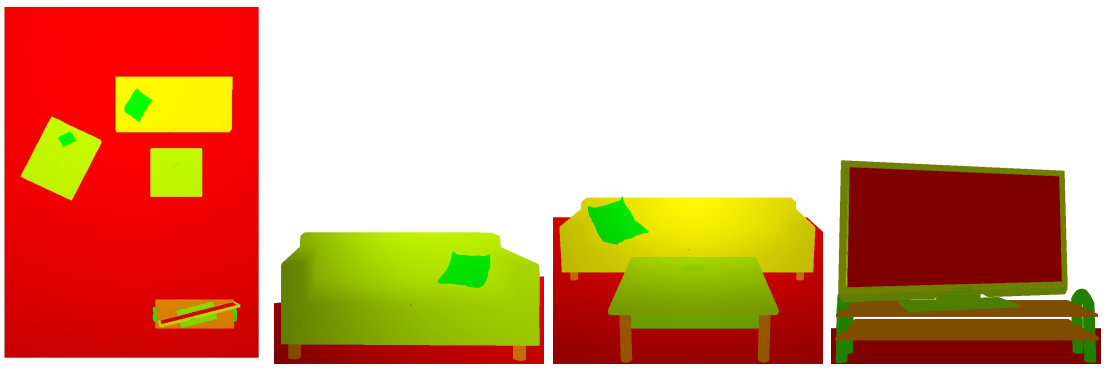


Figure 5.10.: R_{pppg} visualization of the scene. The cushions and the feet of the TV stand receive low ratings and are therefore colored green. The floor and the TV screen are dyed deep red due to their high R_{pppg} values.

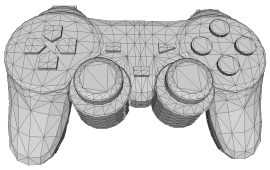


Figure 5.11.: Close up of the mesh of the controller

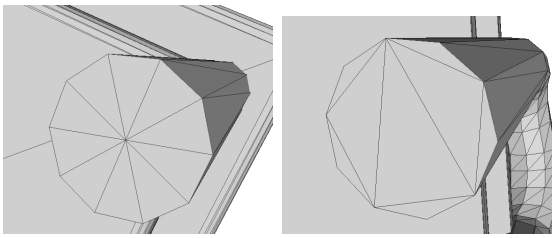


Figure 5.12.: Differences in the base areas of the furniture feet: Couch/Table and TV stand

Fig. 5.13 shows a heatmap of the Planar Groups of individual parts of the scene. The coloration uses the same previously presented color scheme: The elements of the numerically biggest Planar Groups receive a deep red dye, while the singleton Planar Groups are assigned deep green shades. The color of the remaining Planar Groups are calculated by linear interpolation. This more detailed visualization of Planar Groups helps to identify the concrete suboptimal regions of each mesh. This enables the user to more easily reduce the co-planarity or the polygon count of the mesh in question.

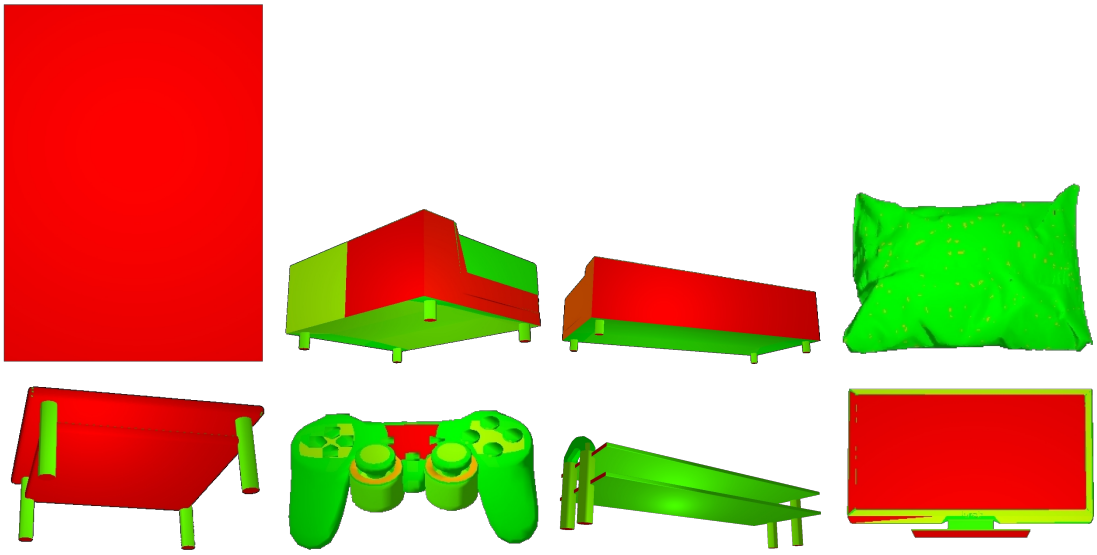


Figure 5.13.: Individual Highlighting of the respectively strongest Planar Groups of each part of the scene: Floor, small couch, big couch, cushion, table, controller, TV stand, TV set

A shortcoming of this kind of rating occurs in its application to strictly two-dimensional meshes. These cases require a minimum amount of co-planar triangles to generate certain shapes. Since all of these triangles are elements of the same planar group, R_{pppg} always returns 1 for those meshes. Therefore, R_{pppg} is not suited to differentiate between two-dimensional meshes with optimal use of triangles and two-dimensional meshes with redundant and inefficient use of triangles. Another limiting factor is the assumption that triangles are the basic polygon used to create the scene. The use of n-polygons with $n > 3$ during the modelling process as well as modelling of non-triangular areas lead to co-planarity of triangles, which may be unpreventable. The thus resulting high rating may not properly reflect the quality of the triangle usage. To solve that problem one would first need to pre-group triangles to generate the adequate n-polygons. These n-polygons could then serve as base primitives for the analysis. Determining the correct n-polygon, however, requires either an algorithm capable of extracting this parameter from the mesh or n being supplied by the user. Additionally, the user benefits from additional marking of the individual subregions causing the co-planarity, as these can make up an arbitrarily small part of an arbitrarily large mesh. A more detailed highlighting then provides a better guidance to the user in determining which regions need to be reworked.

5.2.2. Planar Group Variation

To further get a grasp of the co-planarity of triangles in the scene, I examine the visualized R_{pgv} of the scene. As before, I choose to define co-planarity as two neighboring triangles having the same normal; $\alpha = 0°$. Once again both, the floor and the TV screen, stick out due to the assigned deep red color. Both meshes are but simple planes. Therefore, every comprising triangle is part of the respectively same planar group. Correspondingly, both have received a high R_{pgv} value. Unlike R_{pppg}, R_{pgv} rates the couches, along with the cushions, the controller, the boards of the table, the TV stand's legs and the TV frame very low. This means that, even though the couches, the table boards and the TV frame ontain few planar groups compared to their polygon count, the cardinality of the individual planar groups is comparatively small. The other legs of the furniture and the boards of the TV stand, however, are dyed of a color with a bigger red value, which sets them apart from the otherwise mostly green pieces of furniture. This signifies that the smoothing error in the boards of the TV stand is more severe than the same error in the table boards and the couch, when contrasted with their respective polygon counts. Moreover, the arrangement of triangles to create the base and top areas of the legs of the couches and the table leads to relatively big planar groups, unlike the base areas of the legs of the TV stand. These feature a more efficient approach which helps to minimize the size of the resulting planar groups.

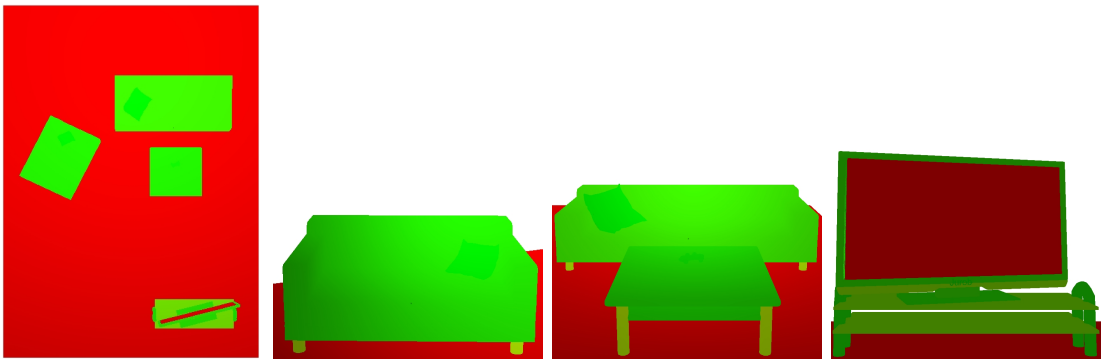


Figure 5.14.: R_{pgv} visualization of the scene. The TV screen and the floor noticeably stand out due to their high R_{pgv} values. The remaining pieces of furniture received low or very low ratings and are therefore colored green.

The shortcomings of this metric are essentially the same as the flaws of R_{pppg}, since both use the same data and analysis method to generate a rating. Once again the rating of two-dimensional meshes may not be meaningful, since *Planar Grouping* analyzes the co-planarity in three-dimensional meshes. Meshes which are created using rectangles or other polygons of a higher vertex count as primitives, instead of triangles, may be rated unfavorably as well. The previously mentioned pre-grouping of triangles into groups suitable to reflect the use of n-polygons in the mesh could help to address this issue. The highlighting of the individual planar groups causing the increase in R_{pgv} also facilitates the understanding of the mesh structure and helps to identify the sections of the mesh in need of improvement.

5.2.3. Relative Density

Detection and comparison of the relative density of a scene's meshes serves to highlight 3D objects intended for the user's attention as well as to detect meshes which are supposed to be of equal significance for the user (e.g. part of the environment, interactable, intended for inspection), yet featuring differing levels of density and complexity. Both aspects draw from higher polygon counts implying a higher level of detail.
The rendering of the density rating of the scene, which can be seen in Fig. 5.15, shows severe discrepancies within the LODs of the scene. Most noticeable is the cushion lying on the smaller couch. While every piece of furniture is colored spring green, said cushion stands out due to its intense yellow color. Similarly, the cushion on the bigger couch, which is structurally the same mesh as the cushion on the small couch, but scaled to be slightly larger, received a green coloration. This makes it stick out as well. However, both cushions were intended to be part of the scenery and therefore of no more importance to the user than the table or the couches.
The controller lying on top of the table, on the other hand, was intended for the user to interact with. Yet, it is hardly discernible as the controller and the table, on which it lies, were assigned similar colors.

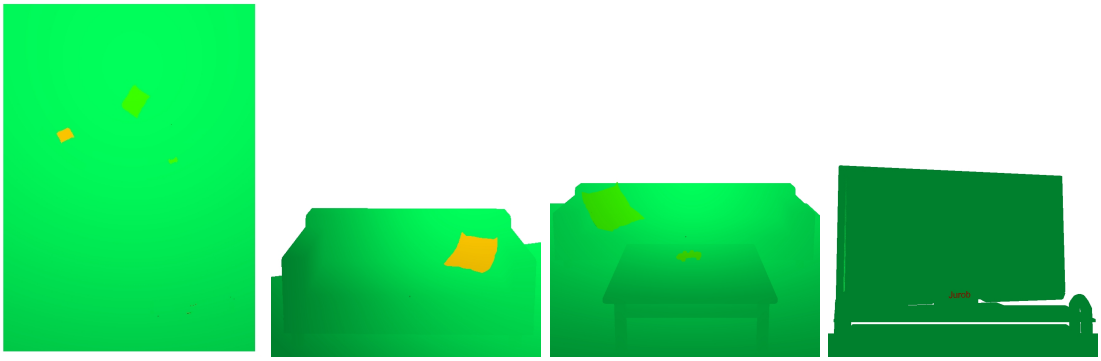


Figure 5.15.: R_{rd} visualization of the scene. The small cushion and the brand name of the TV stick out. The other elements of the scene are difficult to distinguish.

A mesh with significantly higher density than the remainder of the scene, though, can be found on the TV frame. The faux brand name uses a disproportionately big number of polygons to display a small detail. However, the raw data show that there is supposed to be another mesh with even higher density. To facilitate locating this mesh, I choose to add the *Relative* option to the rating to make use of the full color spectrum (see Fig. 5.16). The results mirror the previous findings. Most of the meshes which were intended to build the backdrop of the scene are of the least density. Once again the cushions and the controller are highlighted, though more visibly this time. The big cushion and the controller have a very similar level of density, whereas the small cushion is distinctly more dense. The brand name on the TV frame sticks out again, too. While exploring the scene on eye level I still fail to locate the other high density mesh, however. While hovering above the scene using bird's eye view, I can finally find the two small buttons on top of/behind the TV frame, which represent the most dense mesh of the scene (see Fig. 5.17, red circle).

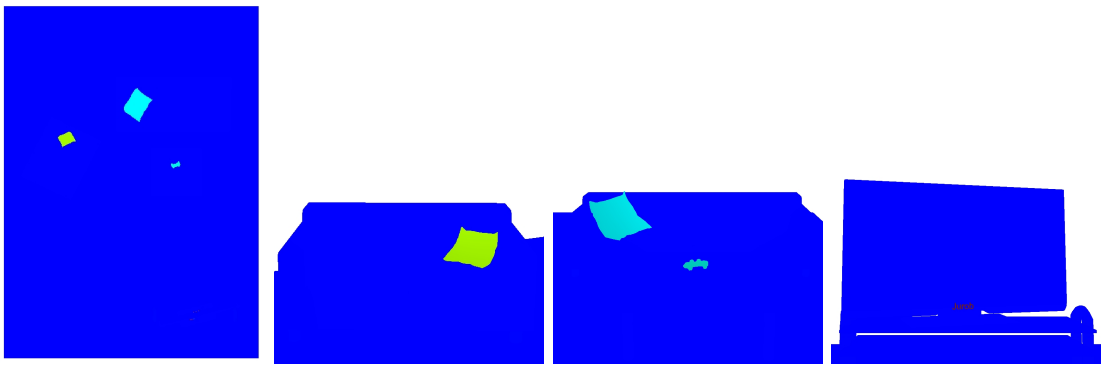


Figure 5.16.: R_{rel_rd} visualization of the scene. The big cushion and the controller are highlighted stronger than before.

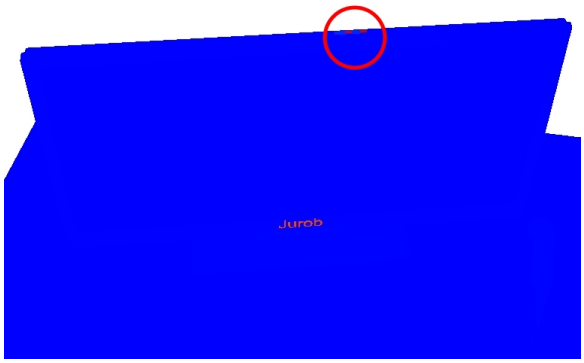


Figure 5.17.: Both highest density meshes

As can be seen in Fig. 5.15 and 5.16, the presence of outliers impedes the analysis of the remaining densities. Due to the use of linear interpolation, meshes with extremely high or extremely low densities cause the rest of the scene, consisting of average density meshes, to take on the same color. This makes comparing non-outlier meshes difficult or even impossible. The use of other interpolation methods, especially logarithmic interpolation, can reduce the outlier sensitivity of the analysis. Additionally, as shown by the two buttons behind the TV frame, outliers may be hard to detect. This is an inherent problem to meshes of low volume and high density, which makes them even more likely to be occluded by other objects in the scene. One possible solution would be the inclusion of the alpha channel into the visualization of the ratings: Making meshes of average rating partially transparent would help locating meshes of exceptional ratings. Alternatively, encoding the rating as scaling factor could increase the visibility of outlying meshes as well, as unexceptionally rated meshes would be displayed less prominently as meshes carrying extreme values.

5.2.4. Distance

Unlike the other ratings of the scene, each of the four visualizations of R_d in Fig. 5.18 shows different rating values for the scene. Furthermore, R_d is intended to quantify the importance of objects within the scene to the user. As the rating value of each mesh depends on its distance to the user, different user positions lead to different ratings. While the advancement of mesh color values along the spectrum (see Fig. 3.4) matches the progressive depth of the object relative to the user's viewpoint, two colors are lacking. Deep green, which ought to represent the mesh closest to the user, is not present in the third picture. On the other hand no mesh is colored deep red, which is meant to highlight the mesh with the largest distance to the user.

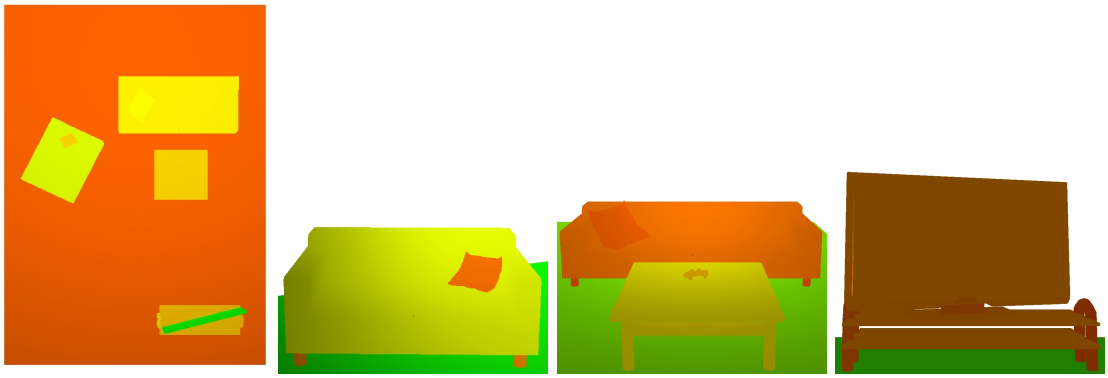


Figure 5.18.: R_d visualization of the scene. Coloration changes with the position of the viewer, since the rating depends on the current distance of each mesh from the viewpoint.

There are two causes for this behavior: R_d does not account for occlusion and includes objects outside of the viewing frustum. The rating and consequently the coloration of the scene is influenced by objects not visible to the user, which are thus of no relevance to the user and should not influence the rating. The third picture in Fig. 5.18 demonstrates the problem best, as it contains neither deep green nor deep red meshes. Neither the floor the user is standing on nor the table directly in front of them is rated as most important. In their stead the TV stand behind the user received a rating of $R_d = 0$, even though it is unobservable in this configuration. Similarly, the two rear feet of the couch are the two meshes with the largest distance to the user and colored deep red. However, they are obscured by the couch itself.
Therefore, accounting for occlusion in the metric formula would be mandatory to make the resulting ratings representative. The ideal implementation would be able to draw from the culling algorithms used during the eventual rendering and convert these into modifications of the distance rating. A huge and comparatively simple improvement of the metric would be the filtering of meshes based on their visibility to the user. Meshes outside of the user's viewing frustum as well as meshes completely obscured by other close meshes would not be included in the gathering of individual mesh distances, which is then used to determine R_d. This would implement both occlusion culling as well as viewing frustum culling.

5.2.5. Co-Occurrence of Triangle Sizes

Highlighting of the contrast of co-occurrences of triangle sizes is meant to identify uneven meshes. On the one hand irregular triangle sizes can be intentional. For example, a mesh can contain large and small triangles in close vincinity as a result from adding small details to a raw 3D object. On the other hand inconsistencies in triangle sizes can be a characteristic of modelling errors.
The visualization of R_{ccm} of the scene, which can be seen in Fig. 5.19 shows that the TV screen is clearly different-colored than the rest of the scene. The deep red, and the high R_{ccm} value causing it, are the result of the mesh comprised of only two triangles with slightly differing surface areas.

Thus, its Co-occurrence Matrix contains but two entries in its 2x2 dimensions: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. This represents the maximum possible contrast and leads to a R_{ccm} value of 1.

The other salient colors were assigned to the feet of the table and the couches. Each of these prisms features but two different kinds of triangles: Those composing the lateral area and those spanning the base and top areas. Respectively all triangle of each kind are congruent and therefore of equal size. Consequently, each triangle has two equal sized triangles and one triangle with the most difference in area value present within the mesh as neighbors. This leads to a small Co-occurrence Matrix with 2/3 of its entries along the main diagonal and 1/3 in the top right and bottom left corners. This, in turn, corresponds to a contrast of 1/3 and results in a yellow green color.
The remainder of the furnishing, in particular the feet of the TV stand and the controller, are dyed in similar shades of deep green. This means that each of these meshes features no or only gradual changes in the sizes of the underlying triangles. The controller owes its smoothness to its uniform mesh aperture. Even its relatively large planar areas are modelled using several small triangles instead of fewer and bigger ones. The feet of the TV stand do not share the R_{ccm} of the other feet of the furniture due to its more efficiently modelled base area.

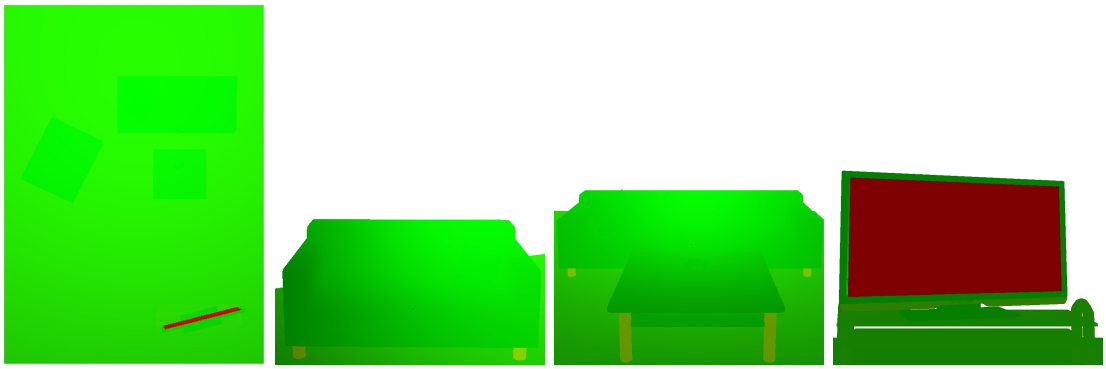


Figure 5.19.: R_{ccm} visualization of the scene. The TV screen stands out significantly. The furniture legs are highlighted to a lesser extent.

As previously shown, R_{ccm} delivers rather extreme results when but few differently sized triangles are present in a mesh as a result of the mesh using very few triangles itself. I normalize the metric using the difference of smallest and the largest triangle of the mesh. With only those two triangles present, as is the case with the TV screen, the metric always reaches a value of 1. This rating rightly represents the strongest possible fluctuation of triangle sizes given those components, even though the absolute difference is possibly very small. This disregards the absolute difference of the smallest and the largest triangle size compared to the triangle sizes themselves. A possible remedy would be the addition of this quotient, either as another factor by which R_{ccm} gets multiplied or a new metric altogether. Furthermore, R_{ccm} is insensitive to strong, but infrequent differences in triangle areas. The reason for this is that the sheer numbers of similarly sized triangles are able to drown out large rare differences, as I designed the metric to take the whole structure of the mesh into account: the bigger the amount of equivalent triangles in the mesh the larger the neccessary size difference to noticeably impact the rating. R_{ccm} is therefore not suited to identify the

scarce strong outliers among the triangle sizes, which are present in the mesh of the table, the TV stand and, to a certain extent, the couches.

5.2.6. Fourier Transform

Since the Fourier transform metric is supposed to detect meshes with stark fluctuations of triangle sizes, I expect similar results as those returned by R_{ccm}. The visualization of the results of R_{ft} in Fig. 5.20, however, shows that this is not the case. While a few meshes are rated similarly, like the floor, most of the rating colorations are different. The most noticable difference lies within the (correctly) different rating of the TV screen. Whereas R_{ccm} is too sensitive to the slight size difference of the two triangles the TV screen is composed of, the R_{ft} rating of the screen resulted in a deep green color. Additionally, R_{ft} is able to pick up the stronger variance of triangle sizes present in the legs of the table as opposed to the variance in the legs of the couches. While R_{ccm} rates the couch legs as well as the table legs consistently around 0.333, R_{ft} rates the couch legs at 0.092 and the table legs at 0.134 throughout (see Table 5.10). The difference in rating values is the result of the table legs being significantly taller than the couch legs, which leads to bigger triangle sizes on the sides of the prisms, while the triangles at the base and top areas remained unchanged.

Figure 5.20.: R_{ft} visualization of the scene. There exists no severe outlier, but several varying small ratings.

Fig. 5.21 shows a graphical representation of the input signal generated by the mesh of the TV buttons. One can see that there exist a wide variety of triangle area sizes. It also entails several strong individual peaks at irregular intervals. The corresponding output sequence generated by the DFT shows a fluctuation in the magnitudes of the frequencies. Multiple frequencies reach magnitudes of more than $0.4 \cdot |X_0|$. This means that the mesh, which received a R_{ft} rating of 0.3091, is shaped unevenly. The signal generated by the mesh of the controller, which can be seen in Fig. 5.22, on the other hand, displays a weak variation of relative area size. Over a series of 4404 data points it contains but two strong peaks and few smaller outliers. The application of the DFT to this series leads to only weak frequencies of various oscillation rates. Thus, the mesh is comparatively uniform, which is also reflected in its R_{ft} rating of 0.0861.

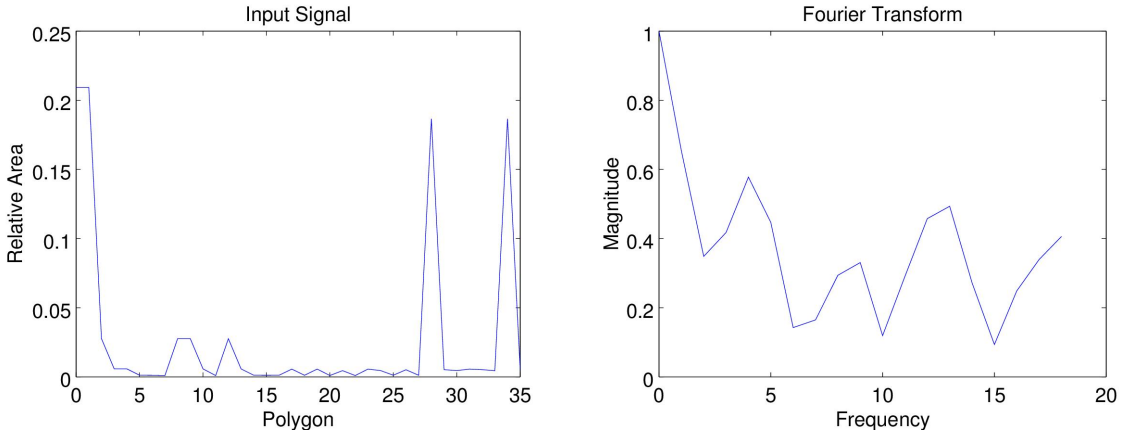


Figure 5.21.: Input Signal and corresponding DFT of the case of the TV buttons

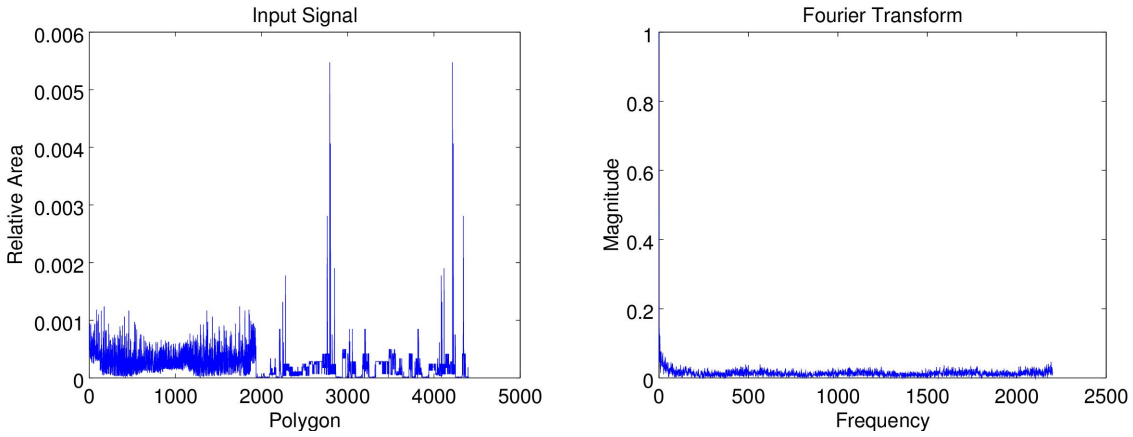


Figure 5.22.: Input Signal and corresponding DFT of the controller

However, the ordering I use is biased regarding triangle sizes. The sorting by distance favors small triangles, since the distance of a triangle's center from the center of its edges is directly proportional to its area. Furthermore, the ordering discourages directionality, as ordering is always dependent on the distance from the unchanging referential triangle. This results in many cases where in the ordered list of triangles consecutive triangles are not even remotely adjacent.
Thus, while the metric delivers meaningful results for a given one-dimensional signal, the signal generating algorithm requires several modifications. The changes to the mapping of the triangles onto a one-dimensional series need to include the preservation of spatial proximity of the triangles. Additionally, a weighting which favors triangles that continue the path of selected triangles in the same direction as the previously chosen triangles could increase the expressiveness of the signal. A signal of pairwise adjacent triangles created by iterating the surface of the mesh in a semi-fixed direction could be more representative of the mesh. Further implementations could also include several iterations along differing directions and combining the resulting signal to further increase the fidelity of the signal describing the mesh.

5.2.7. Relative Average Rendering Time

As R_{rt} is purely empirical, it can be used to make accurate predictions regarding the cost of rendering a specific mesh. It's values scale mostly with the number of polygons used to build the mesh. However, rendering specifications and variations, for instance optimizations like backface culling, can be considered by the metric. Provided that the initial gathering of rendering time data is executed under the same circumstances as the final use, which the scene was built for, dictates, R_{rt} allows realistic and accurate predictions, which otherwise could be difficult to precalculate and quantify.
The rendering of the values of R_{rt} of the scene shows a largely monochrome scene (see Fig. 5.23). Merely both cushions catch the eye of the observer, as their coloration stands out clearly in the otherwise deep green scene. Interestingly, the cushions are colored differently with the small cushion being deep red while the big cushion features a shade of yellow. Structurally, with the exception of a small scaling factor, both cushion meshes are identical and contain the same number of triangles and vertices. Yet, the small cushion is rated significantly higher, even though the rendering of the big cushion requires the calculation of more fragments and therefore more time. This discrepancy arises from the susceptibility of empirical measuring to external factors. The weaker the produced signal, when compared to noise, the more imprecise is the measured signal. Still, both cushions, which use considerably more polygons than the other meshes, stand out from the scene. The controller, however, fades into the background once again. This means that in order to improve the scene towards respective adequate mesh complexity to meet appropriate levels of detail, the cushion meshes need to be reduced severely, while the controller can be provided with more details - and therefore more polygons.

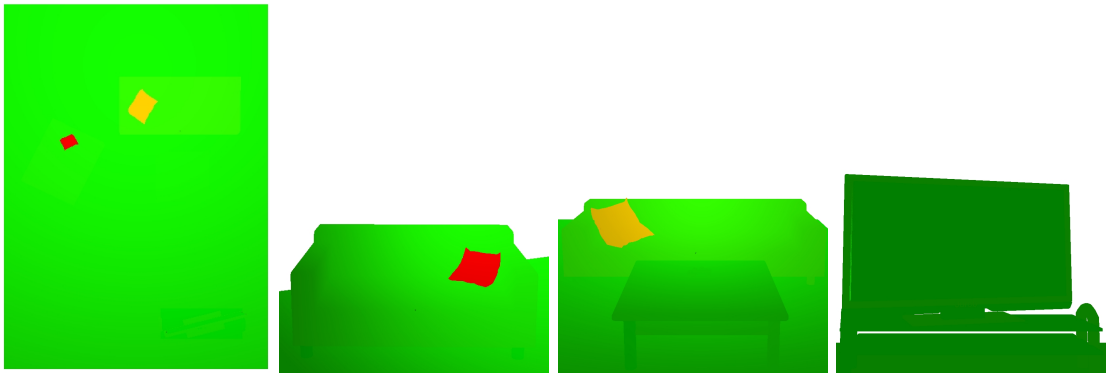


Figure 5.23.: R_{rt} visualization of the scene. Both cushions distinctly stand out.

As a metric which entirely uses empirical data, R_{rt} is susceptible to all accompanying problems. The measured rendering time depends on the hardware used, ambient conditions as well as rendering options. While increasing the amount of measured data points works to reduce the influence of changes in ambient conditions on the data, the measurements remain specific to the used hardware and software. This represents a trade-off, since the data are as realistic and representative as possible for a given setup at the cost of not being directly applicable to different configurations. Additionally, due to the use of linear interpolation, the metric is sensitive to outliers. Different methods of interpolation could be used to curb the influence of outliers on the ratings.

Mesh	Polygons	R_{pppg}	R_{pgv}	R_{rd}	R_{ccm}	R_{ft}	R_{rt}
Big couch	1430	0.5115	0.1284	-0.1765	0.0074	0.1803	0.0914
Big couch: front right leg	40	0.7180	0.3873	-0.1720	0.3332	0.0919	0.0249
Small couch: back right leg	40	0.7180	0.3873	-0.1720	0.3333	0.0919	0.0139
Controller	4404	0.3518	0.0299	0.1402	0.0012	0.0861	0.0772
TV stand: both boards	120	0.7059	0.2517	-0.1771	0.0886	0.3431	0.0449
TV stand: left legs	256	0.1412	0.0574	-0.1722	0.0997	0.1421	0.0398
TV buttons: case	36	0.2286	0.1571	-0.1247	0.1223	0.3091	0.0081
TV screen	2	1.0000	1.0000	-0.1773	1.0000	0.0000	0.0147
Small cushion	40612	0.0079	0.0009	0.6063	0.0001	0.0610	0.7478
Table: bottom boards	332	0.3354	0.0635	-0.1763	0.0158	0.2851	0.0302
table: back left leg	40	0.7180	0.3873	-0.1751	0.3333	0.1337	0.0290
Small couch: back left leg	40	0.7180	0.3873	-0.1720	0.3332	0.0919	0.0488
Table: top boards	332	0.3867	0.0682	-0.1766	0.0158	0.2861	0.0749
TV stand: right legs	256	0.1412	0.0574	-0.1722	0.0997	0.1421	0.0660
Base of TV	252	0.3028	0.0842	-0.1752	0.0209	0.4248	0.0545
TV casing: bottom sides	82	0.1605	0.0960	-0.1751	0.0896	0.2430	0.1182
Big couch: back right leg	40	0.7180	0.3873	-0.1720	0.3332	0.0919	0.0313
TV frame: two buttons	584	0.2161	0.0618	1.0000	0.0238	0.1625	0.0915
Big couch: front left leg	40	0.7180	0.3873	-0.1720	0.3332	0.0919	0.0501
Floor	32	1.0000	1.0000	-0.1773	0.0758	0.0000	0.0475
Small couch	1430	0.3933	0.0723	-0.1761	0.0084	0.1520	0.0674
Table: back right leg	40	0.7180	0.3873	-0.1751	0.3333	0.1337	0.0388
Table: front right leg	40	0.7180	0.3873	-0.1751	0.3332	0.1337	0.0485
TV: brand name	2244	0.2893	0.0242	0.9212	0.0029	0.0858	0.0935
TV frame	360	0.4401	0.1038	-0.1763	0.0074	0.3129	0.0674
Big cushion	40612	0.0071	0.0008	0.1141	0.0001	0.0609	1.0000
Small couch: front right leg	40	0.7180	0.3873	-0.1720	0.3333	0.0919	0.0000
Small couch: front left leg	40	0.7180	0.3873	-0.1720	0.3333	0.0919	0.0027
TV casing: bottom front	22	0.3810	0.2911	-0.1758	0.1755	0.3704	0.0014
TV stand: rack	196	0.3231	0.1225	-0.1740	0.0209	0.2705	0.0085
Table: front left leg	40	0.7180	0.3873	-0.1751	0.3332	0.1337	0.0031
Big couch: back left leg	40	0.7180	0.3873	-0.1720	0.3333	0.0919	0.0031

Table 5.10.: Scene: Calculated ratings by order of rendering

6. Conclusion and Future Work

I have presented an automated method to investigate geometrical characteristics of 3D objects and to highlight the differences found between meshes which are part of the same scene. I designed seven metrics which can be used to analyze different aspects of a mesh: the degree co-planarity of adjacent polygons, the distribution of mesh density in the scene, the importance to the user due to proximity, the fluctuation of polygon sizes and the differences in rendering time. I used normalized and standardized value sets for each metric allowing easy combination and comparison of otherwise heterogeneous data. Furthermore, I implemented a function to map the ratings on color values to visualize the results of the analysis in a comprehendable way.
After detailing the hardware limitations one faces during the implementation of my concept, as well as my corresponding solutions, I presented the results of my automatic analysis of several simple exemplary scenes and one complex scene. I then investigated whether the returned results were fit to represent the respective meshes.
Planar Grouping, and its corresponding metrics, deliver satisfactory results when judging meshes based on their co-planarity. However, these rating values lose meaning when ill-fitting primitives are chosen for the analysis. Investigating a cube regarding the co-planarity of its triangles, for example, leads to results which are correct, but can be misleading. In this case the use of quadrilaterals would lead to more representative results.
The comparison of mesh densities returns useful results as well, although the visualization showed room for improvement. As a relative metric it is susceptible to distortion of the ratings by outliers. Hence, a different interpolation than linear interpolation or a preceding filtering and handling of outliers could improve the perceivability of the visualization of the rating. This holds true for the other relative metrics, distance and rendering time, too.
The rating of the distance of the meshes to the user, however, proves mostly lackluster. As it does not account for culling, particularly frustum culling, occluded meshes are rated non-zero, even though they are not visible to the user. This is a clear violation of the intention behind the metric, since meshes which are not rendered should not receive a recommendation for a higher LOD by the metric. However, a rework of the Relative Distance metric would be an opportunity to not only include culling, but also measures to gauge whether a mesh is the user's center of attention or wether it resides in their visual periphery.
The Fourier transform and the co-occurrence matrix both serve to identify fluctuations in triangle sizes along the surface of a mesh. The metric representing the contrast of the co-occurrence matrix delivers meaningful results, which makes it worthwhile to explore further ways to construct the co-occurrence matrix. For example, including indirect neighbors by using their degree of adjacency as inverse weight could lead to more intuitive and representative ratings. Additionally, the metric would profit from another factor which addresses the proportion of the size difference of the largest and smallest triangle to the size of the smallest triangle. The Fourier transform metric, on the other hand, while functional in theory, works only occasionally during practical applications. Researching different methods to map three-dimensional arrays of polygons onto a one-dimensional series in such a way that

the directionality of the iteration along the mesh surface is conserved would help create a suitable input signal.
Lastly, the rendering time based metric works flawlessly under the caveat common to empirical measurements: different environmental conditions can lead to different data.
The next step for the metrics I designed is the correction of their shortcomings by adjusting the parameters and algorithms. The reworked and polished versions would then be suitable for the neccessary user studies which would be the next testing iteration as the mathematical revision has now been established. These can also be used to identify color schemes which would make small differences in ratings easier to distinguish.
A vast field of opportunities regarding the design of further metrics is based on modifications and adjustments of the metrics I devised and presented. As I previously mentioned, my algorithms use but triangles as base primitives, which is not always the optimal choice. Therefore, a generalisation of the metrics to include different kinds of n-polygons would ensure a more universal applicability. Similarly, the analysis of different attributes of the 3D object could supply the user with a broader set of analysis tools. Among these approaches are, for example, the use of polygon size instead of co-planarity to segment the mesh, the frequency analysis of vertex colors and textures and the empirical measuring of the amount of fragments during the rendering process.
The final and most directly useful future work entails the fully automated processing of the mesh ratings to improve and optimize a scene. The current implementation serves only to visualize meshes with possible room for improvement to the user. A more sophisticated approach could use the ratings to segment a mesh based on local minima and maxima of the ratings. This would also solve the problem of the ratings being mesh-specific instead of region-specific, which leads to varying results depending on the composition of the 3D object, e.g. a table being supplied as a single mesh or as several meshes for the different components like legs and table board. The segmentation would thus represent the individual sub-regions in need of improvement, to which established smoothing algorithms could be applied.

7. Acknowledgements

First and foremost, I would like to thank Lea Weil for providing me with the test scene, which can be seen in chapter 5. Her perspective and graphical work were very valuable and helped in the finding of several results of my thesis. She benefitted from Michael Käsdorf's experience as she received his help with the necessary modelling tools.
Further help in the form of 3D test models and creative input was provided by Tanja Neumayer, which were very important in the nascent stage of the code development.
Last, but certainly not least, I want to thank my girlfriend Lara Hirschbeck for guiding me through the interface of Blender, for processing 3D models and for providing helpful discussion on the topic.

List of Figures

Bibliography

[Ass16] SD Card Association. Capacity (sd/sdhc/sdxc). `https://www.sdcard.org/developers/overview/capacity/`, 2016. [Online; accessed 8-November-2016].

[Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[FJ16] Matteo Frigo and Steven G. Johnson. Fastest fourier transform in the west. `http://www.fftw.org/`, 2016. [Online; accessed 14-October-2016].

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 247–254, New York, NY, USA, 1993. ACM.

[GF08] Aleksey Golovinskiy and Thomas Funkhouser. Randomized cuts for 3d mesh analysis. In *ACM transactions on graphics (TOG)*, volume 27, page 145. ACM, 2008.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 19–26, New York, NY, USA, 1993. ACM.

[HH93] Paul Hinker and Charles Hansen. Geometric optimization. In *Proceedings of the 4th Conference on Visualization '93*, VIS '93, pages 189–195, Washington, DC, USA, 1993. IEEE Computer Society.

[HS+73] Robert M Haralick, Karthikeyan Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[IKN+98] Laurent Itti, Christof Koch, Ernst Niebur, et al. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[Inc16] Apple Inc. iphone 7 tech specs. `http://www.apple.com/iphone-7/specs/`, 2016. [Online; accessed 8-November-2016].

[KFR03] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[KG00] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LJT01] Zheng Lin, Jesse Jin, and Hugues Talbot. Unseeded region growing for 3d image segmentation. In *Selected Papers from the Pan-Sydney Workshop on Visualisation - Volume 2*, VIP '00, pages 31–37, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh saliency. In *ACM transactions on graphics (TOG)*, volume 24, pages 659–666. ACM, 2005.

[Mar16] Andrew Martonik. The galaxy s7 and s7 edge will only come in 32gb internal storage variants. `http://www.androidcentral.com/galaxy-s7-and-s7-edge-will-only-come-32gb-variants-us`, 2016. [Online; accessed 8-November-2016].

[Pir13] Usman Pirzada. Ryse polygon count comparison with other aaa titles - star citizen, crysis 3 and more. `http://wccftech.com/ryse-polygon-count-comparision-aaa-titles-crysis-star-citizen/`, 2013. [Online; accessed 25-June-2016].

[PSG+06] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(3):549–559, 2006.

[Rec95] ITURBT Recommendation. 601-6: Studio encoding parameters of digital television for standard 4: 3 and wide screen 16: 9 aspect ratios. *International Telecommunication Union*, 96, 1995.

[Rup14] Karl Rupp. Cpu, gpu and mic hardware characteristics over time. `https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/`, 2014. [Online; accessed 25-June-2016].

[Sch97] William J Schroeder. A topology modifying progressive decimation algorithm. In *Visualization'97., Proceedings*, pages 205–212. IEEE, 1997.

[SF07] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3d surfaces. *ACM Transactions on Graphics (TOG)*, 26(2):7, 2007.

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM, 1995.

[Tea16] Assimp Development Team. Open asset import library. `http://assimp.sourceforge.net/`, 2016. [Online; accessed 14-October-2016].

[TFTN05] Shigeo Takahashi, Issei Fujishiro, Yuriko Takeshima, and Tomoyuki Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 495–502. IEEE, 2005.

[Yeu10] Karlie Yeung. 3ds cartridges could store up to 8gb. `http://www.nintendoworldreport.com/news/24569/3ds-cartridges-could-store-up-to-8gb`, 2010. [Online; accessed 8-November-2016].

A. Appendix

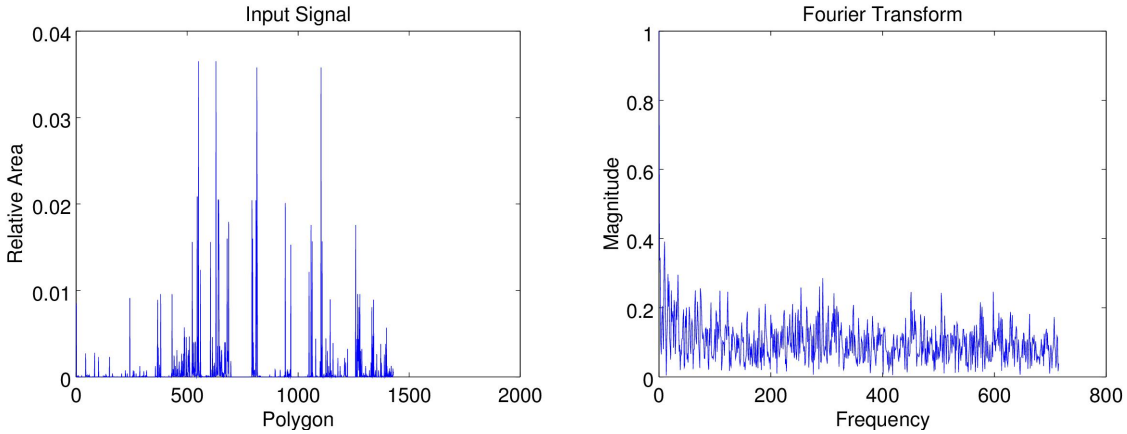


Figure A.1.: Input Signal and corresponding DFT of the big couch

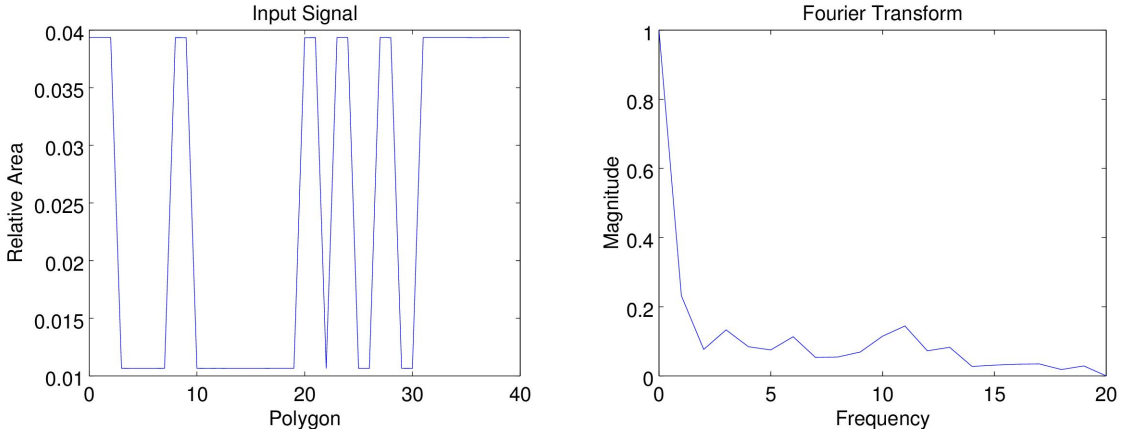


Figure A.2.: Input Signal and corresponding DFT of a couch leg

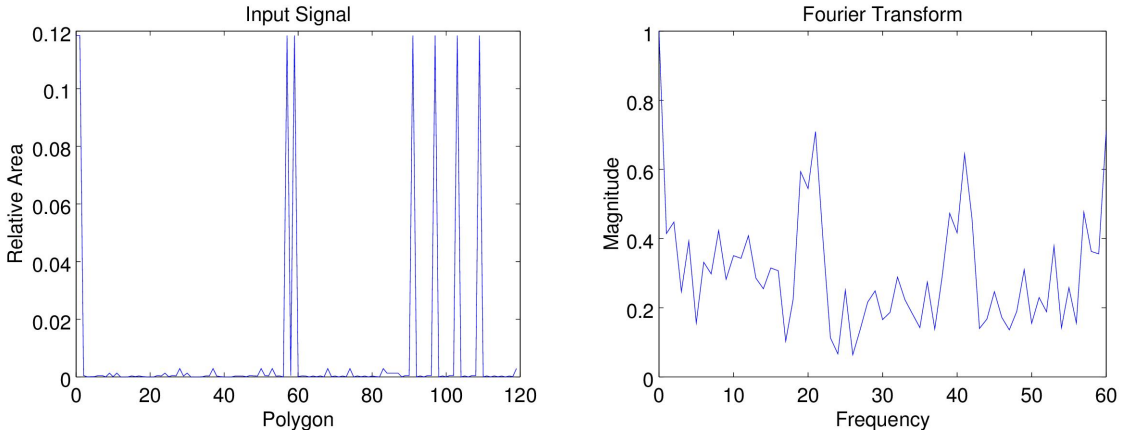


Figure A.3.: Input Signal and corresponding DFT of the boards of the TV stand

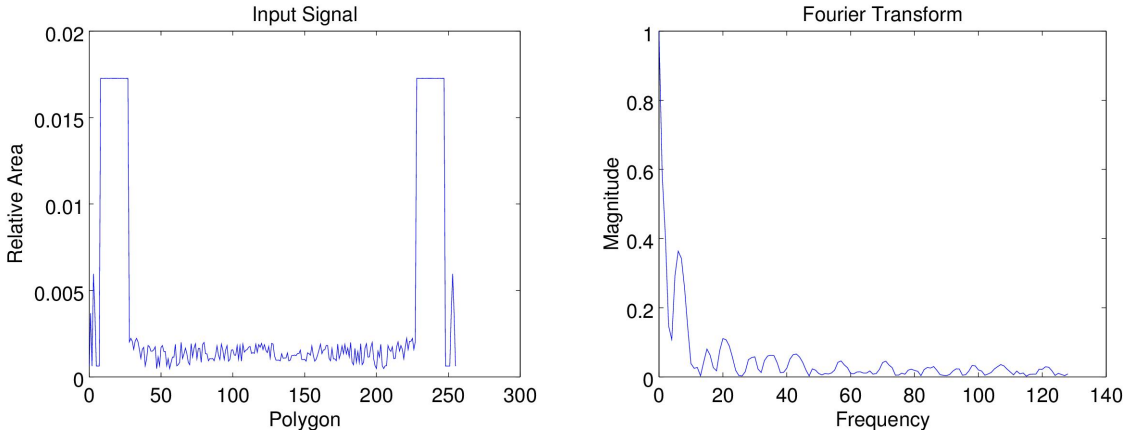


Figure A.4.: Input Signal and corresponding DFT of the legs of the TV stand

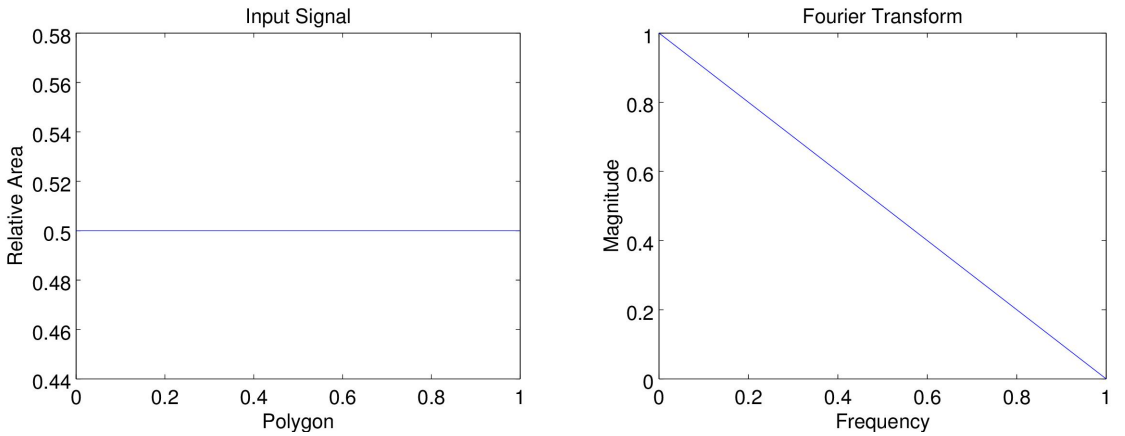


Figure A.5.: Input Signal and corresponding DFT of the TV screen

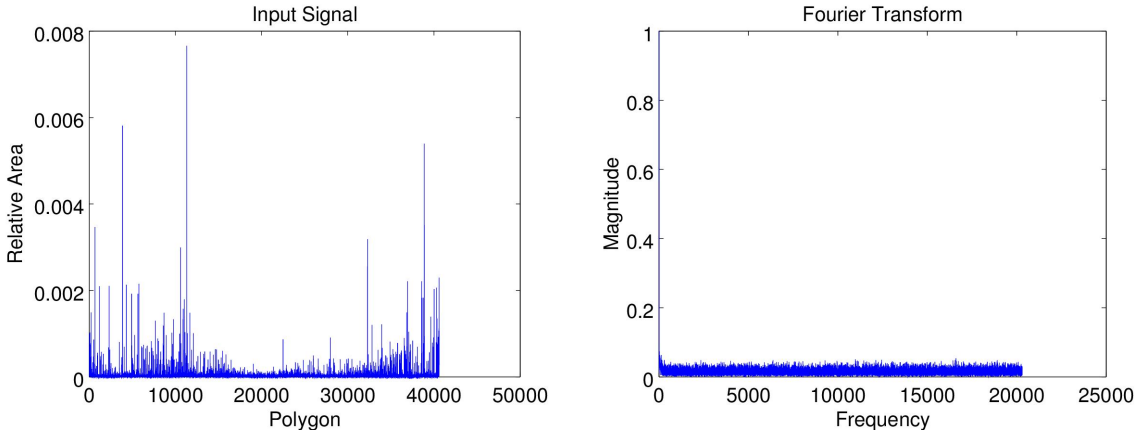


Figure A.6.: Input Signal and corresponding DFT of a cushion

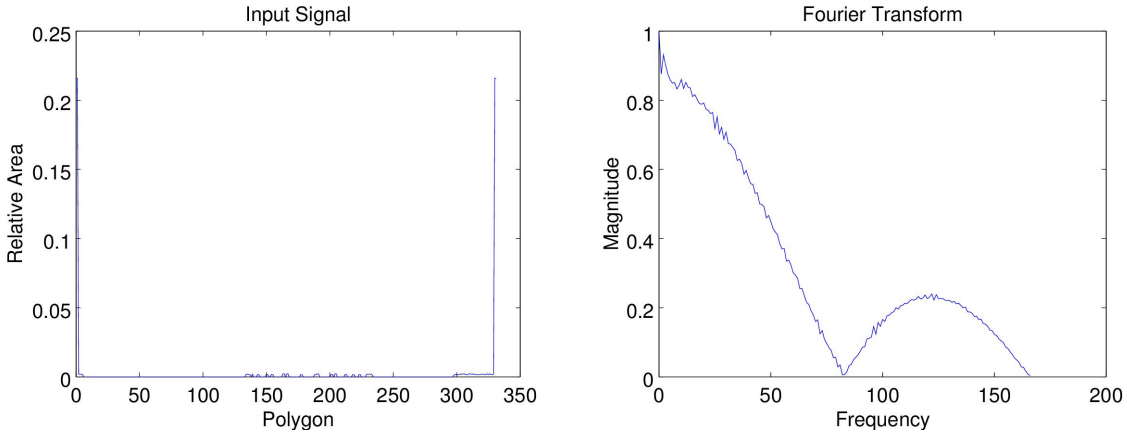


Figure A.7.: Input Signal and corresponding DFT of the bottom board of the table

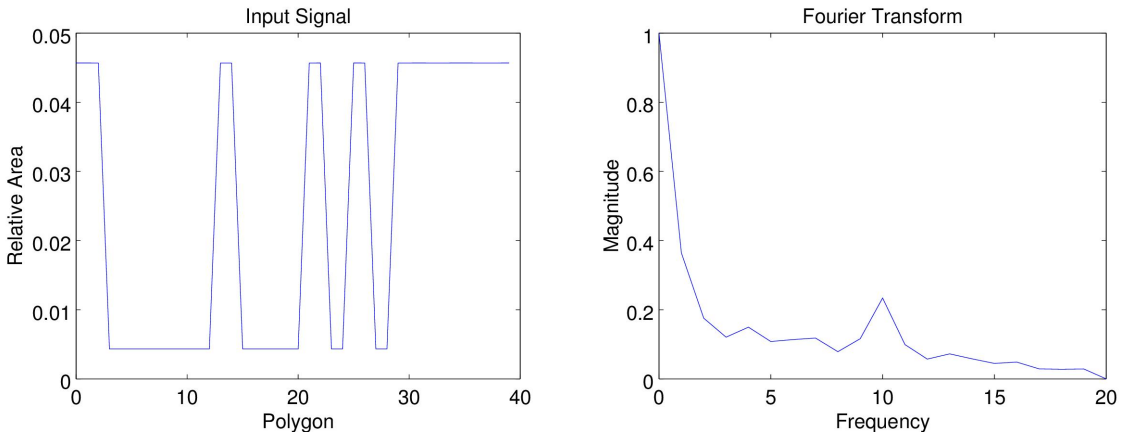


Figure A.8.: Input Signal and corresponding DFT of a table leg

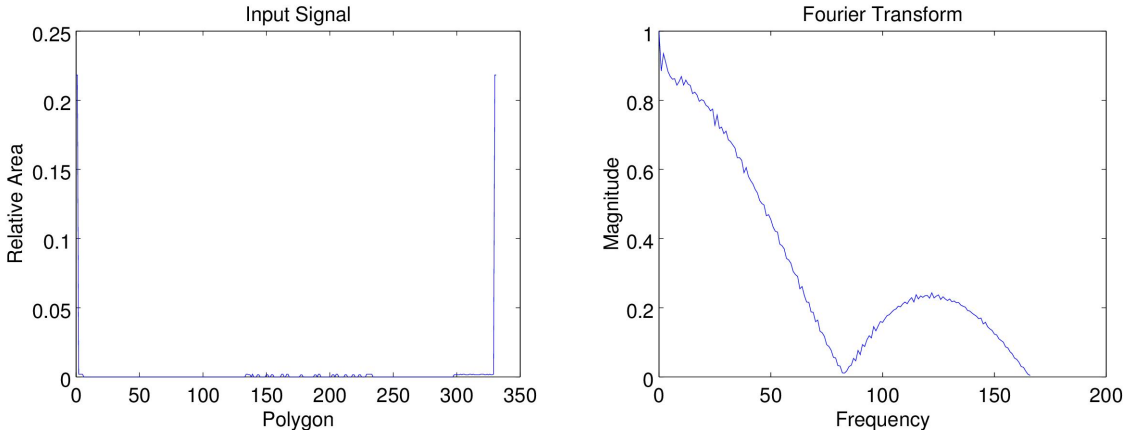


Figure A.9.: Input Signal and corresponding DFT of the top board of the table

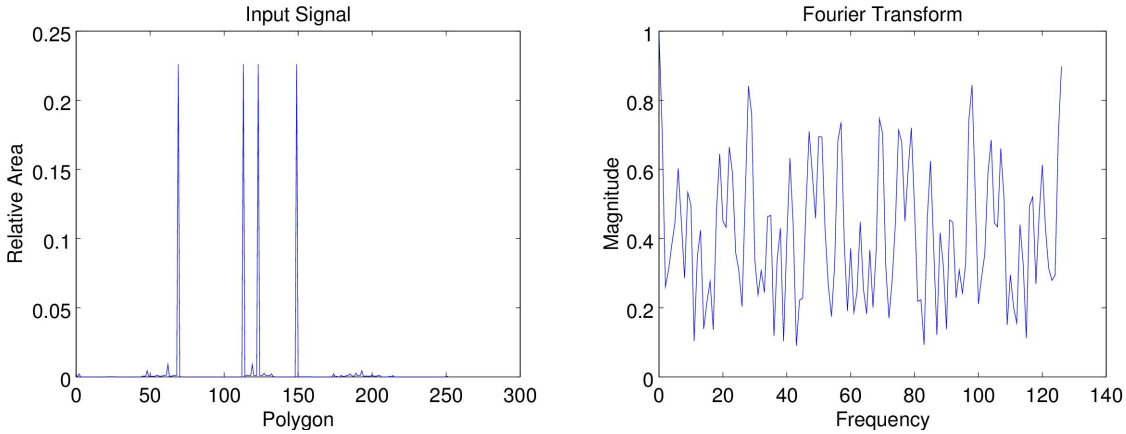


Figure A.10.: Input Signal and corresponding DFT of the base of the TV

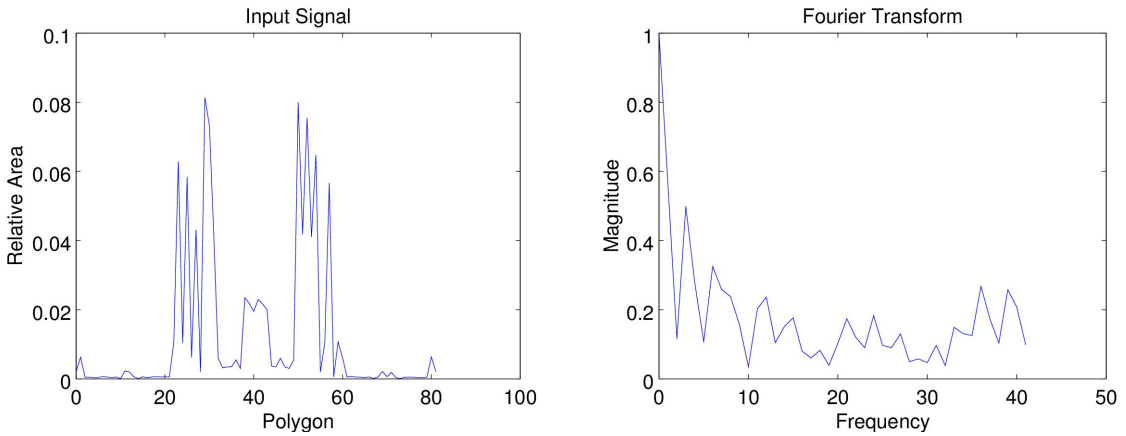


Figure A.11.: Input Signal and corresponding DFT of the sides of the bottom TV casing

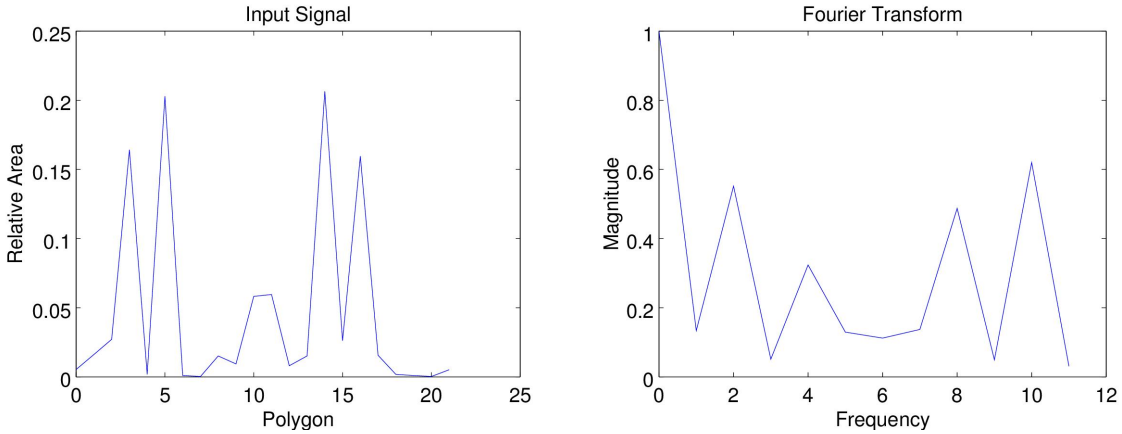


Figure A.12.: Input Signal and corresponding DFT of the front of the bottom TV casing

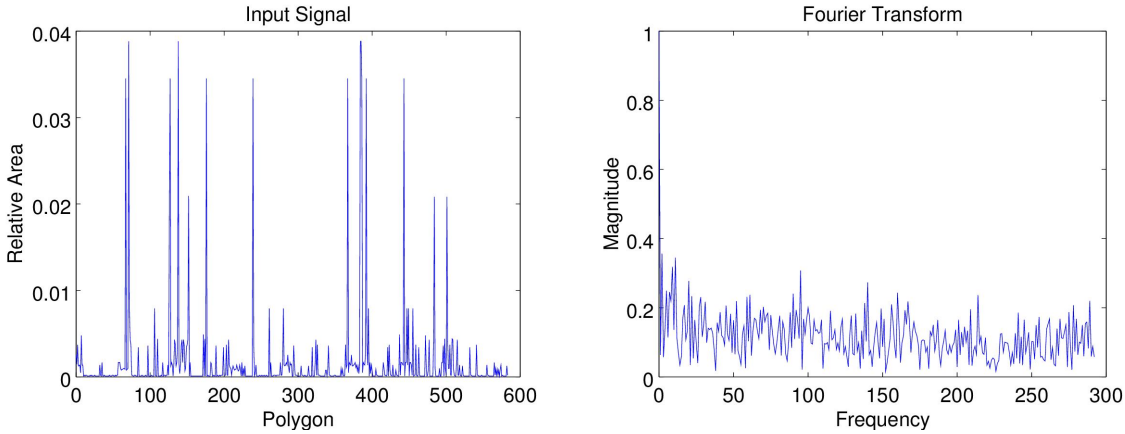


Figure A.13.: Input Signal and corresponding DFT of the TV buttons

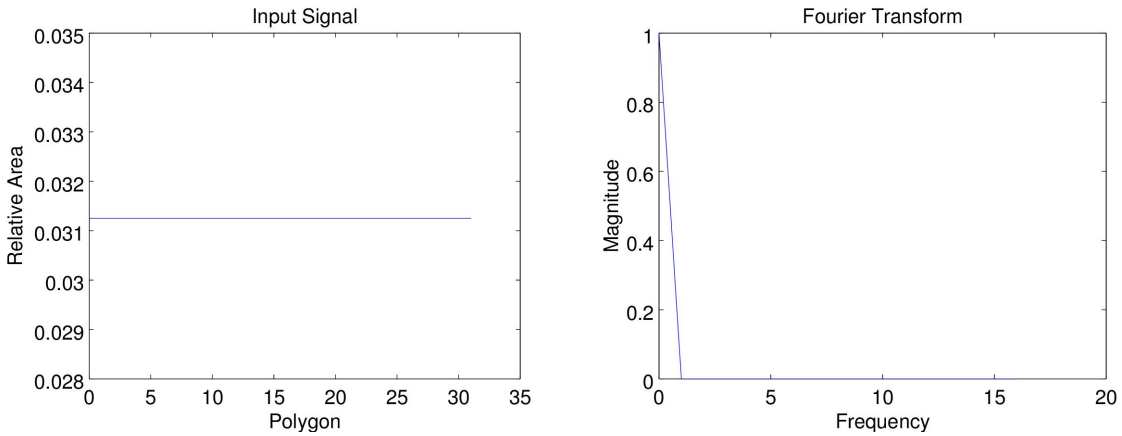


Figure A.14.: Input Signal and corresponding DFT of the floor

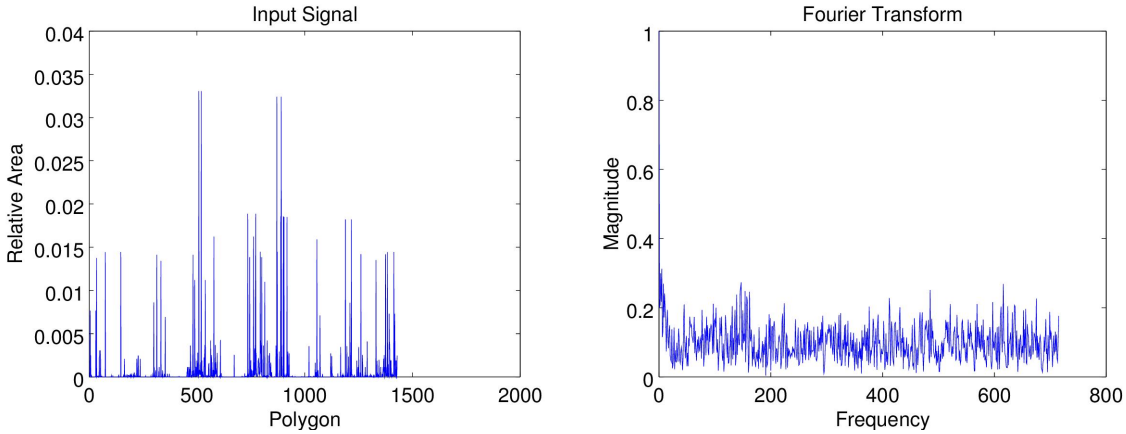


Figure A.15.: Input Signal and corresponding DFT of the small couch

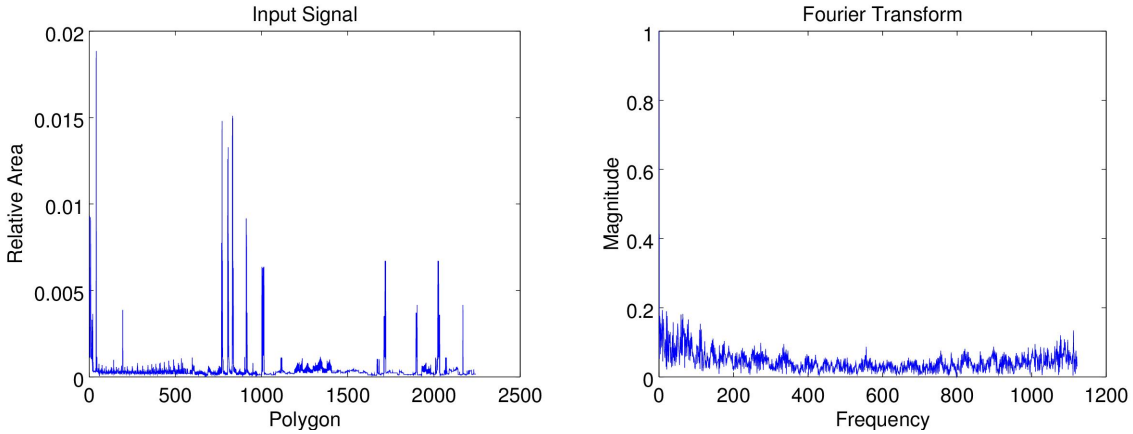


Figure A.16.: Input Signal and corresponding DFT of the brand name of the TV

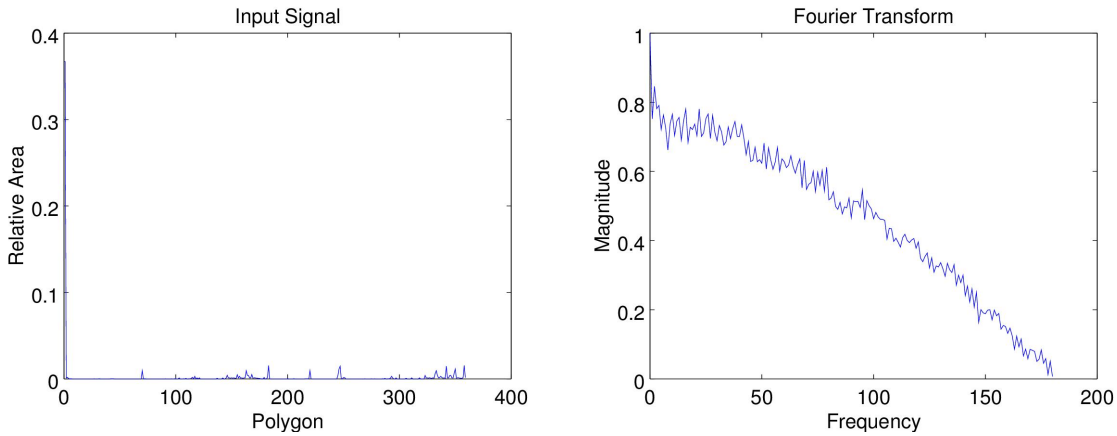


Figure A.17.: Input Signal and corresponding DFT of the TV frame

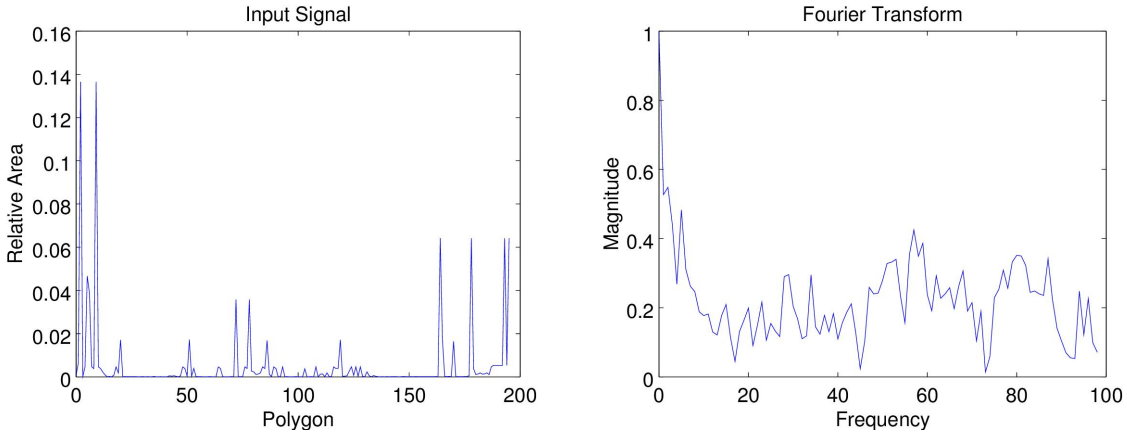


Figure A.18.: Input Signal and corresponding DFT of the rack of the TV