TEXTURE-BASED SEPARATION TO REFINE BUILDING MESHES

Jelle Vermandere[1], Maarten Bassier[1], Maarten Vergauwen[1]

[1] KU Leuven, Department of Civil Engineering, Ghent, Belgium
(jelle.vermandere, maarten.bassier, maarten.vergauwen)@kuleuven.be

KEY WORDS: Mesh, segmentation, texture separation, edge detection, UV mapping

ABSTRACT:

In this paper, we present a texture-based separation approach to refine building meshes, which aims to address the challenges of detecting and isolating different objects in an indoor scene mesh. We propose a novel segmentation model based on the materials of the different parts of the scene. The proposed approach uses factorization-based texture segmentation to separate the different materials in the meshes and detect the edges on the segmented texture. To prevent large faces containing multiple materials from being segmented wrongly, the mesh is cut along the texture boundaries and new faces with a single material are created. Finally, we segment the new faces based on the material index and neighbouring faces using a region-growing algorithm. We evaluate the proposed approach on the Matterport indoor dataset, which shows that our approach performs well on detecting boundaries between distinct materials, but over-segments on complex shapes. Our proposed method improves the segmentation of large flat surfaces like posters or rugs.

1. INTRODUCTION

Indoor scene segmentation of environments in the Architecture, Engineering and Construction (AEC) industry is a field of ongoing research. Current state-of-the-art data acquisition techniques can record highly detailed and dense 3D data but lack the required information layers to efficiently use the building data in applications (Geyter et al., 2022). One of these extra information layers is object segmentation, where the captured environment is split into distinct parts. This problem has been approached from many angles before (Mao et al., 2022) and although the majority of the scenes can be automatically segmented, there are still a large number of edge cases where the segmentation process returns invalid results. These exceptions require tedious and labour-intensive human intervention to achieve good results.

Current segmentation methods largely rely on one of two data types. The first data types are 2D images, where by using trained networks, it is possible to find objects displayed on the images (Zhang et al., 2021). While these methods can accurately and precisely segment a scene, due to occlusions caused by overlapping objects and limited 3D representations, they provide less reliable results in more complex environments. Because the segmentation is only performed on a 2D image, it is difficult to accurately predict the 3D location of the object boundaries.

The second group uses 3D information as a base to segment the scene into 3D bounding boxes or regions. These mostly rely on geometric features like point locations or normals (Schult et al., 2022). Due to the need for a standardised input in neural networks, the 3D segmentation is mostly performed on normalised point clouds or voxel occupancy grids. Using points allows the data to be easily sub-sampled to the correct density, this is why most datasets which do not use this format get converted before the segmentation process. This is especially troublesome for data types like meshes, where a large amount of detail is often simplified in the geometry and only visible in the texture.

Meshes are an efficient way to store data compared to point clouds and are better at representing surface and texture detailing (fig.1). However, detecting and segmenting different objects in an indoor scene mesh can be challenging due to the sparsity of the faces. This sparsity can lead to ambiguity in large faces which are supposed to be part of multiple objects (Bassier et al., 2020).

The textures of a mesh are stored on a 2D image, where each vertex contains a UV coordinate which can be mapped onto the image (Heckbert, 1986). The colour of the face is then calculated based on the UV position of its three vertices. This emphasis on geometric abstraction is done to save on storage and processing costs, something very desirable in most industries. By sampling the mesh into a point cloud the texture detail gets lost and is reduced to a single color per point. The point color is often used as an extra parameter in geometry-based segmentation. However, due to the complex patterns present in some materials like bricks or wallpapers, this extra parameter can lead to more confusion than clarity.

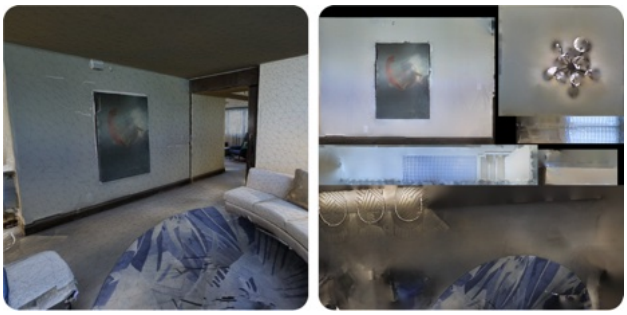

Figure 1: 3D mesh (left) and its corresponding texture map (right)

Our goal is to create a texture-based segmentation model that can cleanly segment a mesh. We propose using a texture-based boundary detection method on a texture map to segment the different zones of the textured mesh. To ensure every face only belongs to a single element, new edges are created at the detected boundaries. Each face is then assigned a material index and connected faces are grouped together. This will create a num-

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-1-W1-2023-479-2023 | © Author(s) 2023. CC BY 4.0 License.

ber of discrete zones, segmented solely by their similar texture and adjacency.

The main contributions of this work are twofold: The first is a novel segmentation model based on the individual materials of the different objects in the scene. Second, the subsequent face splitting based on the detected material patches prevents faces from being part of multiple segments.

The remainder of this work is structured as follows. The background and related work is presented in Section 2. Following is the explanation of the proposed method in Section 3. In Section 4, an overview of the used datasets and their results is presented. Finally, the conclusions are presented in Section 5

2. BACKGROUND AND RELATED WORK

2.1 Instance Segmentation

Most of the state-of-the-art mesh segmentation models rely on the conversion to a normalized point cloud to perform the actual segmentation and return an oriented bounding box or point mask (Schult et al., 2022). While these data formats are sufficient for point clouds, these can lead to inconsistent results when converted back to meshes. Meshes require a vertex or face-based segmentation to determine the boundaries between objects (Kaplansky and Tal, 2009).

Existing texture-based segmentation models rely on 2D images of the scene (Zhang et al., 2021). These do provide good and accurate results when the resulting masks are projected on the mesh, but due to the limited coverage of a single image, there are still some drawbacks. The single viewpoint of an image leads to a lot of occlusions which the segmentation model cannot predict. This can lead to either unsegmented parts, or wrongly segmented parts due to projecting through the geometry. To combat these occlusions, multiple images are used that are taken from different angles. While this solves the occlusion problem for the most part, it becomes very difficult to get a consistent segmentation index throughout the set due to variations in angles and lighting conditions. These problems do not arise when trying to segment a texture map, since every face is uniquely represented on the map. It is, however, much more difficult to detect objects in a randomly distributed patchwork of textures.

2.2 Texture Segmentation

Aside from image-based object segmentation, it is also possible to perform segmentation based on different types of textures or materials. Rather than trying to find each instance of a type of texture, these texture segmentation models find all similar patches of a single texture and group them. The field of texture segmentation has been researched thoroughly and already gives good results. Factorisation-based texture segmentation (FTS) (Yuan et al., 2015) can efficiently segment different textures in images. By using local spectral histograms as features, this method relies on singular value decomposition and non-negative matrix factorisation to discriminate region boundaries. The resulting masks provide a great starting point for edge detection.

When trying to find the boundaries of objects in a 2D image, a common technique used is edge detection. It can be performed by a number of performant detection methods like Canny (Ding and Goshtasby, 2001) or Holistic Edge Detection (HED) (Xie and Tu, 2015). While these work great to detect all the edges in an image, that may not always be the end goal. This is where the Factorisation-based texture segmentation can provide a solution with its simple mask representation.

2.3 Mesh Slicing

Cutting meshes using lines can be performed in several ways. Popular methods rely on using planes as a virtual knife (Minetto et al., 2017). By determining which points of the face are in front or behind the section plane, new vertices can be created at the intersecting lines. Using a 2D pixel line however becomes a bit more complex, because it is no longer possible to easily determine which points are in front or behind a pixel line.

Cutting a plane with a line requires a parameterised representation of pixel-based lines. Hough lines (Illingworth and Kittler, 1988) can simplify the detected edges by finding straight lines through the selected pixels. There are a couple of variations, each with distinct advantages. Hough lines are determined by finding a range of pixels that are close to forming a straight line. Hough lines are always looking for lines that cover the entire image, so smaller lines are harder to detect. This is where PHough lines can provide better results. These can detect smaller line segments in an image and provide a start and end point.

2.4 Clustering

Clustering a collection of similar objects can be achieved using different methods like k-means, DBScan or region growing clustering (Xu and Tian, 2015). k-means clustering (Xu and Tian, 2015) finds an optimal grouping with a set amount of clusters with the lowest total point-to-centroid distance. While this method is often used to find a set amount of clusters in a collection, we do not know the amount of needed clusters beforehand. DBSCAN clustering (Xu and Tian, 2015) does not rely on a predefined amount of clusters, rather, it tries to group objects based on their density. Due to its focus on density, it is able to non-linearly separate different clusters.

Existing region growing implementations largely focus grouping elements based on a single parameter like pixel color of surface normal (Fan et al., 2005). Region growing is performed by choosing a random starting seed and making a comparison against neighbouring elements. If those elements are similar, they are grouped together. This method works great to detect connected components, since different objects in a scene could have a similar material, but they should not be part of the same group.

3. METHODOLOGY

The proposed texture based segmentation is performed in a series of steps outlined in figure 2. Where first, the UV texture map is extracted, then the different materials are segmented. After which the edges are extracted, from which the Hough lines can be determined, those are then used to slice the faces of the mesh to create clear boundaries. Finally the texture zones are used to segment the 3D mesh with the use of a region growing algorithm.

To illustrate the full methodology, a simplified textured mesh is used with a clear boundary as seen in fig. 3. The mesh represents a wall-floor connection where the edge between them

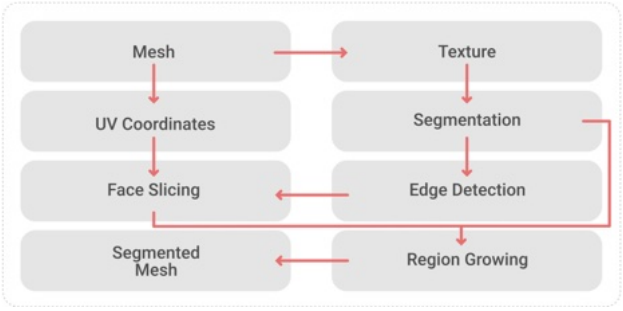

Figure 2: An overview of the proposed workflow.

does not align with any existing geometry. Something which is encountered frequently in large datasets and will be evaluated in the experiments section.

Figure 3: 3D mesh (left) and its corresponding texture map (right)

3.1 Texture Segmentation

The full texture map of the mesh is rarely used to segment the scene, mostly because of the lack of a one-to-one relationship between the object's location in 3D space and on the 2D texture map. This often leads to disjointed parts of a single object spread around on the UV plane. Instead of mainly focusing on the specific object's boundaries, it is much easier to search for patches of similar materials on the texture map. The segmented objects can then be detected by their material instead of geometric shape.

We use factorization-based texture segmentation (FTS) (Yuan et al., 2015) to detect the different materials in the meshes. The algorithm is fine-tuned with a number of parameters: the window size, segmentation number, the omega value and the non-negative constraint. The window size determines the sample area to group textures, for our purpose, a window size of $10px$ was chosen to allow for the most common construction textures to be properly segmented. Fig. 4 shows an overview of the different window sizes and their accuracy. The segmentation number was left at -1 to allow the algorithm to dynamically chose the number of texture segments. The omega value was set to 0.2, to allow it to distinguish between the the different types of homogeneous materials, while still being able to properly group the heavily patterned materials.

FTS is run on a grey-scale image, leading to some false matches between materials. To prevent similar patches of different colours from being grouped, all detected patches $F_i, F_j, ...$ with the same label are compared using the average patch hue $h_{F_i}, h_{F_j}, ...$ When the patch hues are less than a certain threshold t_h apart, they are considered as the same material and grouped as patch F_i' as outlined in eq. 1. For our purpose, a t_h value of 0.1 was chosen, this can split the mesh more accurately and creates more distinct patches.

$$F_i{}' = \left\{ F_i, F_j \middle| \forall F_i, F_j \in \mathbf{F} : |h_{F_i} - h_{F_j}| \leq t_h \right\} \quad (1)$$

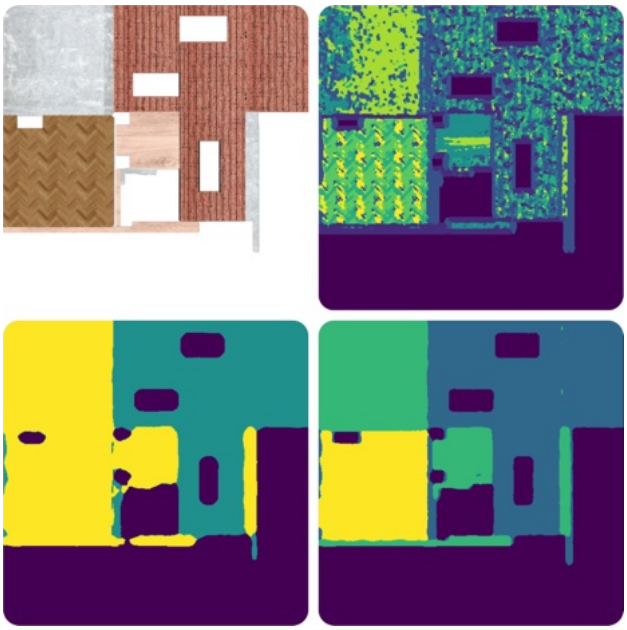

Figure 4: FTS segmentation results: (top-left) the original texture, (top-right) over-segmented texture, (bottom-left) under-segmented texture, (bottom-right) correctly segmented texture.

3.2 Edge Detection

Once the texture boundaries are defined, the edges can be detected. The Canny edge detector (Ding and Goshtasby, 2001) was chosen for its reliability and good results in similar applications. The segmented image is a collection of single-colour zones with very distinct edges, so the edge detector returns all the boundaries with near-perfect accuracy. Figure 5 Shows examples of the detected edges from the segmented texture. A comparison is also made with the original textures to validate the accuracy of the texture boundaries.

It is important to note that due to the texture baking process, a small amount of texture spill can be present on the UV map at the edges of face groups as shown in figure 6. This problem can be avoided by only making use of detected edges that cross existing faces since there is no spill present there.

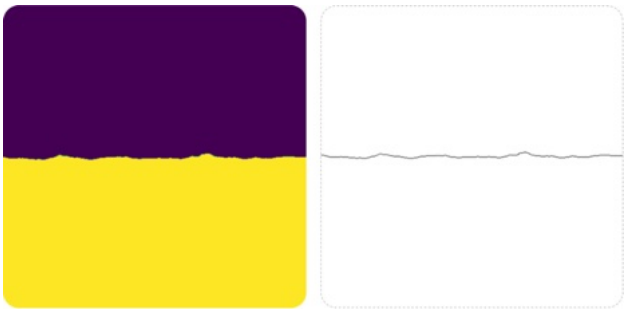

Figure 5: (left) The segmented Image, (right) Canny edge detection results.

The above-mentioned edge detection algorithm provides a binary raster image to represent the edges. Because our end goal is slicing the faces of the mesh with the detected edges, the edges need to be parameterized into straight-line equations. We

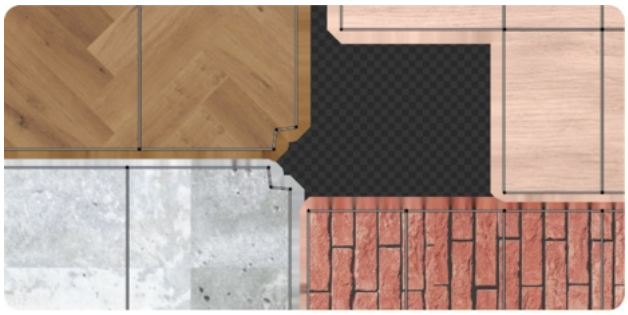

Figure 6: A zoomed in view of texture spill as a result from the texture baking process.

can determine these straight lines by applying a Hough transform detection to the binary edge image (Illingworth and Kittler, 1988). These lines l are defined by their polar coordinates (r_θ, θ)

Because Hough lines are always represented as lines, they cross the whole image. Using the whole line to slice the faces of the mesh would create a large number of unnecessary faces. This is why Probabilistic Hough Line Transforms are chosen for smaller line segments which are defined by a couple of coordinates (x_0, y_0, x_1, y_1). Both methods are compared in fig. 7.

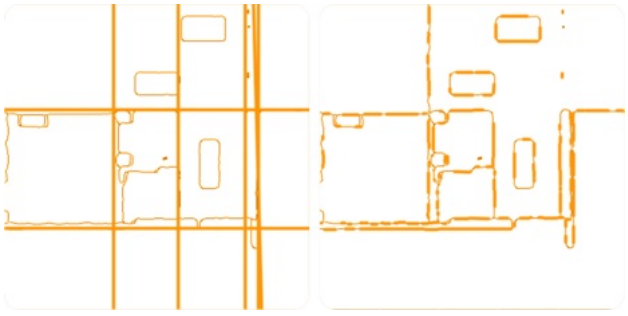

Figure 7: (left) The standard Hough lines, (right) the Probabilistic Hough lines indicated in blue.

3.3 Triangle Mesh Slicing

After detecting the edges, we need to create new triangles at the boundaries of the different materials. To do this, we first project the faces f_i of the mesh onto the UV plane as seen in fig. 8. We then use the detected lines l from the previous step to slice the faces. We define the new faces f_i' with the newly created edge boundaries e_l and assigned them to the appropriate material index i.

The slicing process is performed in a series of steps. First, we check which faces f are intersecting with the lines l, this is performed by a 2D raycast r_l in the direction of the line. Once the intersected triangles f_r are determined we perform a second intersection test, now on a per-edge basis to determine how many edges e are intersecting.

There are 4 cases when checking for an intersection between a triangle and a line as seen in figure 9. Each case creates a different amount of new triangles:

- No intersection: The line and triangle do not intersect, so no slicing is required.
- Point intersection: When the line intersects with one or 2 points and no edges, the original geometry does not need to be altered.

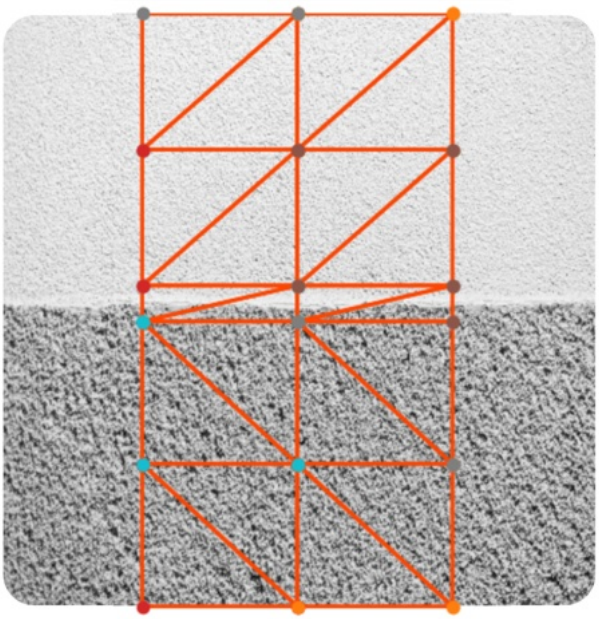

Figure 8: An Image of the projected faces of the mesh on the UV plane. The orange lines indicate the edges of the mesh.

- Point-edge intersection: In this case, the line p intersects the face f_i with one point p_i and the opposite edge e_i. This will result in 2 new triangles f_{i1}, f_{i2}, where the new point p_{ei} is the intersection between l and e_i. Both triangles are then defined by eq. 2

$$\begin{aligned} f_{i1} &= \{p_i, p_{ei}, p_{i1}\} \\ f_{i2} &= \{p_i, p_{ei}, p_{i2}\} \end{aligned} \tag{2}$$

- Edge-edge intersection: When the line intersects with 2 edgese_i, e_j, 2 new points P_{ei}, p_{ej} are created. This results in 3 new triangles f_{i1}, f_{i2}, f_{i3}. The first triangle f_{i1} consists of the 2 new points p_{ei}, p_{ej} and the single existing point from the corner side p_{i1}. The second triangle f_{i2} is created with the 2 new points p_{ei}, p_{ej} and one of the existing edge points p_{i2} on the edge side of the triangle. The last triangle is created with the 2 existing edge points p_{i2}, p_{i3} and one new point p_{ei}. The order of the point allocation is critical to prevent a wrong combination of points. The new triangles are then defined by eq. 3

$$\begin{aligned} f_{i1} &= \{p_{ei}, p_{ej}, p_{i1}\} \\ f_{i2} &= \{p_{ei}, p_{ej}, p_{i2}\} \\ f_{i3} &= \{p_{i2}, p_{i3}, p_{ei}\} \end{aligned} \tag{3}$$

After the new triangles are created on the UV plane as seen in figure 10, they need to be transferred to 3D. Since the new triangles are subdivided from an existing triangle. only the new points need to be interpolated to their new 3D position. This is done though linear interpolation where p_1, p_2 are existing points with 2D coordinates $p_i(x, y)$ and 3D coordinates $p_{i,3D}(x, y, z)$. The new point p_e is only defined by its 2D coordinate $p_e(x, y)$ and its 3D coordinate $p_{e,3D}(x, y, z)$ can be determined according to eq. 4. The new normals $\vec{n_{pe}}$ are determined in a similar way.

$$p_{e,3D} = p_{1,3D} + \frac{p_e - p_1}{p_2 - p_1}(p_{2,3D} - p_{1,3D}) \tag{4}$$

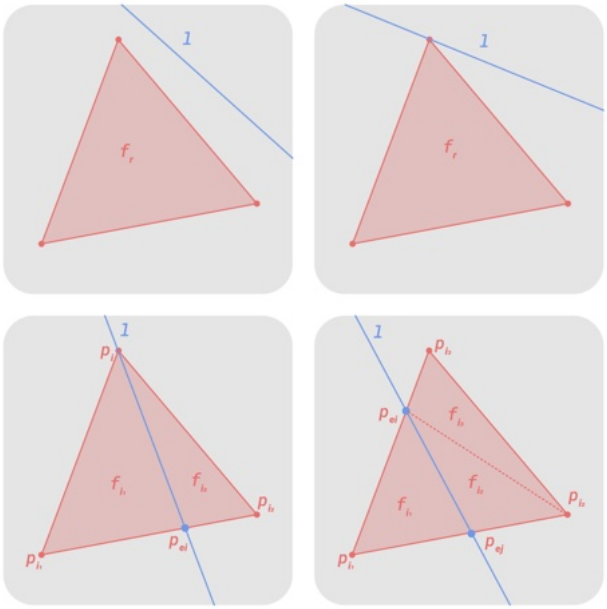

Figure 9: The four cases of a line-triangle intersection. Top-left shows no intersection, top-right shows a point intersection, bottom-left shows a point-edge intersection and bottom-right shows an edge-edge intersection.

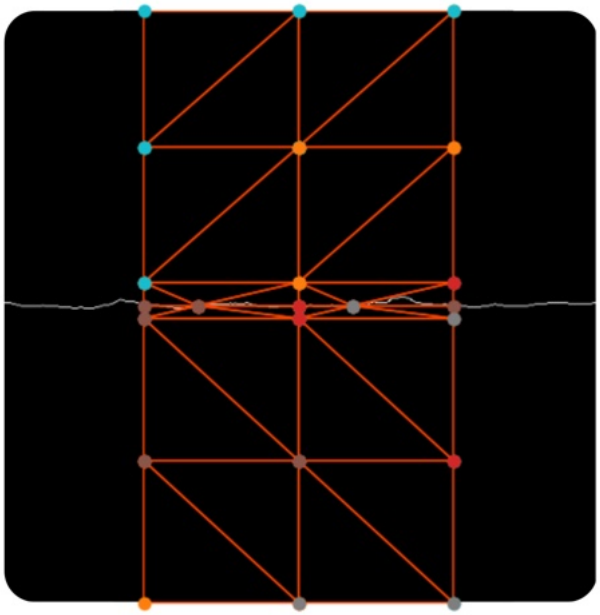

Figure 10: The resulting faces after the pHough lines are used to slice the faces which were overlapping with the lines. The rest of the faces remain unaffected.

3.4 Segmentation

After the new faces are constructed, the segmentation can be performed without any risk of multiple labels on a single face. The segmentation is performed by a region-growing technique where each face is given a segmentation index i. This index is based on the FTS detection from 3.1. Because the segmented patches do not take adjacency into account, if we were to simply group the faces based on that index alone, different objects with the same material would also be grouped. To prevent this, we start by choosing a random face f_i from the mesh and comparing each adjacent face $f_{i,adj}$ for an identical segmentation index i. This process is repeated recursively for each adjacent matching face. Once no more matching faces can be found, the patch is complete and another random, un-grouped, face f_j is chosen.

To prevent over-segmenting small detailed objects with a too sporadic texture that cannot be grouped by FTS, there is also a minimal surface area check. For our purpose, we chose a minimal surface area of $10cm^2$ which allowed our method to group highly irregular textures while still being able to distinguish the most common indoor objects.

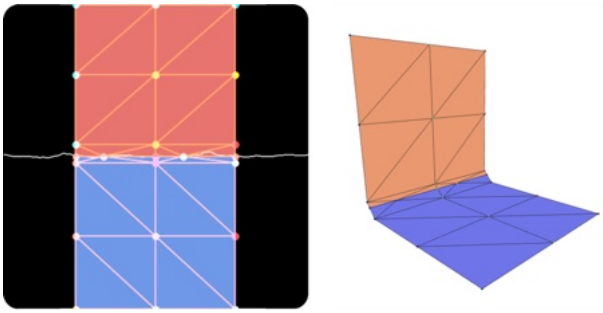

Figure 11: Left: the new faces, each marked with their respective segmentation index based on the texture zone. Right: The resulting segmented mesh.

4. EXPERIMENTS

In this section, we evaluate our segmentation algorithm. First, the datasets and testing metrics are discussed. Then we evaluate the texture boundary detection and instance segmentation separately.

4.1 Datasets and metrics

To evaluate the effectiveness of our proposed method, we conducted experiments on the Matterport (Chang et al., 2017) dataset. This was chosen because of its size and already available metrics and reference studies. The large scenes were subdivided as seen in fig. 12 to increase the processing speed and reduce the complexity. Each zone has +- 200.000 points and 350.00 triangles.

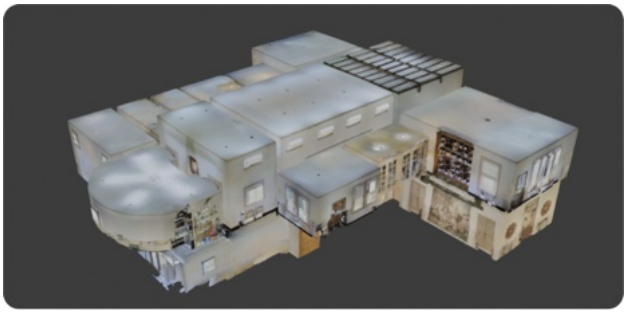

Figure 12: The Matterport dataset mesh

4.2 Texture boundary detection

Because our method heavily relies on clearly defined boundaries between the textures. The textures need to be segmented accurately. The results show this method works well on large surfaces with repeated or small texture detail (fig. 13). This is mostly because of the way the UV maps are laid out. Because large flat surfaces remain connected on the UV map, it becomes easier to segment them.

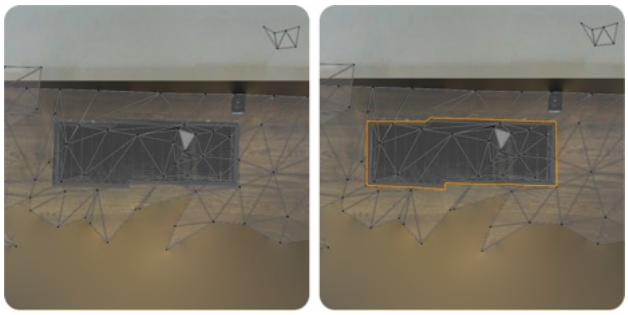

Figure 13: A close up example of a new edge generated by the texture boundary detection. This allows us to clearly assign each face with a unique label

Our method also performs well on large uniformly textured surfaces, like painted walls or floors, something which is also detectable with only edge detection. Our method, however also shows it is capable of segmenting more complex textures like brick walls or wood textures. The newly created edges provide a clear distinction between the different texture zones. Due to the PHough lines, they do not add any useless cuts in irrelevant areas. Fig. 14 shows a good result on a largely uniform texture.

Figure 14: Wall detail. (top-left) uv-texture, (bottom-left) original mesh, (top-right) new detected edges, (bottom-right) sliced mesh

Some patterns are hard to separate due to subtle colour shifts in the image. This causes inconsistent boundaries in the detected zones. The method also struggles with very dense texture maps, where a lot of the faces are laid out separately and do not cover a lot of surface area. fig. 15 shows our method struggling with the clear boundaries that are still detected despite the texture segmentation. The irregular large patterns are not detected as a single texture, so the lines are still present in the final detected lines.

4.3 Instance segmentation

The results of our method are outlined below. From the results acquired in subsection 4.2 it became clear this method would perform well on simple scenes where objects are largely composed of single materials as seen in fig. 16.

Figure 15: Carpet detail. (top-left) uv-texture, (bottom-left) original mesh, (top-right) new detected edges, (bottom-right) sliced mesh

Figure 16: Door detail. (top-left) uv-texture, (bottom-left) original mesh, (top-right) new detected edges, (bottom-right) sliced mesh

The method can detect geometry-less details on flat surfaces, something more traditional segmentation models struggle with as seen in fig. 17.

The method still over-segments on highly varied objects. Finding the balance between over-segmentation and mismatching has proven tricky and has to be evaluated on a scene by scene basis.

5. CONCLUSIONS

In this paper, we proposed a novel method for refining building meshes using texture-based separation. Our method leverages texture segmentation to segment the different materials in the

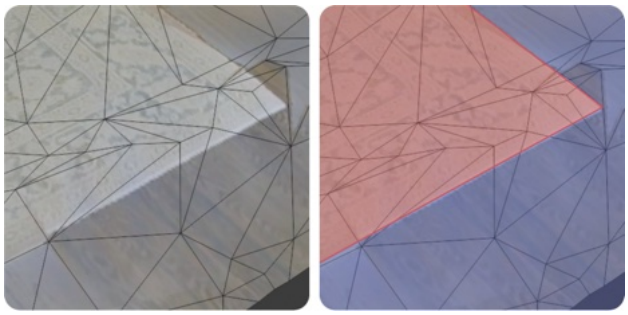

Figure 17: A segmented example of a element that is only present in the texture and not in the geometry

mesh and create new triangles at the boundaries of the materials to refine the mesh. We also proposed a novel object segmentation method that uses region-growing to group faces based on their common texture. Our method outperformed other state-of-the-art mesh-based and point-based segmentation methods on meshes with sparse geometric detail and high texture detail. However, the method still performs worse on non-inform objects. This method could be valuable as a supplementary segmentation model to increase the overall accuracy. Future work could include improved ways of unwrapping the textured meshes for better grouping of similar faces.

ACKNOWLEDGEMENTS

This project has received funding from the FWO-SB grant (grant agreement 1S16923N) and the Geomatics research group of the Department of Civil Engineering, TC Construction at the KU Leuven in Belgium.

REFERENCES

Bassier, M., Vergauwen, M., Poux, F., 2020. Point cloud vs. mesh features for building interior classification. *Remote Sensing*, 12.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., Zhang, Y., 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. http://arxiv.org/abs/1709.06158.

Ding, L., Goshtasby, A., 2001. On the Canny edge detector. *Pattern Recognition*, 34, 721-725. https://www.sciencedirect.com/science/article/pii/S0031320300000236.

Fan, J., Zeng, G., Body, M., Hacid, M. S., 2005. Seeded region growing: An extensive and comparative study. *Pattern Recognition Letters*, 26, 1139-1156.

Geyter, S. D., Vermandere, J., Winter, H. D., Bassier, M., Vergauwen, M., 2022. Point Cloud Validation: On the Impact of Laser Scanning Technologies on the Semantic Segmentation for BIM Modeling and Evaluation. *Remote Sensing*, 14.

Heckbert, P. S., 1986. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6, 56-67. http://ieeexplore.ieee.org/document/4056764/.

Illingworth, J., Kittler, J., 1988. SURVEY A Survey of the Hough Transform. *COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING*, 44, 87-116.

Kaplansky, L., Tal, A., 2009. Mesh Segmentation Refinement. *Computer Graphics Forum*, 28, 1995-2003.

Mao, J., Shi, S., Wang, X., Li, H., 2022. 3D Object Detection for Autonomous Driving: A Comprehensive Survey. http://arxiv.org/abs/2206.09474.

Minetto, R., Volpato, N., Stolfi, J., Gregori, R. M., da Silva, M. V., 2017. An optimal algorithm for 3D triangle mesh slicing. *CAD Computer Aided Design*, 92, 1-10.

Schult, J., Engelmann, F., Hermans, A., Litany, O., Tang, S., Leibe, B., 2022. Mask3D for 3D Semantic Instance Segmentation. http://arxiv.org/abs/2210.03105.

Xie, S., Tu, Z., 2015. Holistically-Nested Edge Detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1395-1403.

Xu, D., Tian, Y., 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2, 165-193. http://link.springer.com/10.1007/s40745-015-0040-1.

Yuan, J., Wang, D., Cheriyadat, A. M., 2015. Factorization-Based Texture Segmentation. *IEEE Transactions on Image Processing*, 24, 3488-3497.

Zhang, Y., Sidibé, D., Morel, O., Mériaudeau, F., 2021. Deep multimodal fusion for semantic image segmentation: A survey. *Image and Vision Computing*, 105.