On-Mesh Bilateral Filtering

Bridging the Gap Between Texture and Object Space

Mihnea Bernevig[1]

Supervisors: Prof. Dr. Elmar Eisemann[1]**, Mathijs Molenaar**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Mihnea Bernevig
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Elmar Eisemann, Mathijs Molenaar, Jing Sun

Abstract

Traditional bilateral filters, effective in 2D image processing, often fail to account for the 3D structure of meshes, leading to artifacts in texture filtering. This thesis introduces On-Mesh Bilateral Filtering, a novel method that adapts the bilateral filter to work with non-contiguous texture mappings by incorporating 3D spatial distances and face adjacency information into the filtering process. The On-Mesh Bilateral Filter combines mesh surface sampling techniques with heat geodesic distance calculations to create a geometry-aware kernel that achieves more accurate and context-sensitive smoothing operations, respecting both the mesh topology and texture space properties. This paper hopes to encourage further research in the area of geometry-aware texture filters.

1 Introduction

Textures are crucial in digital graphics, particularly for 3D models used in gaming, film, and virtual reality. They are essentially images or patterns applied to the surface of a 3D model to give it color, detail, and a sense of material. Textures enhance the visual realism and detail of 3D objects, making them appear more lifelike and engaging by simulating complex surface qualities such as roughness, glossiness, and transparency. Unlike simple surface texturing, 3D models require textures to be mapped through a dedicated texture space. This non-contiguous texture mapping presents significant challenges, where adjacent points on the mesh can correspond to distant points in the texture space, and points that are close in the texture image might map to distant parts on the mesh.

Traditional bilateral filters, which are designed for 2D image processing, focus solely on signal intensity and spatial proximity. When these traditional methods are applied to texture images, the result can often be artifacts or uneven texture filtering due to the disregard of the mesh's 3D structure.

This paper introduces On-Mesh Bilateral Filtering, a novel method that adapts the bilateral filter for use with non-contiguous texture mappings and accounts for texture warping. Our approach modifies the traditional bilateral filter to incorporate 3D spatial distances, crafting a geometry-aware kernel.

This geometry-aware kernel facilitates more accurate and context-sensitive smoothing operations, effectively respecting both the topological layout of the mesh and the inherent properties of the texture space. Current texturing techniques enable artists to paint directly onto a 3D model. In this scenario, an artist may wish to apply effects, such as a blur filter, to the painted content. However, existing methods usually overlook the 3D mesh structure, resulting in inaccurate outcomes. The On-Mesh Filtering method presented in this paper addresses this issue, ensuring proper application of filters.

The implementation of On-Mesh Bilateral Filtering could support texture application by enabling a more intuitive and direct workflow, thereby reducing the time and complexity involved in preparing 3D models for use.

This paper is structured as follows: The Background section summarizes the necessary prior knowledge, including explanations of the bilateral filter, 3D mesh components, and texture mapping. The Problem Description section outlines the challenges in developing the On-Mesh Bilateral Filter. The Methodology section details the algorithm and implementation of the On-Mesh Bilateral Filtering. The Results section presents the findings from applying the filtering method, comparing it with traditional methods, and discussing performance. Finally, the Responsible Research and Conclusions and Future Work sections address the transparency of the research process and potential areas for future investigation.

2 Background

The following section will provide a summary of the prior knowledge necessary for grasping the content of this paper. The purpose and workings of the bilateral filter will be explained, followed by an outline of the components of a 3D mesh, as well as a brief description of texture mapping of 3D meshes.

2.1 The Bilateral Filter

The bilateral filter is a type of edge-preserving, non-linear, noise-reducing filter [1]. It achieves its edge-preserving quality by considering both the distance between pixels as well as the difference in their intensity. This filtering method was originally developed by Tomasi and Manduchi, in 1998, for denoising images while retaining the sharpness of features [2]. The bilateral filter can be described by the following formula:

$$I_{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) \cdot f_r(\|I(x_i) - I(x)\|) \cdot g_s(\|x_i - x\|)$$

where:

- $I_{\textbf{filtered}}(x)$ is the intensity of the filtered image at location x.
- Ω represents the neighborhood around x.
- $I(x_i)$ is the intensity of the input image at the location x_i.
- f_r is the range kernel for smoothing differences in intensities. This is typically a Gaussian function which depends on the intensity difference.
- g_s is the spatial kernel for smoothing differences in coordinates, also typically a Gaussian function.
- W_p is a normalization factor defined as $W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) \cdot g_s(\|x_i - x\|)$, which ensures the filter weights sum to one.

The Gaussian function, often referred to as a normal distribution when used in probability theory, is defined by the formula:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where:

- μ is the mean or expectation of the distribution (and also its peak).
- σ is the standard deviation, which determines the width of the bell curve.
- σ^2 is the variance.

The inclusion of pixel intensity into the weight calculation allows for the blurring of areas with similar intensity while preserving the sharp boundaries where there are significant intensity differences. This feature is particularly valuable in maintaining the distinct edges and textures of the subject, preventing the blending of distinct features into the background.

2.2 3D Meshes

A 3D mesh is a geometric data structure used in computer graphics to represent the surface of a 3D object as a collection of vertices, edges, and faces. In a mesh:

- A **vertex** is a point in 3D space, characterized by its coordinates (x, y, z).
- An **edge** is a line segment connecting two vertices.
- A **face** is a flat surface bounded by edges, typically triangles or quadrilaterals in most meshes.

Meshes are central to the field of computer graphics and are crucial for the simulation of realistic 3D environments and objects in virtual and augmented reality, video games, and CGI in films.

2.3 Texture Mapping of 3D Meshes

Texture mapping is the process of applying a 2D image, or texture, to the surface of a 3D mesh. This technique enhances the visual detail of a 3D model without increasing its geometric complexity. The key to effective texture mapping is the creation of a UV map, which is a flat representation of the surface of the 3D model:

- A **UV map** assigns each vertex in the faces of a mesh a coordinate in the 2D texture space (u, v), which corresponds to a position in the texture image. This will then create a mapping of the faces onto the texture space.
- These UV coordinates ensure that the texture aligns correctly with the model, allowing models to appear more detailed and realistic.
- The shape of the face on the 3D model and the shape of the face mapping onto the texture do not necessarily correspond.

The proposed On-Mesh Bilateral Filtering method addresses the gap between texture and object space by integrating mesh geometry and texture information into a cohesive filter.

3 Problem description

In this section, we delve into the specific challenges involved in developing the On-Mesh Bilateral Filter. As explained in the previous sections, the bilateral filter functions by computing a weighted average over the neighbors of a pixel. This weight then depends on the spatial distance between the pixel and its neighbor, as well as on the difference in intensity between the two. Therefore, to create the On-Mesh Bilateral Filter, we need to answer **three questions**:

1. How can we determine the neighbors of a texel?
2. How can we calculate the spatial distance between a texel and its neighbor?
3. How can we calculate the intensity difference between a texel and its neighbor?

Answering **question 3** is trivial; we can simply take the intensity values directly from the texture image and apply the same weight function.

Question 2, concerning the calculation of spatial distance, initially seems as though it might be resolved using simple Euclidean distance. However, upon closer examination, it becomes evident that a more complex distance calculation is required, as we need to consider the distance traveled on the surface of the mesh. The exact reasoning behind this is best described with an example:

Think of marking two points on a piece of paper, one at the header of the paper and one in its footer, the distance between these two points is the length of a straight line drawn between them. Now, imagine folding this piece of paper at the halfway mark between the two points, bringing them closer together in Euclidean space. From an image processing perspective, this shift should not have any effect on the influence the two points would have on one another, but using Euclidean distance for this calculation would misrepresent this. The true distance between these two points should still be considered as the length of the line drawn between them when the paper is flat.

This kind of distance calculation is referred to as geodesic distance. Several methods to compute this distance exist, such as Fast Marching [3] or the algorithm by Mitchell et al. for computing exact polyhedral distance [4]. This project will use Crane's Heat Geodesic Distance algorithm [5]. This algorithm was chosen for its ability to compute geodesic distances rapidly, maintaining comparable accuracy to other methods. It is important to note that this algorithm is significantly faster only if the input mesh or structure remains unchanged, as it performs a set of precalculations that then allow it to quickly compute geodesic distances from any source point to all other points.

Question 1 can seemingly be addressed by mapping the centers of all texels to corresponding points on the surface of the mesh. The neighbors of a texel are then defined as those whose centers fall within a specified kernel, meaning that the points are geodesically closer than a predefined maximum distance. Figures 1 and 2 show a texture image with UV mapping and the mesh the texture is mapped to respectively. In Figure 2, texel centers have been marked in red. These figures show the inherent issue with this approach: it does not account for the change in texel density.

A better sampling method would generate sample points uniformly across the entire surface of the mesh. To take advantage of the speed-up provided by the Heat Geodesic Distance algorithm, these samples should be generated once for the whole mesh, rather than for each texel center. The algorithm for Bridson's algorithm for Fast Poisson Disk Sampling in Arbitrary Dimensions [6] provides such a sampling

method, with the caveat that the number of samples on any face should be greater or equal to the texel density of the face. In short, each texel should be sampled at least once, with texel density being kept constant across the entire mesh. Figure 3 shows the mesh along with the samples generated by the Poisson Disk Sampling algorithm. Although the sample count is higher in this instance, it is important to see that sample density remains consistent across the entire mesh, although the underlying texture may have been stretched.

Figure 4 shows an example of a dense point cloud generated with Fast Poisson Disk Sampling along with a visualization of geodesic distance from a source point.

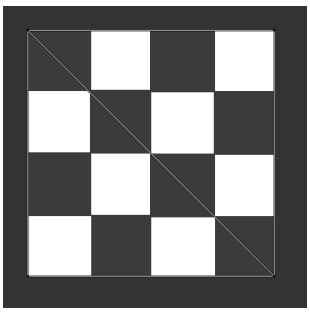


Figure 1: 4px by 4px image with UV mapping of a 2 face planar mesh.

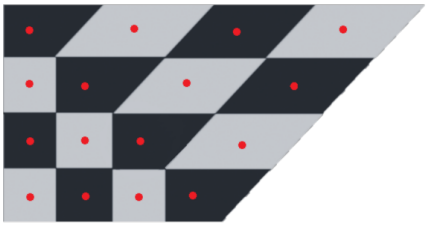


Figure 2: Top-down view of a 2 face planar mesh, with stretched texture. Texel centers are marked in red.

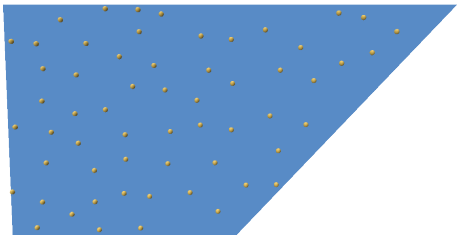


Figure 3: Top-down view of a 2 face planar mesh. Samples generated by Poisson Disk Sampling algorithm marked in yellow. Minimum distance between samples set to 0.1 units.

4 Methodology

The following section will describe the algorithm and implementation of On-Mesh Bilateral Filtering. The section is divided into three main parts: an overview of the algorithm, a detailed description of the implementation, and an overview of the testing data utilized to evaluate the efficacy of the method.

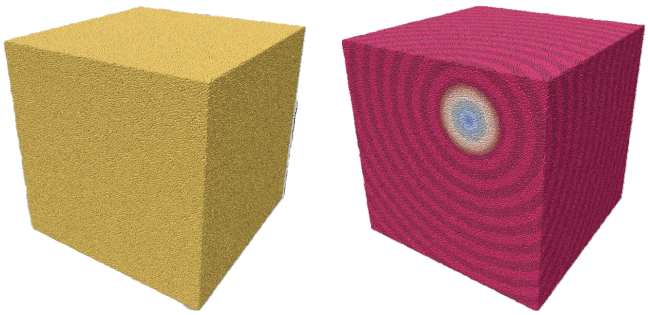
4.1 Algorithm Overview

The algorithm can be split into three main parts, namely:

1. Mesh sampling: Generating uniform samples across the surface of the mesh, with sample density being kept constant across all mesh faces.
2. Texel position mapping: For each texel, map its center to the surface of the mesh. These will be used as sources in the following step.
3. Application of the Bilateral Filter: For each of the previously generated sources, calculate geodesic distances to sample points, calculate intensity differences, and apply the bilateral filter.

4.2 Implementation Details

The implementation of the On-Mesh Bilateral Filtering was carried out using C++ with the Geometry-Central [7] library for mesh sampling and geodesic distance calculation, Polyscope [8] for UI and rendering, and the OpenCV [9] library for image processing operations. The current implementation only works with manifold meshes, as it is a requirement of Geometry-Central's Poisson disk sampling algorithm implementation.



(a) Point Cloud generated from Cube mesh. 98065 sample points, minimum distance between samples of 0.006 units

(b) Geodesic Distance visualization on Point Cloud. Points beyond 0.28 units are marked in deep red

Figure 4: Comparison of Point Cloud visualizations. (a) shows the Point Cloud generated from a Cube mesh with detailed sample information, and (b) shows the Geodesic Distance visualization with points beyond 0.28 units marked in deep red.

Mesh Sample Generation

The Poisson Disk Sampling[1] algorithm, implemented in Geometry-Central, generates samples on the surface of a mesh with a minimum geodesic distance specified by the user. This algorithm ensures that the samples are evenly distributed, avoiding clusters and gaps.

After generating the samples, the UV coordinates for each sample point are determined based on the UV mapping of the

[1] https://geometry-central.net/surface/algorithms/surface_sampling/

mesh face where each sample landed. This results in a point cloud with each sample point tied to its corresponding UV coordinates.

To ensure that enough sample points are generated to sample each texel at least once, the user should provide a minimum distance small enough that the minimum density of sample points on a face is greater or equal to the maximum density of texels on any face of the mesh.

Subfigure 4a shows the point cloud generated after sampling a simple cube mesh, with extreme vertices at $(-1, -1)$ and $(1, 1)$. Using a minimum distance between samples of 0.006 units creates around 98 thousand sample points, with a consistent sample density on all faces of the mesh.

Texel Position Mapping

As mentioned in Section 2, mesh faces are mapped to a texture image using UV coordinates to ensure accurate correspondence between the 3D mesh and the 2D texture. The texels that will appear on the mesh can then be found by calculating bounding boxes from the UV coordinates of each face.

Bounding boxes are generated by examining the UV coordinates of each mesh face. For each face, the minimum and maximum UV coordinates are determined, forming a rectangle (bounding box) that encompasses the face. This rectangle ensures that all texels within the face's area are considered during processing.

Within each bounding box, the algorithm iterates over the texels and checks if they lie within the triangle formed by the UV coordinates of the mesh face. If a texel is within a triangle, the corresponding 3D position on the mesh is calculated. This 3D position will then be used as a **source point** in the following step of the algorithm.

The fact that all texels can be processed independently can be leveraged to increase performance by utilizing multithreading. Each bounding box can be handed to a thread to be processed, and then all results combined to form the final texture image.

In cases where bounding boxes have a very large area, or there are not enough bounding boxes to occupy all threads, the bounding boxes are subdivided to ensure work is fairly split. When dividing a bounding box, the algorithm splits it either vertically or horizontally, depending on which dimension is larger. This process ensures that no thread does a disproportionate amount of work while the others are idling.

Application of the Bilateral Filter

For each of the **source points** computed in the previous step, the closest point in the point cloud is identified. Next, the geodesic distance from this closest point to all other sample points on the mesh is calculated. This is done using Geometry-Central's implementation of the Heat Geodesic Distance calculation for Point Clouds.[2]

For each sample point, the intensity is retrieved from the texture image using the UV coordinates computed earlier. The weights for both spatial proximity and color similarity are calculated. The spatial weight is based on the geodesic distance, while the range weight is determined by the color difference between the texel and the current pixel.

These weights are used to apply the intensity change. The pixel color is weighted by both spatial and color weights and added to an accumulated color value. The total weight is also updated to reflect the contribution of this pixel.

Finally, after considering all relevant points, the accumulated color is divided by the total weight to compute the final color for the texel. The resulting color values are clamped to ensure they fall within valid limits, preventing any out-of-range values.

This process is repeated for each source point until all texels mapped to the mesh have been filtered.

Subfigure 4b shows a visualization of the geodesic distance calculation from a source point. Points marked in red are beyond the maximum distance specified by the user and will not be considered for the filtering process.

5 Results and Discussion

As part of this project, both Bilateral and Gaussian filters have been implemented, with the Gaussian filter being a component of the Bilateral filter. This section will present and discuss the results of applying both Bilateral and Gaussian filtering using the algorithm described in Section 4. Some of the results that will be discussed will feature the output of the Gaussian filter, as those results much more clearly show the benefits of using the On-Mesh Filtering technique, rather than simply applying traditional filtering.

5.1 Testing Data

As previously mentioned, the amount of mesh samples generated needs to increase along with texture image resolution. An increase in mesh sample count translates to an increase in the time required to compute the geodesic distances required for the filtering process.

This has made it impractical, especially in the short time frame of the project, to use high-resolution textures for testing. Test data, therefore, consisted of low-resolution textures, with images of at most 300 x 300 px resolution, and various meshes found on the Sketchfab repository [10], as well as meshes and textures created by the researcher.

All authors of both the meshes and textures can be found credited in the references section and wherever their assets were used.

5.2 Filtering Output

In this section, we explore the results of applying various filtering techniques to 3D meshes. The primary focus is on demonstrating the differences in performance and visual quality between traditional Gaussian blur, On-Mesh Gaussian Filtering, and On-Mesh Bilateral Filtering. The traditional Gaussian filter implementation used for the comparison images is the one available in the GNU Image Manipulation Program (GIMP) [11].

Filtering a Planar Mesh

Applying On-Mesh Gaussian or Bilateral Filtering to a texture mapped onto a planar mesh with the same width-to-height ratio as the texture image yields results similar to tra-

[2] https://geometry-central.net/pointcloud/algorithms/heat_solver/

ditional filtering methods. This is because a planar mesh that preserves the texture image's aspect ratio does not introduce any new features that the filter needs to account for. An example of such a mesh can be seen in Figure 5a. However, if, as in Figure 5b, the mesh is stretched or compressed, distorting the texture, the On-Mesh filter will produce a different output.

(a) Square mesh that preserves the texture image aspect ratio.

(b) Stretched mesh that distorts the texture image.

Figure 5: Comparison of square and stretched meshes in relation to texture image preservation. The square mesh maintains the original aspect ratio of the texture image, while the stretched mesh distorts it.

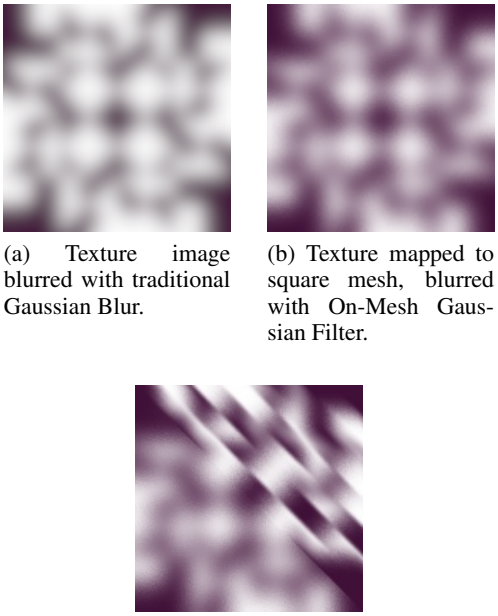
Figure 6 demonstrates the output differences when using different meshes as input, as well as a comparison of output between traditional Gaussian filtering and On-Mesh Gaussian filtering. Differences between Figure 6a and Figure 6b, traditional Gaussian and On-Mesh Gaussian respectively, are minimal, aside from a slight color difference that the traditional Gaussian filter introduced. Figure 6c demonstrates the output difference when a mesh stretches the applied texture, such as in Figure 5b. More detail is preserved where the texel area was increased, while the level of detail drops towards the top left corner, where texels were instead squished and distorted.

Filtering a Cube Mesh

When applying Gaussian blur over an image, one can observe a bleeding effect across the edges defined by the image colors. On a cube mesh with differently colored faces, it would be natural to observe this effect of color seeping over all edges of the face. Applying a Bilateral blur would then remove this color bleeding across the edges.

As a first example, the texture shown in Figure 7 has been mapped to a cube mesh. Larger versions of the images discussed in this example can be found in Appendix A.2, along with the parameters used for generating the output images. The de-noising capability of both the On-Mesh Gaussian and Bilateral filters appears to be on par with that of the traditional Gaussian filter on this particular image.

Figure 8 shows a comparison of the results of applying different filtering techniques. Figure 8a shows the mesh with the original noisy texture. Figure 8b shows the mesh and texture after a traditional Gaussian filter has been applied. Expected color bleeding can be seen between the orange and blue faces, but it can also be noticed that the white background is bleeding between the blue and green faces. Since traditional Gaussian filtering ignores the geometry of the model, the blue and



(a) Texture image blurred with traditional Gaussian Blur.

(b) Texture mapped to square mesh, blurred with On-Mesh Gaussian Filter.

(c) Texture mapped to a stretched mesh, blurred with On-Mesh Gaussian Filter.

Figure 6: Comparison of traditional Gaussian blur and On-Mesh Gaussian Filter applied to a planar mesh and a stretched mesh.

green faces do not influence one another. In contrast, Figure 8c shows the expected filtering behavior, with color bleeding between all faces of the cube. Applying On-Mesh Bilateral Filtering would remove this color bleeding and produce a denoised texture, as in Figure 8d.

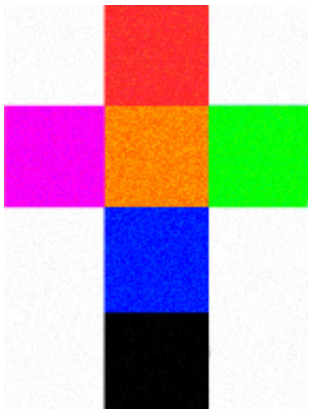
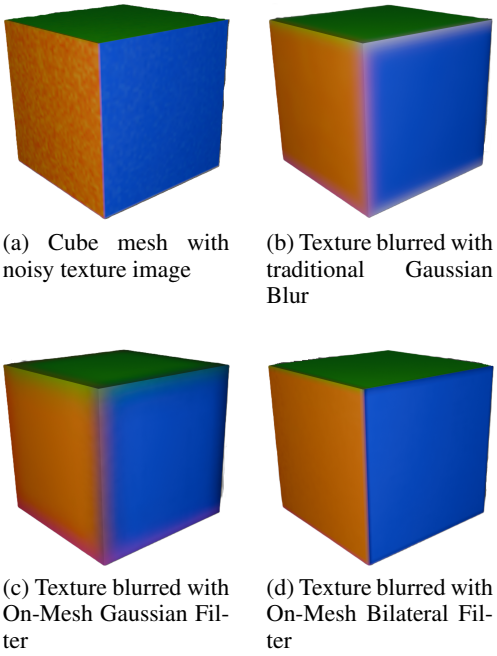


Figure 7: Noisy texture of cube mesh. 144 x 192 px.

Filtering a Complex Mesh

The output of filtering on a more complex mesh can be seen in Figure 9. This mesh and texture were created by Onekro, with the model being titled Chest Pixelart [12].

Meshes can be composed of multiple disconnected substructures, as in the case of the Chest Pixelart. Since this



(a) Cube mesh with noisy texture image

(b) Texture blurred with traditional Gaussian Blur

(c) Texture blurred with On-Mesh Gaussian Filter

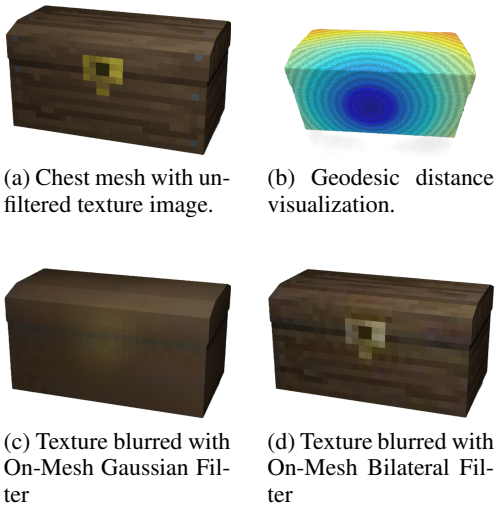
(d) Texture blurred with On-Mesh Bilateral Filter

Figure 8: Comparison of different filtering techniques applied to a noisy texture image mapped to a cube mesh.

algorithm stores generated samples in a Point Cloud without considering the surface from which they were sampled, it may treat two disconnected substructures as connected if they are very close to each other, even if they do not touch. This can also happen in cases where two faces of the same structure sit very close to each other, without actually being connected.

In the case of the Chest Pixelart mesh, seen with its original texture image in Figure 9a, the lid of the chest and the actual chest body are such disconnected structures. Figure 9b demonstrates that although the source point for geodesic distance calculation is on the bottom structure, points on the top structure are not perceived as a separate cluster. This is beneficial in this example, as it would be expected that the blurring effect is continuous across the outer side of the chest since they represent the same material. Based on the particular mesh used as input, however, this might produce unintended results, such as when the arms of a 3D character sit very close to their body. This can easily be fixed by distancing the affected areas from one another.

Nevertheless, the filtering method described in this paper produces the expected output for this mesh, with the output of Gaussian filtering seen in Figure 9c and that of the Bilateral filter in Figure 9d. A particular feature of this mesh is that its bottom structure is bucket-shaped, as the chest has an inner volume where objects might be stored. The Point Cloud Heat Geodesics algorithm handles this feature well, with the shortest path to the inside volume from the outside being over the lip of the chest, rather than simply jumping through the chest's side.



(a) Chest mesh with unfiltered texture image.

(b) Geodesic distance visualization.

(c) Texture blurred with On-Mesh Gaussian Filter

(d) Texture blurred with On-Mesh Bilateral Filter

Figure 9: Visualization of filtering output and geodesic distance calculation on Chest Pixelart mesh. Mesh and textures created by Onekro.

5.3 Observations

The On-Mesh Gaussian and Bilateral Filter implementations perform as expected and have similar de-noising capabilities as their traditional counterparts, provided a sufficient number of mesh samples are generated. The sampling approach correctly adapts to the mesh geometry, and the geodesic distance calculation ensures accurate spatial weights are used in the filtering process. The On-Mesh Gaussian filter exhibits the behavior that is expected, with color bleeding across edges of differently colored mesh faces. The On-Mesh Bilateral filter correctly removes this artifact, preserving hard edges across the texture image.

The performance of the filter implementation is limited, however, as, to ensure accurate filtering behavior, a large number of mesh samples have to be generated. This number of mesh samples required grows considerably as texture resolution increases, making it impractical for this implementation to be used for meshes that use a high-resolution texture. This limitation is not present in the aforementioned traditional 2D image filters, as each of their kernel's samples has the same size. On-Mesh filtering presented this challenge in that texels, although having the same sizes on the original texture image, could have greatly varying sizes after being mapped on the mesh, with users expecting that texels with greater surface area on the mesh would be less blurred or have a greater influence on the rest of the texture image. The approach discussed in this paper trades performance to ensure an accurate and expected filtering output.

6 Responsible Research

The source code used in this research is publicly available and can be accessed by navigating to the author's GitHub. [13] The repository includes all scripts, libraries, and dependencies required to replicate the results presented in this paper. For ease of access, the code is organized into directo-

ries based on functionality and accompanied by a detailed README file that provides instructions for both installation and usage.

The author believes that transparent code is essential for the reproducibility and verification of research findings. Therefore, the source code used to generate results for this research contains clear and understandable comments on all relevant methods. Furthermore, all test data has been collected from publicly available repositories, and the exact parameters used when generating results seen in this paper can be found in Appendix A. This approach ensures that the research process is fully transparent and that others can accurately replicate and verify the findings.

7 Conclusions and Future Work

The research presented in this paper explores the development and implementation of On-Mesh Bilateral Filtering, a novel approach designed to bridge the gap between texture and object space. Traditional filtering methods, while effective in 2D image processing, fall short when applied to textures mapped onto 3D meshes due to their disregard for the underlying geometric structure of the mesh. By incorporating both 3D spatial distances and face adjacency information, the On-Mesh Bilateral Filter offers a geometry-aware solution that enhances the accuracy and visual quality of texture filtering on 3D models.

The On-Mesh Bilateral Filtering method presented in this research provides a robust framework for improving texture filtering on 3D meshes. While challenges remain, particularly in optimizing performance for high-resolution textures, the findings of this study lay a solid foundation for future advancements in this field. Continued research and development will be essential to fully realize the potential of On-Mesh filtering techniques and their applications in digital graphics.

7.1 Limitations

One of the main limitations of this research project was the great computational power required for testing the filter implementation on textures with high texel resolutions. Even with the performance improvements achieved by Crane [5] through his Heat Geodesic Distance algorithm, computing geodesic distances on a dense point cloud for each texel in images with over 3 million pixels is currently unfeasible with the existing filter implementation.

To the author's knowledge, there is a notable lack of previous research in the area of model geometry-aware texture filters. This absence of prior work meant there were no established guidelines or methods to follow for developing an effective sampling strategy. Consequently, a significant portion of the research time was spent exploring sampling methods that ultimately proved to be unproductive.

The short time frame for completing the project further compounded these limitations, as CSE3000 imposed a strict ten-week deadline for project completion. This constrained timeframe limited the scope of the research and the extent to which potential solutions could be explored and refined.

7.2 Further Research

Future research on this particular topic can focus on improving the performance of the On-Mesh Bilateral Filter. The main performance limitations come from the high number of samples that need to be generated, as well as from the fact that the entire sample set must be taken into account when calculating geodesic distance

A possible approach that may achieve this would be to only compute the geodesic distance calculation for a subset of all the samples generated, and instead interpolating the distance to all other samples using triangulation techniques. This could allow for an increase in sample density while simultaneously reducing computation time, leading to a both more accurate and faster method.

Another method that could increase performance could involve creating smaller point clouds out of the entire sampled set. One could, for each face of the mesh, gather all the points in the cloud that are interesting for the entire face, then compute the geodesic distance calculation for the subset of samples that were gathered. This subset could be generated by taking random samples on the edges of each face, calculating the Euclidean distance from these samples to all other faces, and including all mesh sample points that fall within a certain distance. While this method could improve performance, it will also increase the number of precomputations required for the heat geodesic distance calculation, meaning that whether this will increase overall performance is uncertain.

In conclusion, while the project made strides in exploring On-Mesh Texture Filtering methods, future research should focus on optimizing performance, particularly in handling high-resolution textures. Additionally, developing more efficient sampling methods and leveraging advancements in computational techniques could address the current limitations and pave the way for more practical implementations.

References

[1] F. Banterle, M. Corsini, P. Cignoni, and R. Scopigno, "A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain," *Computer Graphics Forum*, vol. 31, no. 1, pp. 19–32, February 2012. [Online]. Available: http://vcg.isti.cnr.it/Publications/2012/BCCS12

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 839–846.

[3] R. Kimmel, "Fast Marching Methods for Computing Distance Maps and Shortest Paths," *Lawrence Berkeley National Laboratory*, 2 1996. [Online]. Available: https://escholarship.org/content/qt7kx079v5/qt7kx079v5.pdf?t=p21n7l

[4] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM journal on computing*, vol. 16, no. 4, pp. 647–668, 8 1987. [Online]. Available: https://doi.org/10.1137/0216045

[5] K. Crane, C. Weischedel, and M. Wardetzky, "The heat method for distance computation," *Commun. ACM*,

vol. 60, no. 11, pp. 90–99, Oct. 2017. [Online]. Available: http://doi.acm.org/10.1145/3131280

[6] R. Bridson, "Fast Poisson disk sampling in arbitrary dimensions," *University of British Columbia*, 8 2007. [Online]. Available: https://doi.org/10.1145/1278780.1278807

[7] N. Sharp, K. Crane *et al.*, "Geometrycentral: A modern c++ library of data structures and algorithms for geometry processing," https://geometry-central.net/, 2019.

[8] N. Sharp *et al.*, "Polyscope," 2019, www.polyscope.run.

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[10] Sketchfab, "Sketchfab - The best 3D viewer on the web." [Online]. Available: https://sketchfab.com/

[11] "GIMP." [Online]. Available: https://www.gimp.org/

[12] Onekro, "Chest pixelart," 2021, licensed under Creative Commons Attribution-NonCommercial (http://creativecommons.org/licenses/by-nc/4.0/). [Online]. Available: https://skfb.ly/onK9F

[13] M. Bernevig, "On-mesh bilateral filter," https://github.com/MBernevig/On-Mesh-Bilateral-Filter, 2024.

A Example Images and Parameters

The following subsections contain larger images of the examples discussed in the paper, along with the parameters used to generate each output.

A.1 Example 1: Planar Meshes

This subsection presents the parameters used for generating the images seen in the example featuring planar meshes.

1. Figure 6a, texture processed using GIMP's Gaussian Blur:
   - Size X = 7
   - Size Y = 7
   - Filter mode = Auto
   - Abyss Policy = Clamp
2. Figure 6b, texture blurred using an On-Mesh Gaussian Filter with parameters:
   - rCoef = 0.006
   - Spatial Sigma = 0.2
   - Maximum Distance = 0.5
3. Figure 6c, texture blurred using an On-Mesh Gaussian Filter with parameters:
   - rCoef = 0.004
   - Spatial Sigma = 0.2
   - Maximum Distance = 0.5

A.2 Example 2: Cube Mesh

This appendix presents high-resolution images of cube meshes processed with various textures and filters. Below is a detailed enumeration of the parameters used for generating each output:

1. Figure 10: Application of a noisy texture on the cube mesh.
2. Figure 11: Cube mesh texture processed using GIMP's Gaussian Blur filter with parameters:
   - Size X = 1.53
   - Size Y = 1.53
   - Filter mode = Auto
   - Abyss Policy = Clamp
3. Figure 12: Cube mesh texture blurred using an On-Mesh Gaussian Filter with parameters:
   - rCoef = 0.01
   - Spatial Sigma = 0.2
   - Maximum Distance = 0.5
4. Figure 13: Cube mesh texture blurred using an On-Mesh Bilateral Filter with parameters:
   - rCoef = 0.01
   - Spatial Sigma = 0.2
   - Maximum Distance = 0.5
   - Color Sigma = 50

A.3 Example 3: Chest Pixelart

This subsection presents the parameters used when creating the images shown in the third example discussed, Chest Pixelart. This mesh and its textures were created by Onekro. [12]

1. Figure 14: Chest mesh with original texture image.
2. Figure 15: Chest mesh with texture blurred using an On-Mesh Gaussian Filter with parameters:
   - rCoef = 0.01
   - Spatial Sigma = 0.3
   - Maximum Distance = 0.6
3. Figure 16: Chest mesh texture blurred using an On-Mesh Bilateral Filter with parameters:
   - rCoef = 0.01
   - Spatial Sigma = 0.3
   - Maximum Distance = 0.6
   - Color Sigma = 20

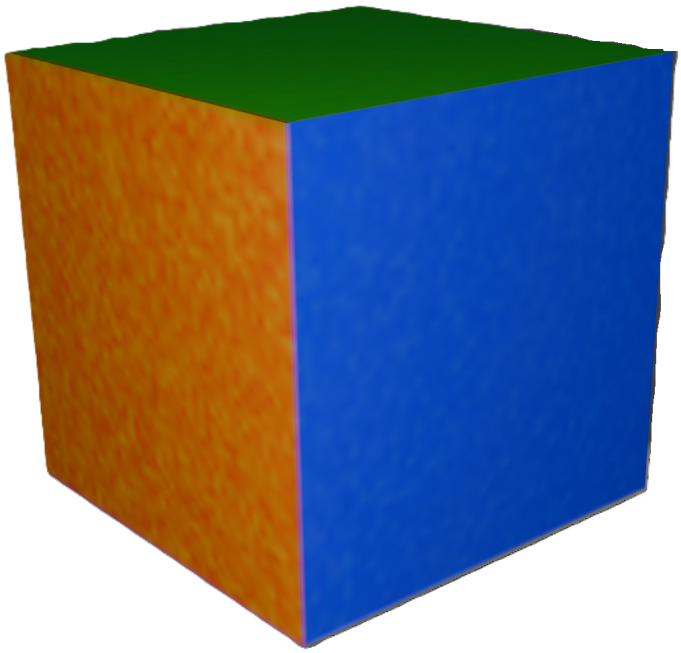


Figure 10: Cube mesh with noisy texture.

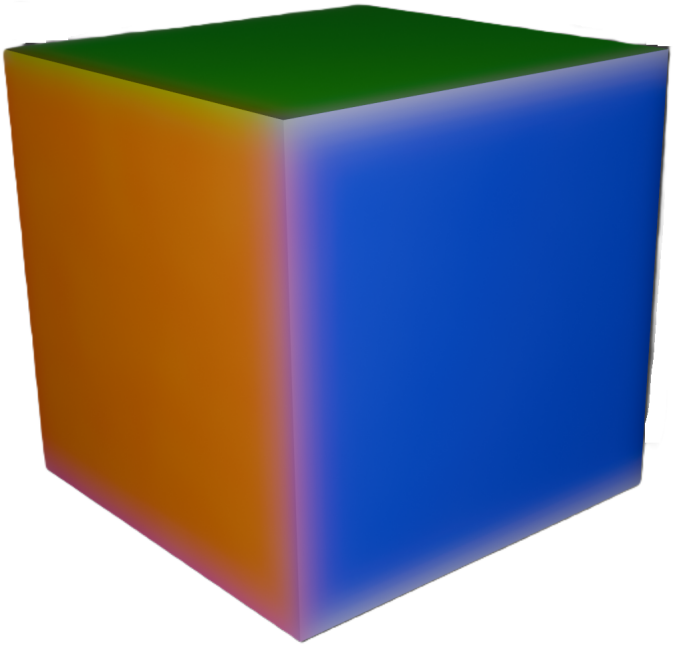


Figure 11: Cube mesh blurred using GIMP's Gaussian Blur filter.

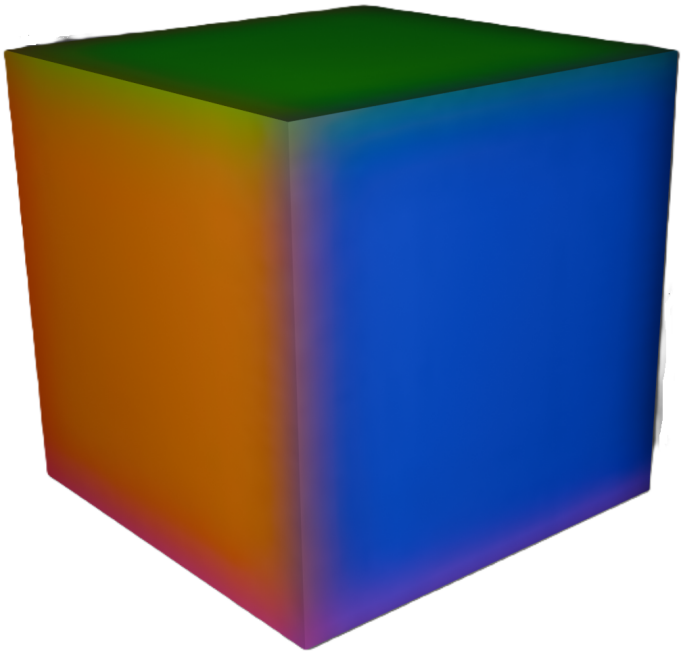


Figure 12: Cube mesh blurred using On-Mesh Gaussian Filter.

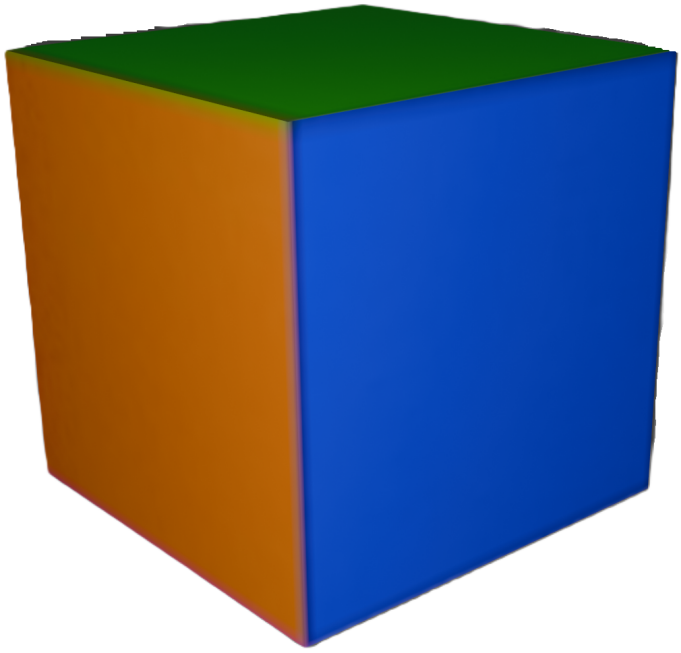


Figure 13: Cube mesh blurred using On-Mesh Bilateral Filter.

Figure 14: Chest mesh with unfiltered texture image.

Figure 15: Texture blurred with On-Mesh Gaussian Filter

Figure 16: Texture blurred with On-Mesh Bilateral Filter