MagicClay: Supplementary Material

ACM Reference Format:
. 2024. MagicClay: Supplementary Material. 1, 1 (October 2024), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 IMPLEMENTATION DETAILS

For reproducibility, we describe the settings used during the optimization. We intend to release the code upon acceptance.

1.1 Initialization

To initialize our hybrid representation at a consistent state, the mesh is encoded as an SDF at the start of the optimization. We use the pySDF library to get ground truth SDF values of the mesh, and in each iteration sample 1M points as described in [Müller et al. 2022], with the loss being the sum of the square differences between each samples value evaluated with the SDF network and the ground truth SDF. Due to the fact that the SDF rendering is sample based, the resulting SDF renders are usually blurry, as seen in Figure 1 Stage 1. The sampling done along the rays is dynamic using the NeRFACC library [Li et al. 2023]. To correct the blurriness and get a more consistent starting point for the optimization (for example, the one shown in Figure 1 Stage 2), we further optimize L2 multiview consistency losses for 300 iterations.

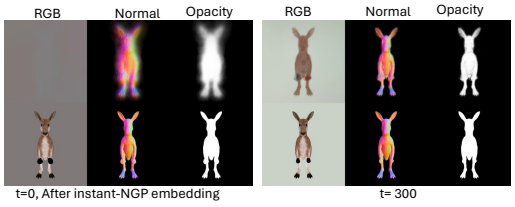

Fig. 1. **Effect of the two-stage initialization.** *Top row*: SDF render. *Bottom row*: mesh render. L2 multiview consistency losses provides a more consistent starting point for the hybrid representation, in addition to allowing initializing the SDF appearance network based on the initial mesh texture on non-editable areas.

Author's address:

ACM XXXX-XXXX/2024/10-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2 SDF representation

The SDF network uses hashgrid encodings similar to Instant-NGP [Müller et al. 2022], which allows to significantly shrink the MLP and consequently the cost of SDF queries, making the initialization take 5-10 minutes for 2000 iterations on a A100 GPU. We use the TextMesh [Tsalicoglou et al. 2024] implementation in ThreeStudio [Guo et al. 2023] as our SDF backbone, with the default hyperparameters (namely, $\lambda_{Eikonal}$ = 1000).

1.3 Training Procedure

We run our pipeline for 10000 iterations. and use λ_{Normal} = 0.1 and λ_{RGB} = 0.1 and $\lambda_{mesh_smoothness}$ = 0.1. Additionally, we apply the mesh smoothness loss only to non-frozen vertices.

1.4 mesh learning rate schedule

We use [Barda et al. 2023] as the re-meshing backbone. We set the learning rate to be 0.1, which we found provides a good balance between mesh responsiveness to the current SDF state while still keeping a stable and uniform expansion front. for the last 500 iterations we reduce the learning rate to 0.05, to promote convergence to a smoother mesh.

1.5 Topology updates to maintain the border between editable region and non-editable region

Special consideration must be given to topology edits when dealing with edges or faces that have a mix of frozen (i.e, not editable) and editable vertices. We outline the rules used in our pipeline:

(1) *Face split.* If one or two of the face's vertices are frozen this face can still be split, and the newly created vertex is editable
(2) *Edge collapse.* If one of the edge's vertices is frozen the edge may be collapsed, with the new vertex belonging to the frozen set of vertices.
(3) *Edge split.* For each edge, we observe it's one-ring vertices. An edge may be flipped if one of these two conditions are met: (i) any of its opposite vertices are editable or (ii) both of its vertices are editable.

2 EVALUATION

2.1 Unconditional text-to-3D

Average Clip Score Calculation. We average the clip score over 20 renders uniformly over a circular camera pattern, illustrated in Figure 2. The camera parameters used (using the standard Threestudio [Guo et al. 2023] camera definitions)
$elevation\_deg = 15^o$
$elevation\_deg = (-10, 90)$
$azimuth\_range = (-180, 180)$
$camera\_distance = 3.0$
$eval\_fovy\_deg = 70^o$

List of prompt in the benchmark. The prompts are randomly chosen from the Threestudio curated list.

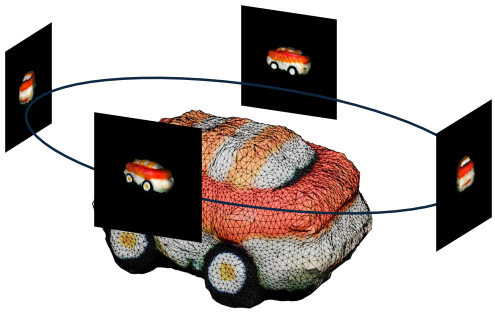

Fig. 2. **Clip Score calculation.** We illustrate four cameras out of 20 that are used to render the mesh and compute a averaged Clip score.

(1) A DSLR photo of a Space Shuttle
(2) A DSLR photo of a pug wearing a bee costume
(3) A DSLR photo of a roast turkey on a platter
(4) A DSLR photo of a squirrel made out of fruit
(5) A DSLR photo of a toy robot
(6) A DSLR photo of a tree stump with an axe buried in it
(7) Flower made out of metal
(8) A lionfish
(9) A plush dragon toy
(10) A typewriter
(11) A DSLR photo of a barbecue grill cooking sausages and burger patties
(12) A DSLR photo of a bulldozer
(13) A DSLR photo of a car made out of sushi
(14) A DSLR photo of a chow chow puppy
(15) A DSLR photo of a delicious croissant
(16) A DSLR photo of a frog wearing a sweater
(17) A DSLR photo of a ghost eating a hamburger
(18) A DSLR photo of a green monster truck
(19) A DSLR photo of an iguana holding a balloon
(20) A delicious hamburger

2.2 More baselines for local generative editing.

Instruct Nerf2Nerf Results. InstructNerf2Nerf [Haque et al. 2023] is a recent pipeline for editing NeRF scenes using prompts. The official implementation uses Instruct-Pix2pix [Brooks et al. 2023], which is trained on real images and is not suitable for synthetic data, Therefore it does not work for sculpting an existing 3D mesh because of the domain gap. Our best attempts at applying Instruct-Nerf2Nerf on synthetic data are given in fig 3. An email has been sent to the author, but as of the submission date, no response has been received.

TextDeformer details. Naively, one can attempt to perform editing using TextDeformer [Gao et al. 2023] by simply masking the gradients for all frozen vertices. In practice, TextDeformer does not perform the optimization directly on the vertices, but rather on their Jacobians, making decoupling the gradients for individual vertices not trivial, which in our experiments resulted in failure to converge. In order to apply TextDeformer for mesh editing tasks, we perform

Fig. 3. IntructNerf2Nerf results on synthetic data. IN2N uses NeRFStudio's Instruct-Pix2Pix as a backbone, which works best on real image. We show our attempts to use IN2N for synthetic data (male human mesh from the SMPL dataset). Applying IN2N on the resulting NeRFs lead to very blurry results.

.

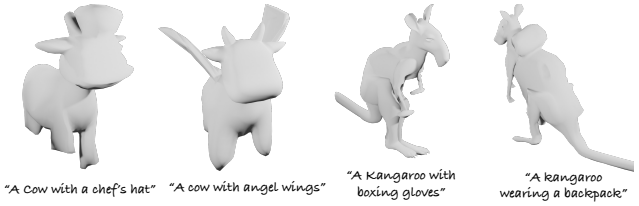

Fig. 4. **TextDeformer output without post-processing**. To complement the results presented in the main paper, we show the results of TextDeformer without our post-processing to localize the edit. As can be seen, the shape is globally affected.

.

a simple post process on the results: we first run TextDeformer on the input mesh with the input prompt to get the results shown in Figure 4. We combine the meshes by taking the frozen vertices of the original mesh and the non-frozen vertices of the edited mesh.

DreamEditor. The official implementation for DreamEditor [Zhuang et al. 2023] has a demo that runs on the authors' preprocessed data. However, running the method on new data is significantly challenging. Other users have opened issues on Github and the matter remains open as of submission time.

2.3 Further Details on Reconstruction Experiment

The model used was "Spot" [Crane et al. 2013] . The cameras were spread uniformly on the unit sphere in a 3X3 grid going over the elevation and azimuth axis respectively, as shown in Figure 5, producing 9 different renders.

2.4 Further ablation

No face super-sampling for mesh colors. Figure 6 illustrates the need for the mesh-driven super-sampling of the appearance network based on Mesh Colors scheme explain in the Method section of the main paper. We optimize our representation with respect to a target image (first column), either sampling colors per-vertex (second column), or using our adaptive sampling using MeshColors (third column). Note that without MeshColors sampling a much higher resolution is needed, which leads to poorer reconstruction and an over-tesellated mesh.

3 APPLICATION

3.1 Application: Animation Transfers

In production, meshes typically carry animation information. For instance, a rig *i.e.* a collection of joints connected in a tree graph structure. Each joint is coupled to a group of vertices, so rig movements can be translated to mesh movement. This coupling is given by vertex groups, which is part of the mesh metadata. Because the non-editing regions are preserved, MagicClay allows us to carry over this information through the optimization.

Vertex group assignment are trivially preserved in the non-editing region. For the editing region, we simply need to ensure that the information is preserved through the various topology updates of ROAR [Barda et al. 2023]. For each vertex, we add a vector of size #VG, which is a one-hot encoding for the respective vertex group.

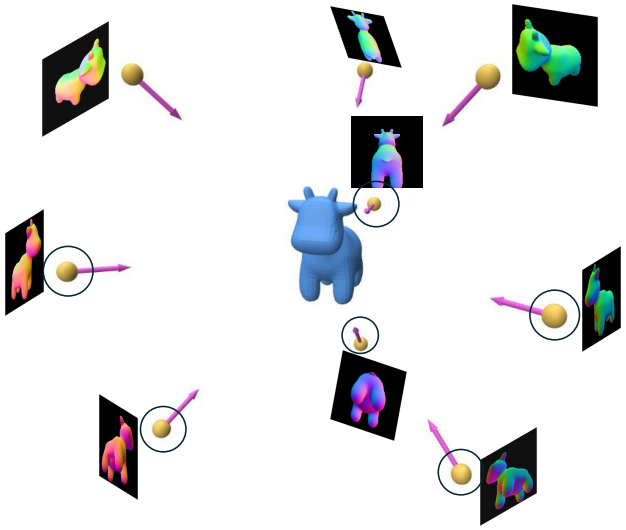

Fig. 5. Camera setup for the multi-view reconstruction experiment. The cameras are set up in a 3X3 configuration uniform on the unit sphere, camera for training views are circled.

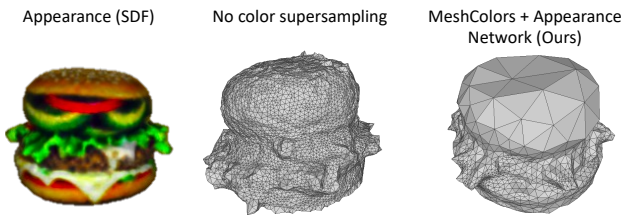

Fig. 6. **Ablation: no color supersampling.** Using mesh-guided supersampling in conjunction with the appearance network allows to decouple geometry and appearance. Using this approach (top right), large faces are used for the mesh, even though the rendering still presents high frequency colors (bottom). When using a color-per-vertex scheme (top middle), significantly larger mesh resolution is required to achieve similar appearance.

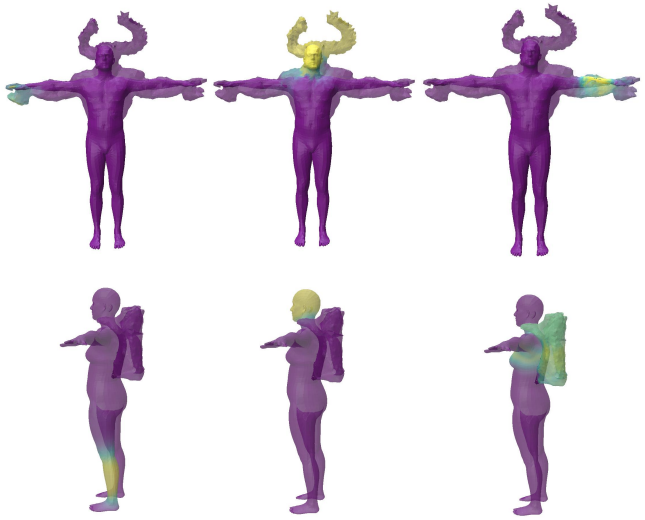

Fig. 7. **Vertex Group Transfer.** We superpose the mesh before and after editing by MagicClay. We color the vertices based on their membership to a specific vertex group. In each column, we choose a different vertex group. The topology updates of ROAR [Barda et al. 2023] preserve vertex features, such as the vertex group used in animations.

Fig. 8. **Animation Transfer.** We perform the animation transfer on the preexisting SMPL [Loper et al. 2015] animations, given in FBX format. the rig joints' position is given by the vertex groups, which are reasonably preserved due to the incremental nature of our mesh evolution process. We can then simply insert our mesh instead of the SMPL mesh and inherit existing animations for the given rig. Note, that in our experiments, for simplicity, we removed all existing blend shapes that are associated with the original vertex groups.

For face splits, we compute a weighted average of the neighboring vertices and take the maximum to assign the new vertex to a group. If an edge is collapsed, we pick the one-hot encoding of the adjacent vertex with the highest Q-slim score. As can be seen in Figure 7, the topology updates preserve the vertex groups.

3.2 Application: Textured Meshes

As described in the main paper, in order to decouple the RGB rendering from the topology, our appearance network is queried on the

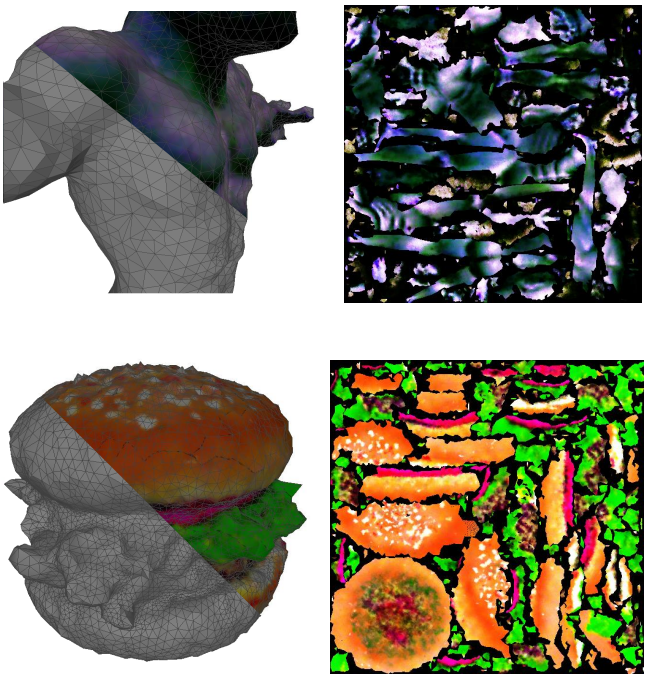

Fig. 9. **Super-sampled colors to UV texture.** One of the benefits of meshes is their ability to describe texture in a light and efficient-to-render way. The super-sampling operation allows to finely sample the color field over the mesh. This sampling can then be baked to a texture in conjunction with uv-unwrapping using Xatlas, to output a textured mesh to the user.

super-sampled faces, in a way similar to that of Mesh colors [Yuksel et al. 2010]. UV textures are more common, and our super-sampled face colors can easily be turned into this representation, as shown in Figure 9.

REFERENCES

Amir Barda, Yotam Erel, Yoni Kasten, and Amit H. Bermano. 2023. ROAR: Robust Adaptive Reconstruction of Shapes Using Planar Projections. (2023). arXiv:2307.00690 [cs.GR]

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. (2023).

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

William Gao, Noam Aigerman, Thibault Groueix, Vladimir G. Kim, and Rana Hanocka. 2023. TextDeformer: Geometry Manipulation using Text Guidance. *SIGGRAPH* (2023).

Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. 2023. ThreeStudio: A unified framework for 3D content generation. https://github.com/threestudio-project/threestudio. (2023).

Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. 2023. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In *ICCV*.

Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. 2023. NerfAcc: Efficient Sampling Accelerates NeRFs. *ICCV* (2023).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: a skinned multi-person linear model. *ACM Trans. Graph.* 34, 6, Article 248 (oct 2015), 16 pages. https://doi.org/10.1145/2816795.2818013

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics* (2022).

Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. 2024. TextMesh: Generation of Realistic 3D Meshes From Text Prompts. *3DV* (2024).

Cem Yuksel, John Keyser, and Donald House. 2010. Mesh Colors. *ACM Transactions on Graphics* (2010).

Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. 2023. DreamEditor: Text-Driven 3D Scene Editing with Neural Fields. *SIGGRAPH Asia* (2023).