3D shape reconstruction from non-realistic multiple-view depictions using NVDiffRec

Joan Colom [1,2,*], and Hideo Saito [1]

[1] *Keio University, Hiyoshi Kohoku-ku, Yokohama, 223-8522, Japan*
[2] *Universitat Politècnica de València, Camino de Vera, Valencia, 46022, Spain*

Abstract

We present a workflow for applying the SOTA in multi-view reconstruction over realistic images in the domain of sketches and drawn-like images. With this aim, we leverage NVDiffRec to study its performance over the non-realistic domain through custom use cases. When doing so, we will expose the challenges of using the system over a different domain and present our solutions. Finally, we will detail the obtained results and conclusions on the viability of NVDiffRec as a possible tool for fictional 3D content generation from concept art.

Keywords

Computer Vision and Pattern Recognition, Graphics, Machine Learning, Image and Video Processing, 3D Reconstruction, Inverse Rendering

1. Introduction

The generation of 3D shapes from 2D content has been widely researched. From point cloud or mesh generation to implicit representations using neural networks, a great variety of approaches are used to tackle this problem. However, most efforts have focused on reconstructing objects depicted in real-life images or realistically rendered synthetic scenes.

Although these systems allow a wide variety of applications for 3D content generation, they have been limited to generating already existing objects. In many media-related tasks, the possibility of generating 3D versions of custom or fictional objects is necessary. These are usually defined through concept art, images depicting the target from multiple points of view, making it possible to see their three-dimensional properties. Therefore, multi-view reconstruction techniques that can work in this domain are required.

When considering such an application, we must take into account the challenges of the medium. By nature, drawings of an object from different views are not perfectly aligned or geometrically coherent, presenting an inherited *looseness* in the 3D shape they convey. Additionally, the number of available source samples is much more limited.

The works in 3D reconstruction from sketches have been trying to solve these issues. However, their application has been limited to, as far as we know, *only* sketches, not considering drawings with color. In this regard, the joint mesh and texture estimation of the SOTA of reconstruction over realistic images could be helpful.

As a result, we aimed to study if, using SOTA techniques in reconstruction with multi-view real images, it is possible to broaden the application domain to any level of developed art depictions. For that, we focused on the NVDiffRec system proposed in [4] due to its promising results and the capability of generating a textured 3D mesh in an exportable format. It is important to note that we will be using NVDiffRec under a different domain than originally intended, but by doing this, we aim to determine if the generalization of this kind of

APMAR'22: Asia-Pacific Workshop on Mixed and Augmented Reality, Dec. 02-03, 2022, Yokohama, Japan
*Corresponding author.
EMAIL: joacoco@inf.upv.es (Joan Colom); hs@keio.jp (Hideo Saito)
ORCID: 0000-0001-7577-0657 (Joan Colom); 0000-0002-2421-9862 (Hideo Saito)

techniques allows working on the artistic field. Our contributions are as follows:

- Aiming to obtain 3D reconstructions not only from sketches but also from any art-like multi-view depiction.
- Finding a workflow that allows using NVDiffRec from images whose masks and viewpoints have not been provided.

This paper will introduce the baseline works and review NVDiffRec in Section 2. In Section 3, we will detail our experiments and the use cases tested. Finally, Section 4 will present the results, Section 5 will draw our conclusions, and Section 6 will propose the next steps for our research.

2. Related work

The reconstruction of 3D shapes from images has been the spotlight of a vast library of previous works. Among them, we can find a great variety of approaches and different reconstruction levels. From voxels to implicit representations going through point clouds and meshes, the variety of representations allows for many different techniques and strategies. Moreover, the type of source and the detail of the reconstruction, whenever only 3D shapes or materials, lighting, and surface are jointly targeted, add additional complexity to the problem. In this section, we will introduce the baseline works for our research and the main system used in our experiments.

2.1. Sketch 3D shape estimation

Although 3D reconstruction from images has received more attention, three-dimensional shape estimation from sketches has also been a broadly researched topic. However, it involves additional challenges due to the differences in drawing styles, inconsistencies between views, lack of shading that hints at the surface, and, in many cases, lack of ground truth data. As baselines for our research, we selected two works that aimed to reconstruct 3D objects directly from sketches represented as regular images without requiring user interaction to guide the reconstruction.

Firstly, Han et al.'s work [6] uses multi-view sketches to optimize a 3D voxel grid and generates the corresponding model. This involves a CGAN that predicts the geometry by generating attenuation maps from the sketches, followed by a Direct Shape Optimization algorithm to optimize an occupancy-based voxel grid. In this approach, sketches must be accompanied by their viewing angle, and for training, a synthetically generated dataset was used.

This work differs from our experiments in optimizing a voxelated internal representation. As we use NVDiffRec, the mesh is directly optimized, allowing for a finer fitting when compared to converting the optimized voxels into meshes [4].

Secondly, the work by Lun et al. [1] gets closer to a traditional multi-view approach to reconstruction. Thanks to a CNN, multi-view sketches can be used to generate the object's depth, normal, and foreground probability maps from 12 fixed views. With this, partial point clouds for each view are obtained, and a final point cloud is generated through optimization. Later, the result is converted into a mesh and further optimized by contour fitting with the sketches.

As far as we know, this is the closest work to our experiments, generating a mesh from multi-view sketches and considering the direct refinement of the mesh. However, a key difference is that it assumes fixed canonical views for the sketches, requiring a new network to be trained for every unique combination of views. By using NVDiffRec, we can provide any sequence of arbitrary views of the object.

Finally, our experiments also present two additional distinctions regarding both works. On the one hand, we consider sketches and drawn-like images, broadening the application domain. Therefore, we will estimate not only the shape but also the textures associated with it. On the other hand, the deep learning approaches used in the exposed works focus on *training* as means of obtaining systems that can be used by *inference*. In contrast, thanks to NVDiffRec, we model the *reconstruction as training*, being the main objective of this process obtaining our 3D object.

2.2. NVDiffRec

As the main system under our experiments, we will briefly expose the concepts and ideas behind NVDiffRec. Developed by Munkberg et al., NVDiffRec aims to jointly estimate an object's 3D shape, materials, and lighting conditions given its multi-view images, associated masks, and camera poses [4]. Thanks to directly optimizing mesh and materials through differentiable rendering, this work allows compatibility with standard existing 3D manipulation tools.

To accomplish this, the mesh is encoded using a tetrahedral grid whose vertex displacements and SDF values are estimated progressively through training. At every step, the grid is converted to mesh by efficient marching tetrahedra and later rendered, obtaining the loss when comparing the result with the ground truth. In turn, that loss modifies the grid values by backpropagation. Therefore, training is required for every new dataset we wish to reconstruct.

For the materials, two types of representations are used. In the first training pass, implicit representation through MLP of the diffuse color, roughness, and metalness is used. On the second pass, learnable textures created from the implicit representation are employed. These two training phases allow focusing on shape estimation in the first pass, fixing it in the second pass for surface and texture refinement.

Environmental light conditions can also be learned. This is thanks to a learnable cubemap texture encoding the specular lighting, whose mipmaps are obtained by filtering, and an additional low-resolution learnable cubemap for encoding the diffuse lighting.

After training over a set of images, the 3D model, materials, and environment map are exported using standard formats. Therefore, compatibility with external tools is accomplished.

3. Our experiments

Our goal was to use NVDiffRec with non-realistic images. To do so, first, it was necessary to determine the characteristics that datasets needed:

- A set of multi-view images of an object.
- A set of masks, one for each image, preserving the target object and hiding the rest. These can be represented in the alpha channel.
- A set of view matrices, one for each image, describing the position and orientation of the camera used to capture the image.

In our application with non-realistic images, the masks could be easily generated if we take the concept art developed by a digital artist. However, this is a challenge when sketches are already rendered or made traditionally. Moreover, the need for camera information can be an even more challenging problem to overcome in these situations. Next, we will present the workflow of our experiments for masks and camera information generation. Figure 1 shows a summary of the said workflow.

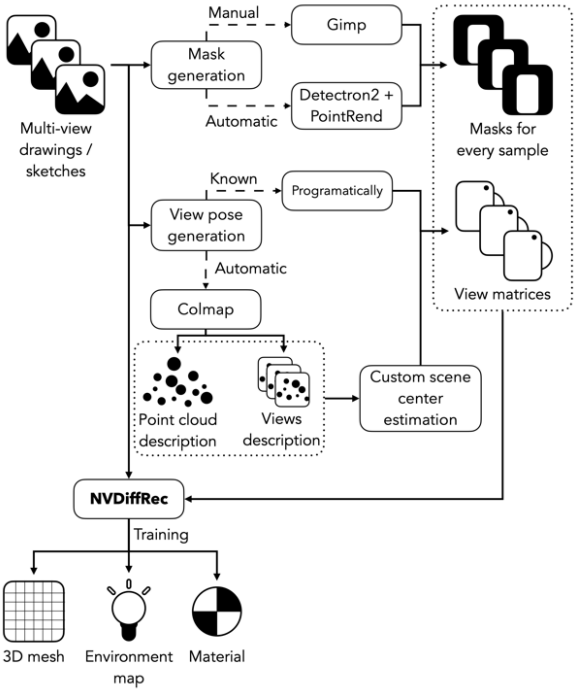

Figure 1: summary of the proposed workflow. Dashed arrows are conditional paths. Multi-view drawings and sketches represented as images are taken. First, mask generation is applied, either manually or automatically. Then view poses are generated either with prior information or automatically using Colmap and adapting the results. Finally, images, masks, and views are used in NVDiffRec to obtain a textured model.

3.1. Generating masks

Masks are required to identify the target object that we want to reconstruct. Given a set of 2D images that are not masked, we can follow two possible approaches to mask them:

- Process them manually with software such as Gimp or Photoshop. This allows more accurate results but is much more costly. For big amounts of data, it becomes unfeasible.
- Automatically analyze them to identify the target object and mask it. This is also known as object segmentation and constitutes an open problem. Depending on the target, this method can produce faulty masks that can mislead the reconstruction. However, it allows the generation of masks for big volumes of samples with a much lower cost.

3.2. Generating view information

NVDiffRec uses rendering to obtain feedback from the samples. Therefore, knowing the view matrix associated with each sample is necessary to render it correctly from the same viewpoint relative to the object.

Given a set of multi-view images with no camera information, we need to generate the view matrices in a way that is consistent with the target object and between shots. In this case, we can also identify two different approaches:

- If the images follow a known uniform transformation, we can simulate this transformation and generate the view matrices.
- When the images do not follow a known uniform transformation, estimating the view matrices is challenging. This falls under the umbrella of research areas such as Structure-from-Motion (SfM) [7] and camera pose regression [8]. Therefore, we must recur to the developed tools in this area.

Choosing the approach to follow depends on the use case we face. The next section will present three examples we used with NVDiffRec and the proposed workflow.

3.3. Use cases studied

We experimented with NVDiffRec over three use cases. Firstly, a drawn depiction of a sphere. Secondly, dog sketches corresponding to a turning animation. And finally, a recording of a character in a cell-shaded game.

3.3.1. Sphere

We introduce a simpler base case by presenting a digitally drawn circle, already masked and seen in Figure 4, from various points of view, simulating multi-view samples of a sphere. Two approaches were used to generate the view matrices:

- Simulating a turn-around of 28 frames around the vertical axis, like in Section 3.3.2.
- Generating completely random rotations around the scene's center at a fixed distance.

3.3.2. Dog sketches

In this use case, sketches of a dog like the ones in Figure 2 are available, corresponding with the 28 frames of a complete turn-around animation. To use them, we generated the masks of the dog, and the camera poses for each frame.

Due to the reduced number of images, we opted for generating the masks manually using Gimp, alpha masking outside the black outline, and erasing the ground line.

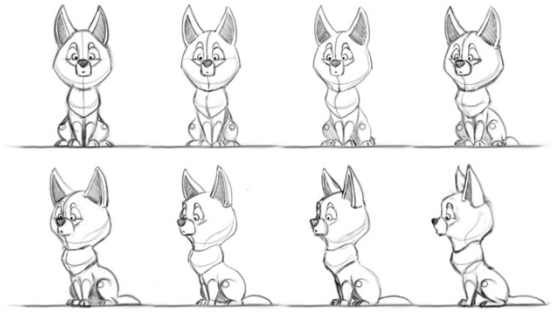

Figure 2: Sketches depicting the partial turn-around of a fictional dog. The complete set, created by Anja Regnery, can be found in [3].

The view matrices for each frame were estimated thanks to the turn-around nature of the source. Knowing that the 28 frames describe a turn of 360 degrees around the vertical axis, we can start at an arbitrary distance from the origin on the first frame and rotate 15 additional degrees around the vertical axis for each next frame.

3.3.3. Game character

The last use case we propose consists of reconstructing the character of a third-person view game in which the camera can freely move around it. However, some remarks must be made.

The reason for this example resides in the non-realistic-looking nature of the content, like a painting, and the ease of generating samples. Nonetheless, its similarity to real drawings is only partial due to the high geometrical consistency that it presents through views given its synthetic origin. Additionally, it allowed us to obtain many samples, which is unfeasible with drawings. Despite these issues, we still consider it a valuable example that allowed us to deal with challenging mask generation, view estimation, and study their effects in the reconstruction.

We took a screen recording of a game, depicting the camera moving around the standing character to obtain the samples. By extracting all the frames, we gained 921 images of the character.

For masking them, we opted for generating masks automatically. This was achieved using

Detectron2's API and the PointRend model [5] to identify recognizable objects and their segmentation masks. By joining all the segmentations, we generated the mask of the image. This approach has the inconvenience of occasionally introducing outlier objects in the masks or masking out the target. We removed from the set those images that, after masking, were empty.

As the camera was controlled manually, we cannot assume uniformity in its movements. Furthermore, the camera's movement is random and cannot be either assumed. Therefore, computing the camera for each frame like in Section 3.3.2 seemed unfeasible. We opted to use Colmap to tackle the problem [24].

This tool allows processing large amounts of images and using their key points to find the spatial relations between them, estimating a point cloud representation of the scene. As a result, from multi-view images, Colmap estimates the camera pose of each image. However, the compatibility between Colmap and NVDiffRec is not direct.

Firstly, Colmap and NVDiffRec have different coordinate systems, with the Z and Y axis inversed in one respect to the other. Secondly, Colmap also estimates the origin of coordinates of the scene. Given that this point depends on the camera distribution, as shown in Figure 3, the center generally does not match the target's center unless a very uniform view distribution is given. This causes a disparity between the render and the ground truth because NVDiffRec places the mesh in the origin, but in the estimated view by Colmap, the target is not in the origin.

Therefore, we must determine the object's center in the coordinates estimated by Colmap and use it as a new origin to solve it. We explored two solutions.

On the one hand, if we do not know the nature of the object and its location in the different views, we consider only the view poses to estimate the real origin. To do so, we can assume that, as the samples capture a single object from different viewpoints, the camera positions are approximately distributed on the surface of a sphere of radius R around the target.

Given these considerations, we can locate the new origin by finding the sphere that most closely explains the camera positions. Moreover, we can also guide our decision by considering the cameras' looking directions. To solve this problem, we designed a GRASP algorithm that, given camera positions and looking directions, tries to approximate the desired sphere by heuristically generating solutions and saving the best one. Further details on this algorithm are given in the Appendix.

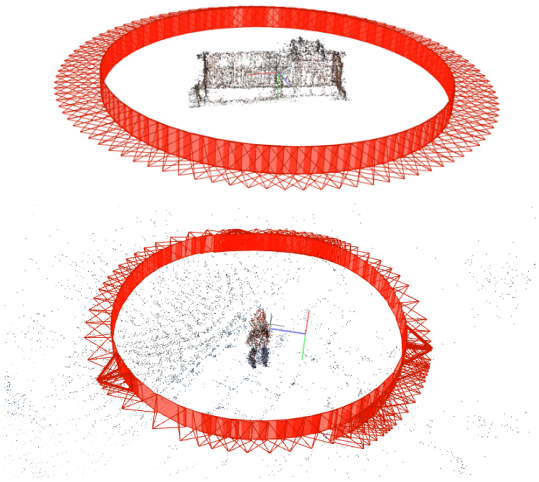

Figure 3: Above, Colmap estimation of a uniformly distributed scene view. The origin falls at the center of the object. Below, estimation of a non-uniformly distributed view in a different scene. The origin does not fall in the object.

On the other hand, we can use additional information to get a better estimation. As we know that in our use case the target is always in the center of the screen, we can assume a known bounding box inside the images that always contains it. This is reasonable as, when taking multi-view samples of an object, it is usually kept in the same area of the image. Moreover, a bounding box could be easily defined by a user.

With this bounding box (BB), we can use the information generated by Colmap to filter the point cloud of the scene and then compute the center of this filtered version. Filtering is archived by applying a voting scheme such that each key point inside the BB in an image receives one vote. After analyzing all the samples, we can preserve only the K most voted key points. Therefore, the center can be computed as the weighted average using the votes as weights.

This approach is intuitive as key points of the target should be more commonly seen. However, it can be limited depending on the use case by the requirement of specifying a bounding box.

Finally, we can use both estimations to obtain a new averaged center. We also tried this approach by weight averaging. The weights were computed using Equation 1, where C is the set of all cameras with look at vector $\vec{v}$ and position $\dot{p}$.

Therefore, we give a higher weight to the points better aligned with the views.

$$w(\dot{x}) = \left(\sum_{(\vec{v},\dot{p}) \in C} 1 - \left(\vec{v} \cdot \frac{(\dot{x} - \dot{p})}{\| \dot{x} - \dot{p} \|} \right) \right)^{-1} \quad (1)$$

4. Results

For all the experiments detailed with NVDiffRec, we used 5000 iterations, random initial textures, texture resolution of 1024 by 1024 pixels, batch size 4, grid resolution of 128, and reconstruction in two phases with learning rates of 0.03 and 0.003, respectively. When Colmap estimation was needed, we used all the full-resolution images without masking. Shared parameters were used for the cameras and default configuration for the remaining attributes.

4.1. Sphere

In this case, we modified the camera used in NVDiffRec to be orthographic to match the ground truth. As far as we know, this is the first time it has been used with this type of camera.

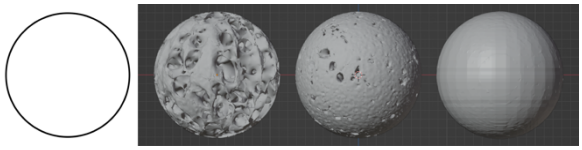

Figure 4: Left to right, ground truth, estimation by NVDiffRec with turn-around, estimation with random rotations, and estimation with Visual Hull [9] with turn-around.

Figure 4 shows the meshes obtained with NVDiffRec when estimating the simulated sphere. The same estimation obtained through the Visual Hull algorithm provided in [9] is also presented as a reference in this comparison. For the Visual Hull, 28 turn-around images of the ground truth outline were used.

4.2. Dog sketches

The first experiment with the dog sketches was executed using a perspective camera projection, environment light optimization, and 550 by 550 pixels training resolution. All images were used for training. The progress result saved during the last iteration can be seen in Figure 5.

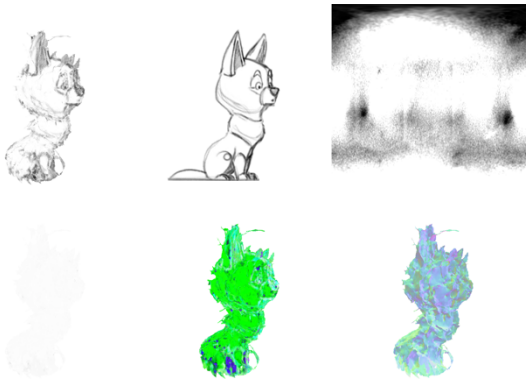

Figure 5: Results saved on the last training iteration in NVDiffRec (initial experiment). From top to bottom, left to right, rendered mesh, ground truth, environment map, diffuse texture, specular texture, and normal map.

NVDiffRec tries to approximate the silhouette of the dog and the general shape obtained when rendering. However, the lower parts of the body, such as the tail and paws, are missing. This can be attributed to the inconsistency between the views and the projection used. Reference sketches tend to avoid perspective deformation; therefore, they are usually more closely explained by an orthographic projection. Furthermore, we can also appreciate how the grey lines are tried to reproduce with the lighting instead of the textures.

We repeated the same training, using an orthographic projection, and fixed white environment light. The result can be found in Figure 6. It can be observed that orthographic projection allows for a closer similarity of silhouettes and outlines between reconstruction and sketches. Moreover, the tail and paws are now included.

Finally, we increased the number of samples. This was possible through interpolation between frames using AnimeInterp [2] to generate two additional frames between the existing ones. With this technique, the number of samples was increased to 84, and the experiment was repeated using all of them. Figure 7 shows a comparison of the meshes obtained with each experiment. Again, the reconstruction obtained via Visual Hull [9] with the 28 samples has also been added for reference.

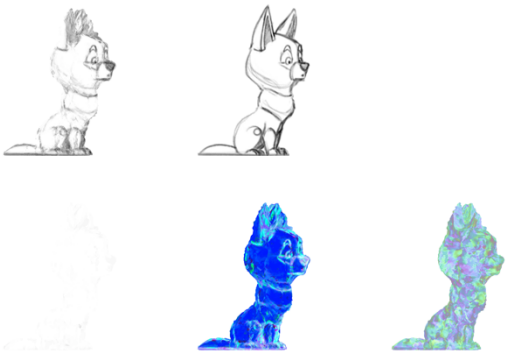

Figure 6: Results saved on the last training iteration in NVDiffRec (second experiment). From top to bottom, left to right, rendered mesh, ground truth, environment map, diffuse texture, specular texture, and normal map.

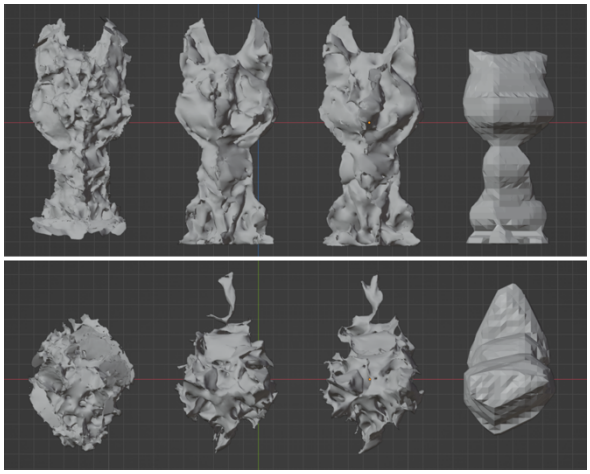

Figure 7: Reconstructed meshes for the dog from the front and top. Left to right, perspective, orthographic, orthographic interpolated, and Visual Hull estimations.

4.3. Game character

A total of 921 images of 1920 by 1342 pixels were obtained by extracting all the frames from the source video. All of them were used for the Colmap estimation, while 737 formed the training split and 184 formed the validation split. These splits were masked, filtered to remove the empty images, and resized to half size.

We divided the experiments into two groups. On the one hand, the experiments in which the different strategies for center estimation were applied with automatic masks. On the other hand, the experiments with improved masks. In all cases, perspective was used, the training resolution was 960 by 671 pixels, and the lighting was learned.

4.3.1. Origin estimation

We applied the NVDiffRec reconstruction training separately for each of the proposed center estimations: by GRAPS, BB projection, and averaging both. Table 1 presents the numeric results obtained in validation, while Figures 8 and 9 visually show the obtained reconstruction. After the masking and filtering, the dataset was reduced to 517 samples for training and 132 for validation.

We can see that, for all center estimation techniques, the results look similar. Looking at Table 1, the averaged and BB estimations obtain slightly better PSNR, although the MSE is similar in all cases. Figure 9 shows that the differences are found in details like the hair shaping and the surface texture. It is worth noting that all cases fail to recover the hands and the foot are very roughly reconstructed. It also shows that the main difference between the models is where the center of the resulting mesh is placed. It is important to note that the models have an implicit rotation. This is due to the Colmap estimation of the system of coordinates. Even though we displaced it, we did not modify its orientation.

Table 1

Average MSE and PSNR in validation of the reconstruction of the game character for different center estimations with automatic masks.

Estimation	MSE ↓	PSNR ↑
GRASP	0.008	23.77
Bounding box	0.008	23.90
Averaged	0.008	23.93

4.3.2. Masks improvement

Given that we had assumed the availability of the BB containing the target, we used this information to improve the masks. This was archived by automatically masking anything located outside the bounding box. Moreover, the filtering was also improved by removing those samples whose bounding box interior was empty. This method allowed for more refined masks, obtaining 501 training samples and 124 evaluation samples.

We tested the reconstruction for the BB and the averaged origin estimations again with the improved masks. Table 2 shows the metrics obtained in validation, while Figure 10 shows a visual comparison of the estimated 3D models.

Table 2

Average MSE and PSNR in validation over improved samples with the obtained models from automatic and improved masks.

Estimation	MSE ↓	PSNR ↑
Bounding box	0.004	25.03
Averaged	0.006	23.85
Bounding box^{+}	0.003	27.73
Averaged^{+}	0.003	27.77

Figure 8: Validation results over the first sample for different center estimations on the game character. Left to right, GRASP, BB, and averaged. Top to bottom, ground truth, rendered reconstruction, diffuse texture, specular texture, and normal map. Images have been cropped for visibility.

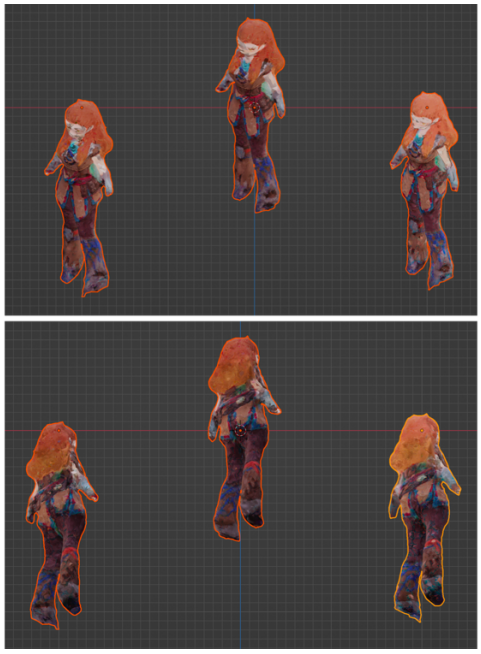

Figure 9: Captures of the generated 3D models of the game character during the first experiment viewed from the front and back. Left to right, GRASP, BB, and averaged estimations.

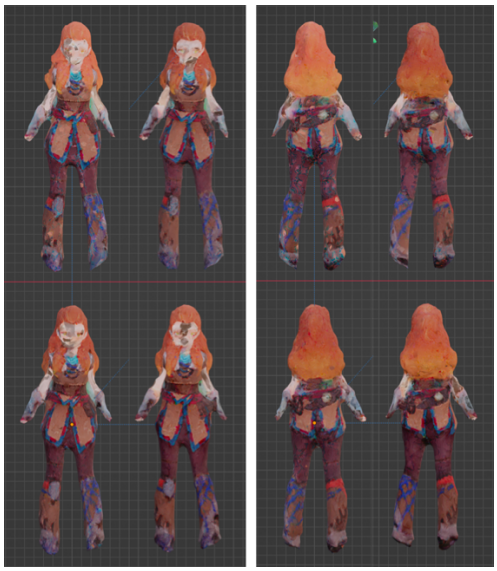

Figure 10: Models obtained for the game character without and with improved masks viewed from front and back. In each set, left to right, estimation with normal and improved masks, top to bottom, average, and BB estimations. All models have been rotated for the comparison; the original orientations were as in Figure 9.

The reconstructions with the improved masks are visually similar to the initially obtained ones. Looking at Figure 10, we can see a slight increase in the sharpness of the textures. Regarding the 3D

shape, we can observe small improvements in the shaping of the hair. In Table 2, we can see that, when evaluated over the more refined validation set, the models trained with improved masks perform better than those trained with fully automatic masks.

5. Conclusions

Through all the experiments and results presented, we can see it is possible to obtain promising results using NVDiffRec. However, it is not ready to be applied by users in this domain.

With the sphere and dog cases, we can see that extracting mesh information from only sketches is highly difficult, considering the inconsistencies of the outlines between views and the lack of shading. In the case of the sphere, a reconstruction through Visual Hull gives a perfect result with only a turn-around. Meanwhile, NVDiffRec has difficulties filling the surface thoroughly, even with a wide variety of random views, producing small holes in the surface. The main reason is that NVDiffRec only uses direct lighting without shadows; therefore, these holes do not produce visual feedback when rendered. That causes NVDiffRec to optimize the shape based on the outlines of the views without diffuse or specular shading. In this way, the higher the number of viewpoints, the more consistent the mesh is with all possible outlines.

Nonetheless, the results obtained with the dog sketches show that, in challenging situations, NVDiffRec allows more detailed results than Visual Hull, providing higher silhouette fitting. This is thanks to all the samples contributing to the optimization. Even though some may be faulty or misaligned, the rest still contribute. This also explains the robustness seen with the game character, even with the defective masks. However, Visual Hull requires a higher consistency, which can also be seen in the algorithm failing to generate any mesh for the game character dataset.

With all this, the dog sketch results are still inadequate for a real user, failing to generate the surface properly. If additional views were provided from the top and bottom as in the sphere, we theorize that better results would be possible, but we have not been able to try it.

We can therefore identify components in NVDiffRec that are not suitable for drawn-like images:

- Lighting estimation. In most concept arts, objects are depicted without strong shadows, with soft shading, or with no lighting. Like with the dog sketches, using fixed lighting may be more beneficial.
- Specular texture estimation. In drawings and sketches, specularities are rare and mostly depicted in a non-realistic way. Therefore, the texture details are more desirable to be wholly integrated into the diffuse map, avoiding results such as Figure 6.
- Camera inputs. Defining the viewpoints for drawings or sketches can be extremely difficult.
- Perspective camera. As we have shown, an orthographic camera can be more suitable in some cases.
- Local lighting.

Despite all this, there are still suitable components in NVDiffRec for drawing reconstruction. The direct optimization of the mesh, as well as the normal map and the diffuse map estimations based on rendering are relevant.

With the game character, we have seen the importance of the view estimation for a good reconstruction, as it will directly affect the render and the consistency with the ground truth. Colmap and the need for many key points limit our workflow for automatic view estimation. This makes its application on actual drawings difficult, especially in the absence of any background scenery in the depiction. However, the experiments with the game character show that the results improve if enough detailed samples in a drawn style are provided.

In conclusion, our workflow application is currently limited by the difficulty of dealing with small amounts of viewpoints for drawn objects and the limitations of local lighting. Therefore, using NVDiffRec does not solve the applications we aimed to tackle, which we presented in Section 1. We theorize that, with differential ray tracing techniques that confer global illumination, the estimations would improve significantly as the holes would reveal themselves and the more robust nature of deep learning approaches could deal with the outline inconsistencies.

6. Future work

Even though NVDiffRec is not ready to use by our intended use cases, we consider the results promising. Therefore, we aim to continue our research by finding ways to tackle the limited set

of viewpoints and explore alternative render pipelines.

Acknowledgments

Special thanks to Anja Regnery for allowing us to use her dog turn-around animation sketches in our experiments [3]. All the materials for this project have been used under Fair Use. The game character images were obtained from a screen recording made by us of the game Genshin Impact.

7. References

[1] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji and R. Wang. "3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks". CoRR, vol. abs/1707.06375 (2017). URL: http://arxiv.org/abs/1707.06375

[2] S. Li, S. Zhao, W. Yu, W. Sun, D.N. Metaxas, C.C. Loy, and Z. Liu. "Deep Animation Video Interpolation in the Wild". CoRR, vol. abs/2104.0249 (2021). URL: https://arxiv.org/abs/2104.02495

[3] A. Regnery, Dog Turnaround Animation, 2020. URL: https://www.behance.net/gallery/95032661/Dog-Turnaround-Animation

[4] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller and S. Fidler. "Extracting Triangular 3D Models, Materials, and Lighting From Images". arXiv (2021). doi:10.48550/ARXIV.2111.12503.

[5] A. Kirillov, Y. Wu, K. He, and R. Girshick, "PointRend: Image Segmentation as Rendering". arXiv (2019). doi:10.48550/ARXIV.1912.08193.

[6] Z. Han, B. Ma, Y.-S. Liu and M. Zwicker, Reconstructing 3D Shapes From Multiple Sketches Using Direct Shape Optimization, IEEE Transactions on Image Processing, vol. 29 (2020) 8721-8734. doi:10.1109/TIP.2020.3018865.

[7] J. L. Schönberger and J.-M. Frahm, Structure-from-Motion Revisited, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104-4113. doi:10.1109/CVPR.2016.445.

[8] T. Sattler, Q. Zhou, M. Pollefeys and L. Leal-Taixe. "Understanding the Limitations of CNN-based Absolute Camera Pose Regression". arXiv (2019). doi:10.48550/ARXIV.1903.07504.

[9] keith2000, VisualHullMesh, 2021. URL: https://github.com/keith2000/VisualHullMesh

Appendix

Finding the sphere that best describes the view distribution of a set of cameras constitutes an optimization problem for which exact methods would be unfeasible in big datasets. Therefore, we try to find an approximation in a reasonable time by using a Greedy Randomized Adaptive Search Procedure (GRASP) algorithm.

Our implementation reduces the sphere estimation problem to the task of finding four cameras whose positions describe a sphere that approximates the distribution of all the views. Consequently, our GRASP can focus on generating solutions formed by a sequence of four camera positions. After obtaining these points, the center and radius of the sphere can be obtained by applying the general equation of the sphere.

Algorithm 1 presents our GRASP proposal for sphere estimation. Following the general scheme of this kind of algorithm, every iteration has two phases:

- A constructive phase in which N solutions are generated. Each solution is built step by step, adding progressively new elements (camera positions). The first element is picked randomly among all the points. Then, every subsequent element is added semi-randomly, considering the cost of every remaining option. The cost of adding an element is defined by the inverse of the sum of the distances to each point in the current solution. In this way, we favor a more dispersed set of points. The best solution of the N generations will be stored if it improves the current best solution.
- A local search phase in which the algorithm tries to improve the current best solution by exploring its neighborhood of solutions. For generating the neighborhood, we take the indices of each point in the current solution and displace them randomly and circularly, one value up, down or maintaining the value. With the new indices, we can find a neighboring set of points. In all iterations, M local solutions are generated. If none is better than the current solution, the search stops. Else, the best replaces the present, and the exploration continues up to the maximum depth.

Once the algorithm reaches the maximum number of iterations, the center and radius of the sphere described by the best solution can be obtained. Note that we define the best solution as the one that allows obtaining a sphere that minimizes Equation 2, where $\dot{x}$ is the center of the sphere, r is the radius, C is the set of all cameras described by a look at vector $\vec{v}$ and a position $\dot{p}$, and w is defined in Equation 1. It is important to point out that, to avoid the sphere growing excessively, the radius of any solution is limited to be considered a valid solution. In our experiments, we have fixed the number of iterations to 1000, N to 20, M to 60, max depth to 50, α to 0.6, and the maximum allowed radius to double the maximum distance between cameras.

$$c(\dot{x}, r) = 0.4 \cdot \sum_{(\vec{v},\dot{p})\in C} |\| \dot{x} - \dot{p} \| - r| + w(\dot{x})^{-1} \qquad (2)$$

Algorithm 1 GRASP Sphere Estimation

```
1: function sphereEstimation(points, dirs)
2:     distances ← distanceMatrix(points)
3:     max_r ← 2 · max(distances)
4:     best_sol ← ∅, best_cost←∞
5:     for _ ← 1 to max_iterations:
6:         for _ ← 1 to N:
7:             sol ← {random(points)}
8:             for _ ← 1 to 3:
9:                 cands ← points ∉ sol
10:                costs ← dists(cands, points)^-1
11:                cmin ← min(costs)
12:                cmax ← max(costs)
13:                cands ← {c ∈ cands | costs[c]
≤ cmin + α · (cmax−cmin)}
14:                sol ← sol ∪ {random(cands)}
15:                if cost(sol) < best_cost and
radius(sol) < max_r:
16:                best_sol ← sol
17:                best_cost ← cost(sol)
15:        for _ ← 1 to max_depth:
16:            neighs ← getNeighbors(best_sol)
17:            for sol in neighs:
19:                if cost(sol) < best_cost and
radius(sol) < max_r:
20:                    best_sol ← sol
21:                    best_cost ← cost(sol)
22:                else:
23:                    break
24:    return sphere(best_sol)
```