3D Reconstruction from single RGB Images

Guided research project supervised by Prof. M. Nießner

Alexander Sheldrick
TUM
alexander.sheldrick@tum.de

Abstract

In this work we present an investigation of recent advances for implicit representations of 3D shapes for the task of 3D reconstruction from single images, and present a model that naturally extends to prediction of full color RGB meshes. In this context, we present SOTA results IoU and L2-Chamfer distance reconstruction metrics for the ShapeNet-cars dataset, and show qualitative results for its full color mesh outputs.

The presented architecture relies local multi-scale 2D and 3D feature extraction, and on incorporating recent advances for training implicit functions, especially the 1-Cycle policy [17] and Fourier embedding of coordinates [18]. Furthermore, the contribution of different feature modalities (2D images, 3D colored density voxels) are quantitatively and qualitatively evaluated. Using multiple feature modalities shows distinctive advantages over related works, i.e. higher visual clarity of reconstructed meshes and lower memory requirement during training.

1. Introduction

Extracting precise geometry from singular views is simultaneously a problem of immense popular interest, and an ill posed one. Indeed, "the inherent ambiguity in depth perception, the clutter and complexity of real-world environments make it still challenging to fully recover the scene context (both semantics and geometry) merely from a single image" [11]. In recent years however, with an increasing need for digital assets, the topic has received a surge of interest.

In 2019, with different contemporaneous works [9, 12] presenting implicit functions as a promising alternative to classical explicit modeling of 3D scenes, whose memory requirements scale unfavourably with their resolution, a new line of research opened up. Rather than explicitly representing the scene, a neural network, conditioned on shape or more general features, implicitly regresses an occupancy or signed distance value for a query-point. The actual geometry is then extracted in post-processing by sampling over a dense grid and applying e.g. marching cubes.

Implicit representations of 3D geometry and material properties have achieved state of the art results on staple data-sets such as ShapeNet [1]. These methods are of special interest, as they promise to solve memory and resolution issues of discrete occupancy methods [2]. Lately, with the success of differential rendering, and community efforts to make these methods simple to use and easy to access [14], there has been a heavy focus on stereo reconstruction of scenes and modeling of view dependant effects via neural radiance fields (NERF) [10].

In this work we are taking a step back and focus on the 2019 trend of single view 3D reconstruction via implicit functions without explicit rendering, but allow for given ground truth depth map and camera parameters (I, E), which could be readily extracted from modern consumer cameras, or available from from high quality scans. In the case that these parameters are absent, there is a rich body of work to infer these parameters for uncalibrated images [20, 7], but that is not the scope of this work.

Our contributions include the application and implementation of modern best practices for the training of neural implicit functions on a well studied dataset, and an investigation on how different feature modalities available from single views help to address the correspondence problem, which we aim to solve by employing 2D local (projected) multi-scale features. Further, we explore the impact of supplying additional 3D features available from the view-point, i.e. visible points, reprojected and voxelized. Lastly, we investigate whether color information can provide additional useful features for 3D reconstruction, and if not, if the performance is substantially impacted by having the network infer color as an additional constraint, or if a colored 3D consistent mesh can be extracted at little extra cost.

2. Related Works

Works addressing the problem of 3D reconstruction can be broadly categorized in their respective surface representations: explicit representations, i.e. **meshes**, **voxels** and **pointclouds**, and **implicit representations** such as functions and MLPs.

Voxel-based 3D representations are the natural extension from 2D images to 3D space. They allow for usage of much of the same toolkit that ushered in the deep learning revolution and are most commonly used for 3D generation and reconstruction [19], but too suffer from the curse of dimensionality, i.e. a memory footprint that scales cubically with resolution, usually restricted to a maximum of 128^3, and a generally expensive approach to encoding empty space. Multi-resolution approaches such as Octrees allow for a smarter use of available memory and for grids of sizes of up to 256^3, but are generally more complicated to implement and require multiple passes over the input.

Mesh based representations have also been considered as output representations of 3D reconstruction [5, 8]. Unfortunately most approaches are prone to generating self-intersecting meshes, are limited to simple mesh topology, for which they require a template from the same object class to deform, and do not produce high-quality reconstructions.

Pointclouds have shown great success for classification and semantic segmention [13], but the earlier works did not consider the information contained within the metric distances of points and empty space. Unlike voxels, the idea is to apply a fully connected network to each point followed by a global pooling operation to ensure permutation invariance. Further work has sought to keep the information lost by these operations through hierarchical grouping of points by their metric proximity [16], but contrary to voxels and meshes, they still require complicated post-processing to extract geometry.

Implicit functions: In contrast to aforementioned approaches, implicit modeling of occupancy- and truncated signed distance fields (TSDF) ”lead to high resolution closed surfaces without self-intersections and do not require template meshes from the same object class” [9], and have recently achieved state-of-the-art 3D reconstruction results.

3. Method

See fig. 1 for a qualitative introduction to signed distance fields. We assign to every point $\mathbf{p}$ a value S indicating the metric distance to the closest surface, and a sign, to discriminate points lying inside the object (red) or outside the object (blue). The zero level-set, i.e. the objects surface, is depicted in white.

The network, conditioned on some form of input, is taught to output a scalar value for every query point. The reconstruction is then done by point-wise SDF prediction on a 3D query grid. With a value for every point on the grid, a mesh can be extracted via marching cubes in post-processing. In our work, the task is to reconstruct colored 3D meshes from a single RGB image. Given a point in 3D space, the presented neural network implicitly regresses 4 values (S, RGB) conditioned on an image (2D features) and optionally conditioned on voxelized points from the visible surface of the depicted object (3D features).

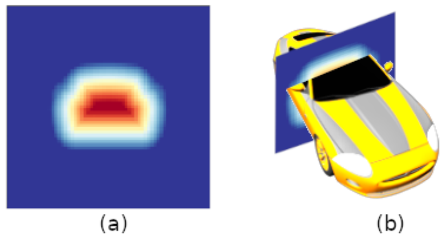

Figure 1. Depiction of a signed distance field in a scene. (a) Cross-section through object center. Note that the object surface traces the white, zero-level outline. (b) SDF portrayed within object in 3D space. Negative values in red (inside: $S < 0$), zero in white (object surface: $S = 0$) and positive in blue (outside: $S > 0$).

3.1. Fourier Feature embedding

Fourier embedding [18] have shown that mapping low dimensional coordinates to a high dimensional space via a simple Fourier mapping allows networks to learn high-frequency functions in low-dimensional problem domains. Preliminary testing on the proposed architecture has shown an enormous speedup for training until convergence with a simultaneous increase in performance. While we refer to the aforementioned work for a detailed explanation, the simple explanation is that the network learns a shift equivariant prior for point coordinates. We therefore map every 3D point $\mathbf{p}$ to a higher dimensional feature space with the following point-embedding transformation γ ($\mathbf{p}$):

$$\gamma(\mathbf{p}) = [cos(2\pi\mathbf{B}\mathbf{p}), sin(2\pi\mathbf{B}\mathbf{p})], \tag{1}$$

$\mathbf{B}$ denotes a random Gaussian matrix whose entries are drawn independently from a normal distribution N(0, σ^2). The two hyper-parameters for this transformation are the embedding size, 256 for all our experiments, and the scale parameter σ, which we set to 0.8 from initial experiments. The magnitude of the scale parameter acts like a low-pass filter, the higher the parameter is set, the better it can learn (and over-fit) to high frequency signals. Setting this parameter too low leads to overly smooth geometries, too high and the network learns high frequency artifacts, over-fitting to high frequency noise in the training data.

3.2. Multi-Scale feature extraction

The goal of this network is to learn a function that, given the (optional) voxel-grid $\mathbf{X}$, image $\mathbf{I}$ and a point $\mathbf{p}$, determines the RGB color $\mathbf{C}$ and TSDF value S of the point $\mathbf{p}$. To this end we extract features of images (2D) and (optionally) voxel grids (3D). Inspired by the success of works using local multi-scale features, namely [2, 20, 15], we employ local multi-scale features. The specific differences between 2D and 3D feature extraction are mentioned in the following paragraphs.

During ablations, different backbones for the 2D encoder and different 3D encoder architectures (i.e. a simple PointNet [13] or standard 3D convolutions without point subsampling) have performed marginally worse, but the proposed architecture is robust to the exact choice of encoder backbone.

Feature Encoding and extraction: We apply convolutions on the ND feature grid (i.e. an image or voxel-grid), each of which is followed by a down-scaling layer. At every down-scaling level, we sub-sample and interpolate the feature grid at the (projected-) point locations. Features from every scale are then concatenated to obtain a set of features at multiple scales. Now every feature grid $\nu \in \{\mathbf{X}, \mathbf{I}\}$ is encoded by encoder $\mathbf{G_i}$ via the features $\mathbf{F_k}(\mathbf{p})$ sub-sampled at $\mathbf{p}$ at stage k. The feature vector encoded by $\mathbf{G}$ has the following local and global multi-scale features that encode the global and local structure of the scene and is of the shape:

$$G_i(\nu) := [\mathbf{F_1}(\mathbf{p}), .., \mathbf{F_n}(\mathbf{p})], \tag{2}$$

where k denotes the down-sampling stage at which the feature was extracted and $\mathbf{F}_k \in \mathcal{F}_k^{K\times K}$ (for images), or $\mathbf{F}_k \in \mathcal{F}_k^{K\times K\times K}$ respectively for voxels, the local sub-sampled feature vector of $\mathbf{p}$ at k of decreasing resolution $K = \frac{N}{2^{k-1}}$, and variable channel dimensionality $F_k \in \mathbb{N}$ at each stage $\mathcal{F}_k \subset \mathbb{R}^{F_k}$.
During ablations, we investigate the impact of different feature modalities, and the performance in- or decrease for removing the 3D encoder, or reducing the available information.

Feature extraction 2D: We use a ResNet50 [6] without pre-training, but in general the style of architecture is similar to the encoder used by authors of DISN [20]. 2D features are extracted locally, or rather via projection, at multiple scales. Concretely, we perspectively project the 3D point into the (image-) feature plane via given GT camera intrinsics and extrinsics. Point- and image-coordinates are normalized to NDC space, i.e. $\{x_{NDC}, y_{NDC}\} \in [-1, 1]$. After every down-sampling layer, for every feature map and every point, we sub-sample the feature map at the projected 2D point location on the image plane. To reduce the total amount of 2D-features extracted, the feature-maps at every scale are reduced along the feature map dimension with

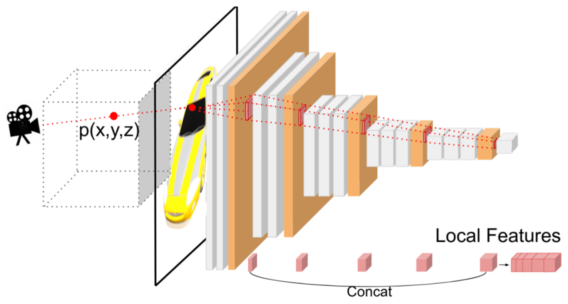

Figure 2. Overview of proposed 2D encoder with projective feature sub-selection. Image and encoder adapted from [20]. At every scale, feature maps are sub-selected via 3D-2D point projection onto the image plane and extracted along the feature-channel dimension. These per scale features are finally concatenated and passed to a decoder along with embedded coordinates and (optional) 3D features.

1D convolutions and ReLU activations before concatenating them. In contrast to DISN, the last feature-map is not flattened and concatenated: in our experiments the down-scaled sub-sampled features provided enough of the local neighborhood to capture global details in aggregate.

Feature extraction 3D: We use a similar architecture as proposed by the authors of IF-Net [2]. In contrast to IF-Net, we only extract the features at the location of the query point $\mathbf{p}$ and not its Cartesian neighborhood. The approach of extracting also a local neighborhood of interpolated features has led to quick over-fitting and limited success in the proposed architecture.

Feature decoding: The 2D (and 3D) features encoded by $\mathbf{G_i}$ extracted at point $\mathbf{p}$ are concatenated with the Fourier embedding $\gamma(\mathbf{p})$. This point-specific feature vector is fed into a five-layer fully connected network $f(\cdot)$, with two additional separate decision-heads, one for SDF and RGB each. In principle, the structure of the decoder is similar to that of [12]: the feature vector is injected at Layer1 and re-injected at Layer3 by concatenating it with the output of Layer2. Some intermediate layers of the original DeepSDF architecture were pruned as they had no tangible effect on performance for our dataset in preliminary experiments. Every layer but the decision-heads is activated with ReLUs, initialized with the asymmetric Kaiming initialization and weight normalized, which showed improved training stability over BatchNorm. The decision heads are activated by tanh (S) and sigmoid (RGB) respectively and together determine the 4 tuple (S,R,G,B), i.e. the TSDF and RGB values at $\mathbf{p}$:

$$f(G_i, \gamma(\mathbf{p})) : \mathcal{F}_1 \times ... \times \mathcal{F}_n \mapsto [S, R, G, B] \tag{3}$$

3.3. Method Training

During training, the network is provided by the coordinates (XYZ) of a point in canonic Cartesian space, its color values (RGB) and its TSDF value (S) denoting the signed distance to the closest surface. (S,RGB) are only used as supervision for the loss formulation. Then, conditioned on a feature grid $\nu \in \{\mathbf{X}, \mathbf{I}\}$, i.e. an image and (optionally) a voxel-grid, the network predicts for any point coordinate its corresponding color and TSDF value.
During training, depending on the kind of experiment, we provide the network with different amounts of 3D information, these are, from least information to most information:

- Only conditioned on an image, no 3D information
- Visible points voxelized into a voxel-grid
- N surface points voxelized into a voxel-grid
- Ground truth 32^3 occupancy grid
- Ground truth 32^3 SDF grid

Projection of Image-points into 3D
As a preprocessing step, we reproject the 30 renderings, at regular intervals from the upper hemisphere, of each object into a dense pointcloud using the depth map and camera parameters. Depending on the task at hand, either we sub-sample (a) visible points (b) points from the pointcloud. These points are then binned in 3D bins of set size (32^3 for most experiments) and voxelized into (i) an occupancy grid, (ii) a color averaged density grid, where each voxel denotes the average color and the fraction of total points inside its volume.

Preliminary experiments showed that visible point clouds only occupy single digit percentages and below of the available voxel volume, the amount of occupied voxels inversely scaling with voxel resolution. So although one would expect a large benefit to increasing the resolution, the gains are severely diminished by having to reduce batch sizes due to increased memory cost of the voxels.

Supervision and loss: At training time, the network, conditioned on the encoded feature grid $G_i(\nu)$, predicts the 4 tuple (S, RGB) for every input point $\mathbf{p}$, and is supervised by the points respective ground truth values via scaled L1 mini-batch loss:

$$L_{\mathcal{B}}(\mathbf{p}|G_i(\nu)) = \sum_{p \in \mathcal{B}} \lambda_C \|C - C_{GT}\| + \lambda_S \|S - S_{GT}\| \tag{4}$$

Where $\mathcal{B}$ denotes the batch, C and S the predicted colors and TSDF values, and C_{GT} and S_{GT} the ground truth values for supervision. λ_C (=0.5) and λ_S (=10) are loss-scaling factors.

3.4. Method Inference

During testing, the network is evaluated on points on a grid of the desired resolution (256^3 in most cases). Contrary to explicit representations, we can query the network at arbitrary grid resolutions, although the inference time scales cubically with the resolution. The mesh is then extracted via marching cubes, and evaluated according to section 4.1.

For color inference, the vertices of the extracted mesh are passed back for RGB inference at the vertex coordinates, the vertex [N,3]-list is then updated to include vertex colors as a [N,6]-list. For qualitative results the vertex-colored mesh is then rendered from the input perspective and four preset novel view points.

4. Experiments

To set a scope for reasonable expectations, we benchmark the proposed architecture and training regiment on the same splits but with varying information provided to the model. We then compare the models performance against the SOTA. After setting an expectation for the model, we consider (1.) Point cloud completion and (2.) 3D reconstruction from images without unseen 3D information. In all cases, the model is conditioned on an RGB image and queried for a point in 3D space for the 4D tuple (S, RGB). The initial simplified watertight models were extracted from a TSDF at 256^3 voxel resolution and provided by [20]. From these simplified meshes, we re-compute TSDF at 32^3 resolution and train the model on these ground truth TSDF // Occupancy voxel-grids, to set an upper boundary for our expectations of the metrics, as the network simply has to learn a reasonable marching cubes approximation. Failure would be a good indicator for erroneous or misaligned training data or out of distribution samples. Indeed, after simplification many meshes are seen to show artifacts, but are not removed from the split to keep comparisons fair.

4.1. Experimental setup

Data: As dataset we use the cars subset of ShapeNet [1], which consists of textured, synthetic 3D car models. This dataset features a wide variety of cars, but also many erroneous meshes and an inconsistent approach to model-detail (some feature extreme dashboard-knob level of detail, some are minimalist without interior geometry).

Since our needs deviate from the related works, we employ a slightly modified procedure from that of DeepSDF and OccNet. Models are rendered into RGB images in blender from the upper hemisphere at 12° increments (30 images per model). Ground truth depth maps and camera parameters (I, E) are saved to extract colored pointclouds.

The visible object-pixels are re-projected using the camera parameters, depth- and alpha- channels. The reprojected point clouds are then merged across all views of an object to form a dense, colored pointcloud.

The simplified meshes provided by [20] are centered, scaled and aligned via ICP to the reprojected pointcloud using CloudCompare [3] - these meshes will later serve as GT to compute metrics and SDF values. From the colored pointcloud, a subset of 100k points is copied and perturbed with Gaussian noise of different σ, similar to the procedure of [2, 9, 12], but we additionally determine the RGB values of these new points via the average of their K nearest neighbors (K=2) in respect to the original pointcloud. The motivation of using an averaged KNN is the assumption that a smooth RGB scalar field might be easier to approximate by the model.

Lastly, for each point we calculate the corresponding truncated signed distance field values S $\in [-0.1, 0.1]$ respective to the aligned, simplified meshes. For the sake of comparison, we use the same split as IF-Net but additionally remove further 13 extreme outlier meshes from the training data, 2 from the validation split and 0 from the test set, to keep comparisons fair. The complete split set is available with the code online.

Metrics: We follow the lead of the authors of IF-Net to measure the reconstruction quality quantitatively and evaluate the meshes with these three established metrics: *volumetric intersection over union* (IoU) measuring the spatial overlap of the volumes (higher is better), *Chamfer-L_2 distance*, to measure the accuracy and completeness of the surface (lower is better), and *normal consistency* measuring the accuracy and completeness of the shape normals (higher is better).

Baseline: The main goal is to investigate impact and interplay of using different feature modalities in the task of 3D shape reconstruction. To this end, we first provide a sensible baseline to establish model capacity by providing it with near ground truth data. We will then compare to [20, 9, 2] - keep in mind however that our model has only trained on a single ShapeNet class, it is hard to gauge if training on the entire dataset would enhance or diminish the models performance.

Hyper-parameters: The authors of the 1-Cycle policy recommend setting the highest maximum learning rate that does not lead to complete divergence, we therefor set it to 2e-3. This policy seems fairly robust in our case, and all experiments between 1e-2 and 1e-3 have converged to the same basin of attraction with similar loss curves and loss values. The learning rate decay plays a secondary role to learning rate and has been kept at a constant $w_d = 10^{-4}/step$, we use the largest batch-size that fits the GPU (10 for 32 voxel resolution, 7 for 64).

Data augmentation and regularization: Although much work has been spent on providing robust, on the fly data-augmentation and regularization techniques such as the Eikonal regularizer [4], which forces the predicted SDF scalarfield to have a unit norm gradient everywhere, the implicit regularization properties of the 1-Cycle policy outperformed them and rendered them redundant in the proposed architecture. For that reason, we do not augment or regularize the network besides the aforementioned parameters for the 1C policy LR schedule over 80 Epochs.

Training speed: Thanks to Fourier embedding and the 1-Cycle policy, the training converges within 4 hours to lower values than manual setting of the learning rate. If the reader takes away one lesson from this report, its this: we cannot overemphasize the gain in time and performance by simultaneous usage of both the 1C policy and Fourier embedding for implicit regression tasks.

4.2. Setting expectations and ablations

To set a sense for the capacity of the network, we train it on the entire data-set and evaluate its performance conditioned on different amounts of available information (Table 1). The first two entries designate results when trained with a ground truth voxel grid of 32^3 resolution rather than a voxelized pointcloud, i.e. where GT-SDF designates an SDF grid and GT-OCC an occupancy grid, we compare with meshes extracted via marching cubes from 256^3 TSDF files, in principle the task of super-resolution. For the cases of (i) GT-SDF and (ii) we see that the model learns to infer (i) an adequate super-resolution marching cubes and (ii) voxel super-resolution, although the accuracy degrades as we offer the model less information.

Clearly the presented 30M parameter model has enough capacity to learn from the ca. 2000 car training samples and generalize to unseen models of cars. An important note is that while the average IoU for (i) lies at 93.4%, the median %IoU lies at 95.6, demonstrating that few, heavy outliers distort the overall results, a common trend visible in every experiment. Indeed these are the expected outliers of out of distribution items, i.e. cars with rolled down windows and detailed interior geometry or unrealistically elongated limousines.

4.3. Pointcloud completion

Next, we apply the network on the problem of completing sparse and dense point clouds - to this end we subsample 300 points (sparse) and 3000 points (dense) respec-

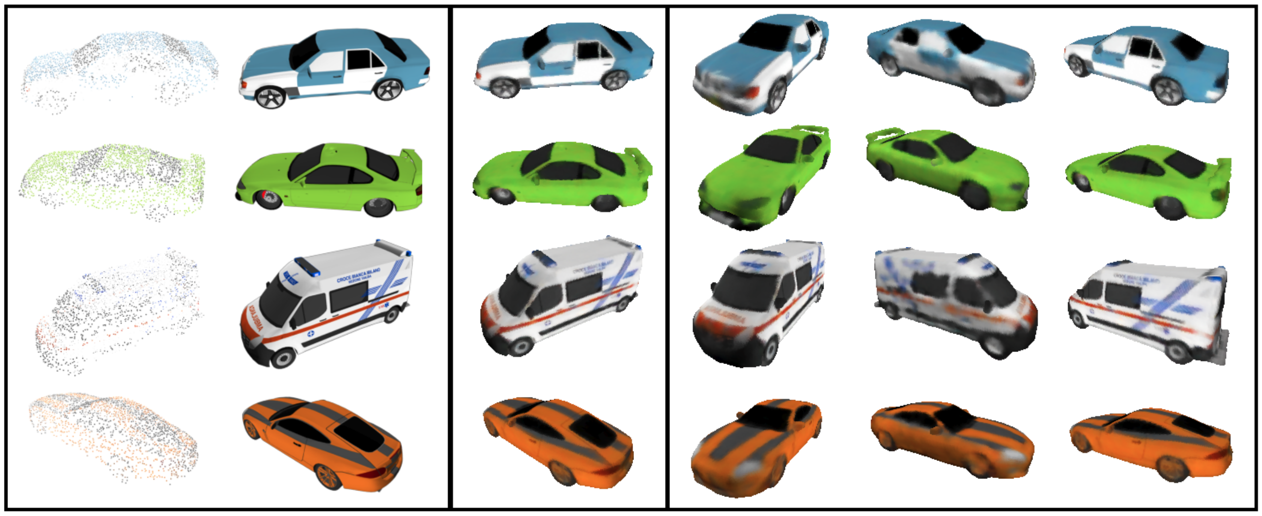

Figure 3. Qualitative results from the dense (3000 surface points, 64^3 voxelization) of ours^{64} (table 2) pointcloud reconstruction. The model, conditioned on (a: input) a pointcloud and rendered image of a textured model, reconstructs a vertex-colored 3D mesh rendered from (b) the input view and (c) novel views. Visual clarity is highest from the input perspective, showing that 2D features are richly extracted and lead to accurate reconstruction of the model. Novel views plausibly and faithfully reconstruct the original model, although with less visual clarity. Notice that the model can reconstruct fine details such as spoilers, tire-covers and mirrors.

	(%) IoU ↑	(%) Normals ↑	L2 ↓
GT-SDF	93.4	95.4	1.76
GT-OCC	91.0	92.9	1.85
2D only	74.5	84.4	8.22
Occ. Voxels	77.8	86.1	3.73
ours	78.7	86.2	3.05
ours*	**79.1**	**86.4**	**2.84**
m(ours)	86.3	87.8	2.05

Table 1. L2 signifies Chamfer-L2 distance * 10^{-4}. Values averaged over test-set. Models denoted with GT are supplied with 32^3 resolution voxel grids (TSDF, Occupancy), Model 2D receives no 3D shape-information and the other models receive the visible (up to $1.5 * 10^4$) voxelized view-points as input. For ours* we set λ_C = 0, demonstrating that color prediction has a minor adverse effect on performance. m(ours) indicates median performance of ours rather than average, indicating the presence of some extreme outliers that the model is not able to faithfully reconstruct.

tively from the (from all available views) re-projected surface point-cloud. These (colored) points are then voxelized at set resolution with 4 channels (density, RGB, 32^3 and 64^3) and fed to the 3D encoder. As such, it is no longer purely single-view reconstruction but an exploration into how additional conditional image features affect performance in regard to methods without [9, 2], whose works will serve as baseline. One notable difference besides the extra feature modality, is that our network runs at much lower voxelization resolution compared to the other works, who use a very fine and memory intensive gird of 128^3 voxels, a 16000% increase in memory cost, which gives the network a special advantage for the dense sampling case. Interestingly, we note large deviations between mean and median performance for our metrics, indicating extreme outliers and possible artifacts from the data processing. We note that e.g. %IoU in the dense case of ours^{64} is 82.6, while the median m(ours^{64}) %IoU is 90.1, outperforming all other methods with reduced memory footprint.

The explanation for why similar models evaluate so differently on the same split is obvious. While IF-Net samples directly from the simplified GT 3D mesh, inherently able to capture interior geometry and varying levels of detail, we only use points visible as pixels during rendering of the original textured ShapeNet model. Therefore, we cannot accurately consider interior geometry or general out of distribution outliers. It stands to reason that such samples could be found in the training data and degrade overall prediction quality. Still, we note the mean values by ours^{64} and median values by m(ours^{64}) separately and without emboldening to keep comparisons fair and clear.

4.4. Single view reconstruction

Lastly, we compare our network, conditioned on images, and reprojected image-points, with other works that rely on a single feature modality. Our method outperforms the other works by a significant margin, especially when considering it's median performance(see m(ours) table 3).

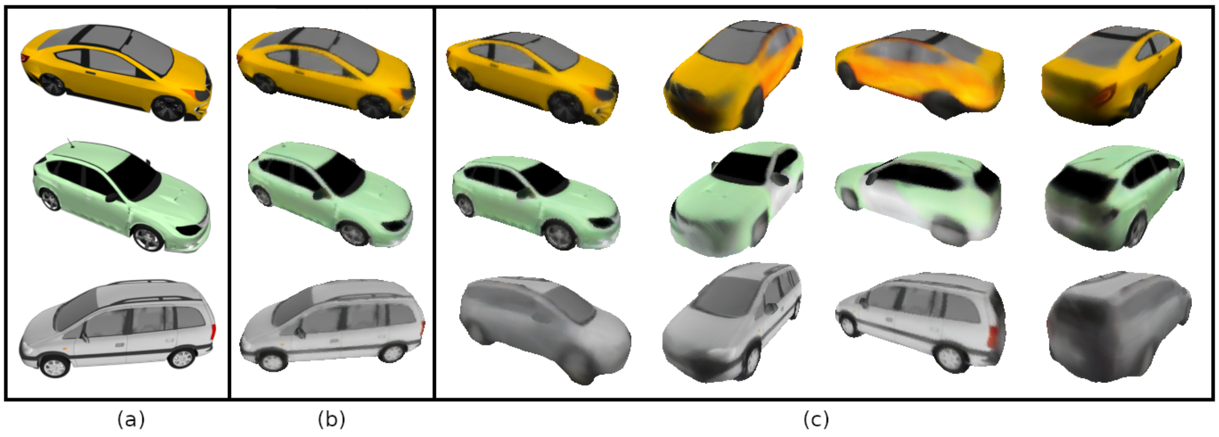

Figure 4. Qualitative results from the single view results (visible surface points are reprojected and voxelized at 32^3 voxelization) of ours (table 1). The model conditioned on (a: input) a rendered image of a textured model, reconstructs a vertex-colored 3D mesh rendered from (b) the input view and (c) novel views. Just like in previous experiments, visual clarity remains highest from the input perspective. Novel views plausibly and faithfully reconstruct the original model, but are more approximative and blurry in nature, with cone shaped artifacts emanating from the camera center towards the mesh. Notice that the model can reconstruct unseen fine details such as wheels, windows and mirrors.

	(%) IoU ↑		(%) Normals ↑		L2 ↓	
OccNet	73	72	88	88	30	40
IfNet	79	**88**	**90**	**95**	2	2
ours^{32}	79.7	81.4	86.7	88.0	2.2	1.8
ours^{64}	**80.3**	82.6	86.6	89.3	2.3	**1.5**
m(ours^{64})	87.2	90.1	87.2	90.5	1.8	0.96

Table 2. Left number indicates score from 300 points, right one from 3000. L2 signifies Chamfer-L2 distance * 10^{-4}, we compare with the results reported by the authors of [9, 2], who evaluated their results on a ≈26 000 item split, while ours are trained and evaluated only on the ≈2500 cars-subsplit. ours voxelizes points at 4 channels and 32^3, while the other works voxelize at 128^3, a 16000% memory increase. ours^{64} is identical to ours besides voxelizing at 4 channels and 64^3, demonstrating the improvement that finer voxelization offers, while m(ours^{64}) shows median instead of average performance among the test set, indicating heavy outliers.

Qualitatively, fig. 4, we see that here too the images from the view-perspective are faithfully reconstructed, matching those from the dense pointcloud reconstruction. However, novel view points of occluded sides show ray-like artifacts emanating conically from the camera center. This is clearly an artifact resulting from the projective feature sub-selection, as features from visible pixels are responsible for all points lying on a cone between camera center and 3D point. While the visual clarity of occluded sides is diminished over that of the dense reconstruction, the model manages to sensibly complete the models shape and is even able to infer details like window outlines, tire-covers and lights, and mostly continues the reconstruction in a reasonable color.

One common artifact is a strongly shaded back side, resulting from lighting placed in-front of the model during rendering, a procedure kept constant for all models. To avoid these kind of artifacts it would make sense to render models with per-model random or ambient lighting.

	(%) IoU ↑	(%) Normals ↑	L1 ↓
OccNet	73.7	85.5	15.9
DISN	77.0	n.A.	4.92
ours	**79.1**	**86.4**	**1.06**

Table 3. L1 signifies Chamfer-L1 distance * 10^{-2}, we evaluate our results wrt. the simplified high-resolution mesh provided by [20] and we compare with the results reported by [9, 20].

5. Discussion and Conclusion

In this work we demonstrate SOTA results for the reconstruction of vertex-colored meshes from ShapeNet cars. We find that multi-modal feature input positively affects reconstruction quality, and show lower memory requirements during training allowing for higher batch-sizes and faster convergence. We present a simple adaptation for the construction of datasets for vertex colored mesh inference in analogy to[12], and qualitatively present the models' high quality colored mesh outputs. However, as we are unable to find any positive correlation for our loss formulation with explicit RGB loss on IoU, we note that more work is required to establish a useful RGB-value informed loss.

Much time and effort has been spent on creating gradient step models, to circumvent differential rendering or ray-marching, but none were able to outperform the presented approach. The performance decrease, most likely caused by additional strain on the models capacity, i.e. forcing it to predict 300% more scalar values per point coordinate, can be minimized with the presented 4 channel (Density, RGB) pointcloud-voxelization, rather than voxelization into classical voxel grids. Another possible explanation for the degradation of performance using this loss formulation, is that the model is punished less for a warped geometry than wrong color values: the exact cause for the reduction in performance is therefore still unclear.

A natural and straight forward extension to the presented work is the extension to true stereo reconstruction, with multiple images as input. Exploratory results were presented in 4.3, and while point-cloud reconstruction results were satisfactory, especially in regard to the evaluated geometry metrics, the visual clarity was best from the image's point of view, highlighting how beneficial 2D features are for visual clarity in 3D reconstruction. An extension to true stereo could therefore substantially increase reconstruction results and visual clarity of mesh-colorization.

Limitations: The presented method requires a fairly simplistic dataset, clean exterior hulls of objects, and as it relies on surface-point re-projection, it requires calibrated depth and camera parameters. Out of distribution models with detailed interior geometry naturally perform badly with the presented evaluation scheme, causing mean and median performance to deviate strongly on ShapeNet-cars. In addition, the models' robustness to noisy real world data, or approximated camera parameters and depth maps, rather than the here used ground truth information, is still in question and requires further research.

References

[1] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 1, 4

[2] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion, 2020. 1, 3, 5, 6, 7

[3] Daniel Girardeau-Montaut. Cloudcompare, version 2.11.3, 2020, [gpl software], http://www.cloudcompare.org/. 5

[4] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes, 2020. 5

[5] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3

[7] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds, 2018. 1

[8] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose, 2018. 2

[9] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 5, 6, 7

[10] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1

[11] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image, 2020. 1

[12] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 1, 3, 5, 7

[13] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. 2, 3

[14] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1

[15] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization, 2019. 3

[16] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions, 2019. 2

[17] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018. 1

[18] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 1, 2

[19] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T. Freeman, and Joshua B. Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction, 2018. 2

[20] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction, 2019. 1, 3, 4, 5, 7