Black-Box Test-Time Shape REFINEment for Single View 3D Reconstruction

Brandon Leung  Chih-Hui Ho  Nuno Vasconcelos
University of California, San Diego
{b7leung, chh279, nvasconcelos}@ucsd.edu

Abstract

Much recent progress has been made in reconstructing 3D object shape from an image, i.e. single view 3D reconstruction. However, due to the difficulty of collecting large datasets in the wild with 3D ground truth, it remains a significant challenge for methods to generalize across domain, viewpoint, and class. Current methods also tend to produce averaged “nearest-neighbor” memorized shapes instead of genuinely understanding the image, thus eliminating important details. To address this we propose REFINE, a post-processing mesh refinement step easily integratable into the pipeline of any black-box method in the literature. At test time, REFINE optimizes a network per mesh instance, to encourage consistency between the mesh and the given object view. This, with a novel combination of losses addressing degenerate solutions, reduces domain gap and restores details to achieve state of the art performance. A new hierarchical multiview, multidomain image dataset with 3D meshes called 3D-ODDS is also proposed as a uniquely challenging benchmark. We believe that the novel REFINE paradigm and 3D-ODDS are important steps towards truly robust, accurate 3D reconstructions.

1. Introduction

Single view reconstruction (SVR) aims to generate the 3D shape of an object from an image of it. SVR networks are usually learned from datasets with many views of different objects. While ideally these datasets should be large, composed of real images, cover many object classes with many views per object, and come with corresponding 3D ground truth, this is extremely difficult to achieve in practice. As a result most methods are trained on renders of synthetic 3D CAD models [34, 51, 57], or only applicable to a specific object class per network, such as birds [21, 28], beyond which they cannot generalize. The goal of learning universal SVR models, applicable to any object, remains a significant challenge. This is compounded by the difficulty of generalizing across domains. As illustrated in Figures 1 and 4, the application of an SVR network trained on ShapeNet [3] to real images leads to severe reconstruction failures. Even with 3D synthetic data, current methods tend to recognize the object, perform a “nearest-neighbor” search for a “mean class shape” memorized during training [46], and make slight adjustments that are usually not enough to recover intricate shape details. As shown in Figure 2, while reconstructions (bottom row) reflect the category of the object in the image (top row), details that determine fine-grained identity are usually lost.

Test-Time Shape Refinement (TTSR) [39] is a promising solution to these problems. It poses the question of whether the SVR network reconstruction can be improved upon at test-time by providing some additional information about the object, e.g. a silhouette. TTSR has at least two interesting properties. First, because it is a test-time operation, it only requires relatively small datasets to design and evaluate. This enables the collection of datasets in the lab, to explicitly test how TTSR can enhance the robustness of SVR. Gains for both accuracy and robustness are shown for REFINE (right).
main dataset called the 3D Object Domain Dataset Suite (3D-ODDS), containing 71,496 real images of objects collected under many different controlled poses and domains, along with their scanned 3D meshes (Figures 1, 8). A second interesting property of TTSR is that it provides the opportunity to exploit optimization at test time, instead of just a forward pass, to improve reconstruction results. This was shown in [39] but posed as a fine-tuning problem, where parameters of their network are adjusted to achieve this goal.

In this work we ask the broader question of whether TTSR can be performed by an external network which refines the mesh shape produced by the SVR network, a posteriori as illustrated in Figure 1, and is applicable to any SVR method. We denote this as black-box TTSR (bbTTSR). There are several advantages in bbTTSR over TTSR. First, it is agnostic to the SVR architecture. As demonstrated in this work, it can be equally easily applied to approaches like DeepSDF [37] or OccNet [33] which use implicit functions, Mesh R-CNN [12] or Pix2Vox [57] which have voxel-based components, and AtlasNet [13] which represents meshes using atlas surface elements. Second, because it does not even require knowledge of the inner workings of the SVR network, it supports applications where the latter is provided by a third party and not publicly available. Finally, unlike network finetuning, bbTTSR encourages the joint development of networks and losses that explicitly address the degenerate solution tendencies and extreme data efficiency challenges seen in test-time refinement.

Given these potential advantages, we propose a RE-Fine INstances at Evaluation (REFINE) architecture for bbTTSR. REFINE utilizes a mesh feature encoder with a graph refiner network, trained using a novel combination of loss functions encouraging both silhouette consistency and confidence-based mesh symmetry. We then combine existing datasets [3, 5, 43] with 3D-ODDS to produce an experimental framework to test how bbTTSR methods improve the effectiveness and robustness of SVR. These extensive experiments rigorously show that REFINE improves the reconstruction accuracy of many SVR networks as measured by several metrics, both in the presence and absence of domain gap between training and inference data, for both synthetic and real images, across diverse object classes/views.

Overall, this work makes four main contributions. The first is bbTTSR, i.e. using external post-processing network at test time, to improve the quality of meshes produced by SVR methods. The second is the 3D-ODDS dataset. This is the first SVR dataset to deliberately target questions such as robustness of SVR to domain shift, using real world images of objects from many classes, and precise control of object pose. Third, we propose the first solution to the challenging bbTTSR problem with REFINE, which successfully suppresses degenerate solutions to provide performance gains. Finally, extensive experiments show that REFINE outperforms the state of the art in TTSR, is an effective solution for bbTTSR, enhancing performance of many SVR networks under many experimental conditions.

2. Related Work

Single View 3D Reconstruction. While many SVR methods have been proposed, they all suffer from the inconsistencies of Figure 2, and can benefit from REFINE. The main 3D output modalities are voxels, pointclouds, and meshes. Voxel methods typically encode an image into a latent vector, then decoded into a 3D voxel grid with upsampling 3D convolutions [5, 57]. Octrees can enable higher voxel resolution [35, 52]. Pointclouds have been explored as an alternative to voxels [8, 29] but usually require voxel or mesh conversion for use by downstream tasks. Among mesh methods, some learn to displace vertices on a sphere [22, 51] or a mean shape [21] to reconstruct. Current state of the art methods rely on an intermediate implicit function representation to describe shape [11, 34, 36, 37, 58], mapped into a mesh by marching cubes [33].

Methods also vary by their level of supervision. Most are fully supervised, requiring a large dataset of 3D shapes such as ShapeNet [3]. Recently, weakly-supervised methods have also been introduced, using semantic keypoints [21] or 2.5D sketches [53] as supervision. Alternatively, [28] has proposed a fully unsupervised method, combining part segmentation and differentiable rendering. Few-shot is considered in [35, 49] where classes have limited training data. Domain adaptation was explored in [38], which assumes access to data from a known target domain.

Despite progress in single view 3D reconstruction, questions arise on what is actually being learned. In particular, [46] shows that simple nearest-neighbor model retrieval can beat state of the art reconstruction methods. This raises concerns that current methods bypass genuine reconstruction, simply combining image recognition and shape memorization. Such memorization is consistent with Figures 1 and 2, leading to suboptimal reconstructions and inability to generalize across domains. It is likely a consequence of learning the reconstruction network over a training set of many instances from the same class. In contrast, REFINE uses test-time optimization to refine a single shape, encouraging consistency with a single silhouette. This prevents memorization, directly addressing the concerns of [46]. It also...
makes REFINE complementary to reconstruction methods and applicable as a postprocessing stage to any of them.

**Test-Time Optimization.** Test-time training [44] or optimization usually exploits inherent structure of the data in a self-supervised manner, as no ground truth labels are available. In [44], an auxiliary self-supervised rotation angle prediction task is leveraged to reduce domain shift in object classification. The same goal is achieved in [50] by test-time entropy minimization. Meanwhile, [47] uses self-supervision at test time to improve human motion capture. Additionally, interactive user feedback serves to dynamically optimize segmentation predictions [19, 40, 42].

**Test-Time Shape Refinement.** Test-time shape refinement (TTSR) requires a postprocessing procedure to improve the accuracy of meshes produced by a reconstruction network. Most previous approaches are white-box methods, i.e., they are specific to a particular model (or class of models) and require access to the internal workings of the model. Examples include methods that exploit temporal consistency in videos, akin to multi-view 3D reconstruction [27, 30]. [27] requires the unsupervised part-based video reconstruction architecture proposed by the authors and [60] optimizes over a shape space specific to their architecture for zebra images. Among white-box methods, the approach closest to REFINE is that of [39], which finetunes the weights of the reconstruction network at test time, to better match the object silhouette. But even this method is specific to sign distance function (DeepSDF [37]) networks. By instead adopting the black-box bbTTSR paradigm, where the mesh refinement step is intentionally decoupled from the reconstruction process, REFINE is capable of learning vertex-based deformations for a mesh generated by any reconstruction architecture. Our experiments show that it can be effectively applied to improve the reconstruction performance of many networks and achieves state of the art results for test-time shape refinement, even outperforming [39] for DeepSDF networks. In summary, unlike prior approaches, REFINE is a black-box technique that can be universally applied to improve reconstruction accuracy, a posteriori.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.9</td>
<td>12.7</td>
<td>13.2</td>
<td>18.8</td>
<td>20.3</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. REFINE improves reconstruction by introducing only a small number of parameters, relative to popular networks.

In [39], it was explored whether or not the use of additional auxiliary test-time information could help mitigate these problems. Their approach involved optimizing the parameters of $S$ on-the-fly during inference, given a coarsely reconstructed mesh $S(x) = M_c = (V_c, E_c) \in \mathcal{M}$, an object silhouette $x_s$, and the camera pose $p$. We call this problem setting test-time shape refinement (TTSR), and we propose to investigate an alternative class of black-box TTSR (bbTTSR) solutions which abstracts shape refinement from any black-box reconstruction network $S$. This consists of introducing a dedicated refinement network $R$, external to $S$, to implement the shape refinement. $R$ is trained at test-time, so that the 3D mesh $R \circ S(x)$ more accurately approximates the object shape, as illustrated in Figure 1. We denote the approach as REFINe and as the REFINEment network. In this formulation, $R$ predicts a set of 3D displacements $V_{dis} \in \mathbb{R}^{N \times 3}$ for the vertices in $V_c$. These are used to compute the REFINEd mesh $M_r = (V_r, E_r) = (V_c + V_{dis}, E_c)$ whose render best matches the silhouette $x_s$. Displacements are complemented by a set of symmetry confidence scores $V_{sConf} \in [0, 1]^{N \times 1}$, which regularizes $V_{dis}$ through a symmetry prior, as detailed in Section 3.3.

Several advantages derive from bbTTSR’s abstraction of refinement from reconstruction. First, REFINE is a black-box technique, applicable to any network $S$. In fact, the network does not even have to be available, only the mesh $S(x)$, which gives REFINE a great deal of flexibility. For example, while MeshSDF can only be used with DeepSDF networks, REFINE is applicable even to voxel and point-cloud reconstruction methods, by using mesh conversions [1, 2, 23, 24]. This property is important, as different methods are better suited for different downstream applications. For example, implicit methods [34, 37] tend to produce the best reconstructions but can have slow inference [37]. Meanwhile, AtlasNet [13] is less accurate but much more efficient, and inherently provides a parametric patch representation useful for downstream applications like shape correspondence. Second, because the refinement network $R$ and loss functions used to train it are independent of the reconstruction network $S$, they can be specialized to the test-time shape refinement goal. This is important because the regularization required to avoid degenerate solutions for the learning of $R$, which is based on a single mesh instance, is quite different from that of $S$, which is learned from a large

$$S : \mathbb{R}^{W \times H \times 3} \rightarrow \mathcal{M} \in \mathbb{V} \times \mathcal{E},$$

where an RGB image $x \in \mathbb{R}^{W \times H \times 3}$ of width $W$ and height $H$ is mapped to a mesh $M = (V, E) = S(x)$ by a reconstruction network, where $V \in \mathbb{V} \subset \mathbb{R}^{N \times 3}$ is a set of vertices and $E \in \mathcal{E} \subset \mathbb{B}^{(2)}$ a set of edges. $\mathbb{B}$ is a boolean do-
dataset. In REFINE, several loss functions tailored for test-time training are proposed to achieve this. We also design $R$ to be much smaller than $S$, to lessen the additional computational overhead for refinement. As shown in Table 1, the REFINE network is at least 10 times smaller than most currently popular reconstruction networks.

### 3.2. Architecture

Figure 3 summarizes the REFINE architecture. This is a combination of an encoder $E$ and a graph refiner $G$ followed by 2 branches $B_{dis}$ and $B_{sConf}$, which predict the vertex displacements and vertex confidence scores respectively. The encoder module $E$ contains $L$ neural network layers of parameters $\{\theta_i\}_{i=1}^L$, takes silhouette $s_{x_0}$ as input, and outputs a set of $L$ feature maps $F(x_s; \Theta = \{\theta_j\}_{j=1}^J) \in \mathcal{R}^{W_1 \times H_1 \times C_1}$, of width $W_1$, height $H_1$ and $C_1$ channels. In our implementation, $E$ is based on ResNet [15]; $L$ is set to 2, where $C_1 = 64$ and $C_2 = 128$.

Given feature map $F(x_s; \Theta_i)$, the feature vector $f^v_i$ corresponding to a vertex $v$ in $M_c$ is computed by projecting the vertex position onto the feature map [12, 51],

$$f^v_i = \text{Proj}(v; F(x_s; \Theta_i), p) \in \mathcal{R}^{C_1},$$

where $p$ is the camera viewpoint and $\text{Proj}$ a perspective projection with bilinear interpolation. Vertices are represented at different resolutions, by concatenating the feature vectors of different layers into $F^v = (f^v_1, \ldots, f^v_L)^T$. The set $\{F^v\}_{v=1}^V$ of concatenated feature vectors extracted from all vertices is then processed by a graph convolution [25] refiner $G$, of parameters $\phi$, to produce an improved set of feature vectors $\{H^v\}_{v=1}^V = G\{(F^v)_{v=1}^V; \phi\}$. Finally, this set is mapped into the displacement vector $v_{dis}$

$$V_{dis} = B_{dis}\{(H^v)_{v=1}^V; \psi_{dis}\},$$

by a fully connected branch $B_{dis}$ of parameters $\psi_{dis}$ and into the confidence vector

$$V_{sConf} = B_{sConf}\{(H^v)_{v=1}^V; \phi; \psi_{sConf}\},$$

by a fully connected branch $B_{sConf}$ of parameters $\psi_{sConf}$. Overall, the REFINE network implements the mapping

$$R(x_s, M_c; \{\Theta_i\}, \phi, \psi_{dis}, \psi_{sConf}, p) = (V_{dis}, V_{sConf}).$$

### 3.3. Optimization

The REFINE optimization combines popular reconstruction losses with novel losses tailored for test-time shape refinement. In what follows we use $M^p$ to denote a differentiable renderer [22, 31] that maps mesh $M \in \mathcal{M}$ into its image captured by a camera of parameters $p$. We also define sets $V_s^x$, $V_{dis}$, and $V_{sConf}$ of size $N$, constructed with the rows of $V_s$, $V_{dis}$, and $V_{sConf}$ respectively. A set of popular reconstruction losses are used in REFINE, as follows.

**Silhouette Loss:** Penalizes shape and silhouette mismatch

$$L_{Sil} = L_{BCE}(x_s, \gamma(M^p)), \quad (6)$$

where $\gamma(M^p)$ is the silhouette of the rendered shape, using the 2D binary cross entropy loss

$$L_{BCE}(a, b) = \sum_{ij} a_{ij} \log(b_{ij}) + (1 - a_{ij}) \log(1 - b_{ij}). \quad (7)$$

**Displacement Loss:** Discourages overly large vertex deformations, with

$$L_{Dis} = \sum_{v_i \in V_{dis}} \|v_i\|^2_2. \quad (8)$$

**Normal Consistency & Laplacian Losses:** $L_{NC}$ and $L_{LP}$ are widely used [7, 51] and encourages mesh smoothness.

A second set of losses is introduced to avoid degenerate solutions, namely overfitting to the input view during bbTSR. These leverage the structural prior that many real world objects are bilaterally symmetric about a reflection plane $\mathcal{Z}$. Symmetry has long been exploited in computer vision, graphics, and geometry [32]. Many methods (e.g. [34, 51]) learn symmetry implicitly from the training data. Since datasets like Shapenet [3] are composed primarily of symmetric objects, a learned bias towards symmetry is almost impossible to avoid. Symmetry can also be explicit, e.g. [59] predicts planes of symmetry given 2D images to improve monocular depth estimation, or used to regularize learning, e.g. with horizontal flips during training [54].

Rather than 2D images, we exploit 3D shape symmetry by imposing two test-time constraints on reconstructed 3D meshes: 1) object vertices should be symmetric, and 2) mesh rendered images should reflect this symmetry.
The loss is defined as
\[
\text{Vertex-Based Symmetry Loss:}\quad V_{\text{sym}} = \frac{1}{N_c} \sum_{i=1}^{N_c} \sigma_i \min_{v_j \in V_i} \| T v_i - v_j \|^2 + \lambda_{\text{Sym}} \ln \left( \frac{1}{\sigma_i} \right),
\]
where \( \sigma_i \) is the minimum confidence score over all vertices in the mesh. The first term penalizes distances between each vertex and its nearest neighbor upon reflection about \( Z \). This is weighted by the confidence score \( \sigma_i \), which is low for vertices that are asymmetric based on the object silhouette. The second term penalizes small confidence scores, preventing trivial solutions. The trade-off between these two terms is controlled by hyperparameter \( \lambda_{\text{Sym}} \in [0, \infty) \). As shown in Figure 6, scores \( \sigma_i \) are large except in areas of clear asymmetry.

**Render-Based Image Symmetry Loss:** Encourages image projections that reflect object symmetry. Given \( m \) camera viewpoints \( P_{\text{sym}} = \{ p_1, ..., p_m \} \), \( T \) is used to obtain differentially rendered pairs from symmetric camera viewpoints \( \{ (M_{p1}, M_{p1}^T), ..., (M_{pm}, M_{pm}^T) \} \), as shown in the rows of Figure 5. The loss is defined as
\[
L_{\text{sym}} = \frac{1}{m} \sum_{i=1}^{m} \sum_{j,k} \left[ (1 - \gamma(h(M_{p_j}^T)_{j,k} - \gamma(M_{p_j}^T)_{j,k}))^2 + \lambda_{\text{Sym}} \ln \left( \frac{1}{\sigma_{j,k}} \right) \right],
\]
where \( h(\cdot) \) is an horizontal image flip and \( j,k \) are image coordinates. Symmetry is enforced by minimizing the distance between the horizontal flip of each render \( M_{p_j}^T \) and the render \( M_{p_j}^T \) at the symmetrical camera viewpoint. This is akin to comparing a “virtual image” of what the mesh should symmetrically look like. Pixel-based confidence scores \( \sigma_{j,k} \) are used as in (9). However, they are not learned, but derived from the vertex confidences \( \sigma_i, i \in V_{\text{sym}} \) of (9) by barycentric interpolation on the mesh faces, where \( V_{\text{sym}} \) are mesh face vertices projected into pixel \( j,k \). **Overall Loss:** REFINEm is trained with a weighed combination of the six losses
\[
L_{\text{total}} = \lambda_{\text{Sil}} L_{\text{Sil}} + \lambda_{\text{Sym}} L_{\text{sym}} + \lambda_{\text{svr}} L_{\text{svr}} + \lambda_{\text{Disc}} L_{\text{Disc}} + \lambda_{\text{Rec}} L_{\text{Rec}} + \lambda_{\text{Lp}} L_{\text{Lp}}.
\]

\( L_{\text{Sil}} \) is the main driving factor to ensure input silhouette consistency, while other losses serve as regularizers to prevent degenerate solutions. Figure 7 shows that REFINEd shape quality tracks the evolution of this loss, for an airplane whose body has been truncated in the original reconstruction. As the REFINEm loss steadily decreases, the mesh progressively becomes more faithful to the input image; this is seen in the elongated body and corrected wing shape.

### 3.4. Implementation Details

Several details of our implementation are worth noting. In all experiments we used \( P_{\text{sym}} \) of 6 viewpoints, with azimuths in \( \{15^\circ, 45^\circ, 75^\circ\} \) and elevations in \( \{-45^\circ, 45^\circ\} \). The learning rate is 0.00007, \( \lambda_{\text{Sil}} = 10, \lambda_{\text{Sym}} = 80, \lambda_{\text{svr}} = 20, \lambda_{\text{Sym}} = 80, \lambda_{\text{Disc}} = 100, \lambda_{\text{Rec}} = 10, \) and \( \lambda_{\text{Lp}} = 10 \). Also, REFINEm supports a variable number of vertices per mesh, generally converges in 400 iterations, and takes only seconds to complete when performed in parallel. More details are given in the supplementary.

### 4. Multiview, Multidomain 2D & 3D Dataset

SVR is usually benchmarked on synthetic CAD datasets [3, 55] because these, albeit unrealistic, allow renders of images from many viewpoints. While real data can also be collected [4, 18, 26, 41], this has various difficulties resulting in datasets with different limitations. For example, Pascal3D [56] contains diverse real indoor/outdoor settings, but meshes are only approximations manually chosen from a CAD library. Pix3D [43] includes ground truth meshes but is relatively small and primarily depicts furniture in indoor locations with uncontrolled viewpoints. No existing real-world dataset enables systematic study of reconstruc-
In this work, we introduce the 3D-ODDS dataset to address these two fundamental challenges towards generalizable vision. 3D-ODDS contains DSLR-captured images of 331 objects from 16 different classes with dense pose coverage (72 azimuths, 3 elevations) for 216 images per object, and 71,496 images total. These images were used to generate 3D meshes for each object (331 meshes total, details in supplementary). Crucially, 232 of the objects can also be found in two real-world, multiview image datasets: OOWL [16] and OWILD [17]. They depict the same objects with 45° azimuth increments in different domains. OOWL images were collected using drone cameras during flight, OWILD in diverse indoor/outdoor locations with smartphones.

Note that the relatively small dataset size reflects the difficulty of real-world 3D data collection. While insufficient for large scale SVR network training, 3D-ODDS is ideally suited for tasks such as TTSR, domain adaptation, or few-shot learning, needed to translate shape reconstruction research into real applications. Using synthetic CAD datasets alone is also inadequate in achieving this goal. As illustrated in Figure 8, 3D-ODDS combines OTURN (our collected turntable images and 3D meshes) with OOWL and OWILD to study invariance to pose and image domain.

Datasets: Five datasets are considered, to carefully address these two fundamental challenges towards generalizable vision. We believe that 3D-ODDS (to be released publicly) will be an important testing ground to evaluate the real world robustness of SVR methods.
Table 3. Reconstruction accuracies with no domain shift. Top: single view reconstruction (SVR) networks. Bottom: test-time shape refinement (TTSR) methods. TTSR results presented by accuracy before → after refinement, with gain shown in parenthesis.

<table>
<thead>
<tr>
<th>Method</th>
<th>EM numerator</th>
<th>CD-δ2</th>
<th>Vol. IoU</th>
<th>F-Score</th>
<th>Vol. IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AtlasNet</td>
<td>8.0</td>
<td>13.0</td>
<td>89</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>Mesh R-CNN</td>
<td>4.2</td>
<td>10.3</td>
<td>90</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>DISN [38]</td>
<td>3.4</td>
<td>8.0</td>
<td>93</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>MeshSDF [39]</td>
<td>2.6</td>
<td>9.7</td>
<td>91</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>TTSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MeshSDF [39]</td>
<td>3.0±0.25</td>
<td>12.0±0.05</td>
<td>91±205</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>REFINEd OccNet [34]</td>
<td>2.9±0.23</td>
<td>12.2±0.75</td>
<td>91±96</td>
<td>57±59</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. REFINEd in the presence of mild domain shift, namely RerenderedShapeNet reconstructions by ShapeNet trained networks. Gains occur under all networks, classes, and metrics.

<table>
<thead>
<tr>
<th>Method</th>
<th>EM denominator</th>
<th>CD-δ2 denominator</th>
<th>F-Score denominator</th>
<th>Vol. IoU denominator</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFINEd OccNet [34]</td>
<td>4.3→3.3</td>
<td>(-1.0)</td>
<td>11.5</td>
<td>(4)</td>
</tr>
<tr>
<td>REFINEd AtlasNet [13]</td>
<td>6.2→4.9</td>
<td>(-1.3)</td>
<td>23.2</td>
<td>(14)</td>
</tr>
<tr>
<td>REFINEd MeshSDF [39]</td>
<td>6.5→3.3</td>
<td>(-3.2)</td>
<td>21.3</td>
<td>(15)</td>
</tr>
</tbody>
</table>

Figure 9. Leftmost column: input image and mesh. Other columns: REFINEd improves with an increasingly larger set of losses (left to right), Best viewed enlarged.

The middle section of the table uses all losses, ablating architectural components by removing both encoder E and graph refiner G (directly optimizing the mesh deformation with no network), removing only E, and randomly initializing E. These refinements improve on the original mesh, but underperform the implementation of REFINEd using G and E with ImageNet weights (row 5). We hypothesize this is because E and G provide a useful high dimensional projection for mesh deformation, similar to the inductive bias from architectural parameterization studied in [14, 48].

The bottom three rows of Table 2 use ShapeNetAsym to study the effect of asymmetry on REFINEd performance. The sixth row is not refined. The seventh row shows that when the confidence scores of (9) and (10) are removed (by setting $\lambda_{Sym,B} = 1$, in which case the confidence scores become approximately 1) the refinement of asymmetrical meshes is significantly less accurate than that of the default configuration (eighth row, $\lambda_{Sym,B} = 0.0005$). It can also be seen that, when confidence scores are used, the reconstruction quality is significantly superior to that of the original meshes. In summary, the proposed confidence mechanism enables effective REFINEd of non-symmetric objects.

Figure 9 illustrates the contribution of each loss. The leftmost column shows the input airplane image (top) and mesh (bottom). From the second column on, we progressively add more losses. With only the silhouette loss, degenerate solutions occur, severely overfitting to the input viewpoint. The displacement loss helps regularize deformation magnitude; the smoothness losses reduce jagged artifacts; the symmetry losses correct shape details (e.g. airplane tail) by enforcing a symmetry prior. These operate intuitively and can be tweaked for target applications. For example, if only symmetric objects are considered $\lambda_{Sym,B}$ can be increased.

5.3. bbTTSR Results

We next consider the robustness of REFINEd postprocessing to different levels of domain gap. A first set of experiments was performed without domain gap, with reconstruction networks trained and tested on the ShapeNet renders of [5]. Table 3 compares different reconstruction networks and TTSR methods (full per-class results in supplementary). Since the weights used in the state of the art method of [39] are not publicly available, we instead REFINEd OccNet† [34]. The REFINEd+OccNet combination beats the state of the art, despite a somewhat unfair comparison, since REFINEd performs black-box TTSR and is applicable to any network while the MeshSDF refinement of [39] is specific to its network.

Several experiments were next conducted to evaluate the effectiveness of REFINEd in the presence of domain gap. Table 4 gives reconstruction accuracy for RerenderedShapeNet reconstructions, before and after REFINEd, of ShapeNet pretrained networks. Four representatives of different reconstruction strategies are considered: OccNet (implicit functions [34]), Pixel2Mesh (ellipsoid deformation [51]), AtlasNet (surface atlas elements [13]), and Pix2Vox (voxel outputs, converted to mesh [33, 57]). A larger table with per-class results is presented in the supplementary; REFINEd provides gains for all classes. The pre-refinement results of Table 4 are generally worse than those of Table 3. While the methods perform well on the training domain, they struggle to generalize to out-of-distribution data. However, REFINEd significantly recovers much of the lost performance for all networks, for relatively little extra computational overhead. Gains are particularly large for the Chamfer distance (-11.5 for OccNet, -14.9 for Pixel2Mesh, and -29.6 for AtlasNet) and increase with network sensitivity to domain gap (e.g. largest for AtlasNet, which has the weakest performance).

We next considered real-world datasets, which have the largest domain gap and are more interesting for applications. Table 5 shows that on Pix3D, REFINEd gains are qualitatively identical to those of Table 4. A comparison to the unrefined version of MeshSDF are comparable (both implicit based) and have nearly identical performance prior to refinement.

†OccNet and the unrefined version of MeshSDF are comparable (both implicit based) and have nearly identical performance prior to refinement.
the TTSR method of [39] on “Chair” shapes (the only class considered in [39]) again shows that REFINE substantially improves on the state of the art. This occurs even though performance prior to refinement is actually worse for Occ-Net than MeshSDF (Chamfer Distance 110.7 vs 102).

Finally, we studied pose and domain invariance using the 3D-ODDS dataset. For simplicity, we focused on OccNet and the F-score metric (as EMD and CD are unbounded). For each object, we measured accuracy before and after REFINEment using its 24 images as input. Boxplots of example results are shown in Figure 10 (full version in supplementary). Averaged per-object mean accuracy before and after REFINE, over all objects, were 37.2 and 44.4 respectively, while averaged per-object standard deviation were 16.2 and 14.3. This indicates that REFINE improves both reconstruction accuracy and invariance. Figure 11 summarizes averaged performance across pose angle, domain, and object class. REFINE improves reconstruction in all cases.

These results provide insight on the limitations of current reconstruction networks. OWILD (noisiest due to drone camera shake) is the hardest domain on average, followed by OOWL and OTURN (least noisy). Viewpoints at 0 and 180 degrees are most challenging: it is generally more difficult to infer object shape directly from the front or back. Geometrically simple classes like bottles, cans, and bowls perform better than average, with some exceptions like remotes (simple but do not do well). REFINE is beneficial for both classes seen and not seen during training (the latter marked by asterisks). To quantify the relationship between the 3 factors (pose, domain, class) and REFINEd accuracy, we used a 3-way ANOVA [9], with a blocked design to account for object-specific variability. Details are given in the supplementary; all factors were found statistically significant and total variability was decomposed into 13% class, 2% pose, 1% domain, 19% object instance, and 17% from interaction effects between pose/class/domain.

Overall, Tables 3, 4, 5, and Figures 10, 11 show bbTTSR with REFINE achieves state of the art reconstruction accuracies, consistently providing performance gains regardless of metric, original base reconstruction network, class, viewpoint, domain, or dataset. Furthermore, gains are consistent or slightly better as domain gap widens; for the best performing OccNet, utilizing REFINE yields F-Score average improvements of 5, 4, 6, 5, 7, and 6 on ShapeNet, RenderedShapeNet, Pix3D, OTURN, OOWL, and OWILD. These improvements are illustrated in Figure 4. REFINE can both sharpen details (i.e. airplane’s elongated nose) and create entirely new parts (set of wings in the back). It can also recover from very poor reconstructions due to significant domain shift, such as in the table and chair from Pix3D. It especially excels in unusual “outlier” shapes, such as the phone’s antenna or convertible car from 3D-ODDS and is cant domain shift, such as in the table and chair from Pix3D. The table and chair from Pix3D. Figure 11. Performance & standard deviation on 3D-ODDS across viewpoint, domain, & class (asterisked unseen during training). Compared to original scores (orange), REFINEing (green) generally improves accuracy while decreasing variability.

6. Conclusion

In this paper, we demonstrated the effectiveness of black-box test-time shape refinement (bbTTSR) for single view 3D reconstruction. The proposed REFINE method enforces regularized input image consistency, applicable to any reconstruction network in the literature. Experiments show systematic significant improvements over the state of the art, for many metrics, datasets, and reconstruction methods. A new hierarchical multiview, multidomain image dataset with 3D meshes, 3D-ODDS, was also proposed and shown to be a uniquely challenging benchmark for SVR.

Acknowledgments This work was partially funded by NSF award IIS-1924937, award IIS-2041009, and a gift from Amazon. It is supported by the National Science Foundation Graduate Research Fellowship. We also thank the use of the Nautilus platform for some of the experiments discussed above.

<table>
<thead>
<tr>
<th></th>
<th>EMD</th>
<th>CD</th>
<th>F-Score</th>
<th>Viol. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshSDF [39]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chair</td>
<td>11.0±8.5</td>
<td>105.7±463</td>
<td>57±62</td>
<td>20±29</td>
</tr>
<tr>
<td>Bedroom</td>
<td>7.5±6.1</td>
<td>79.1±47.9</td>
<td>57±62</td>
<td>22±23</td>
</tr>
<tr>
<td>Bookcase</td>
<td>7.4±6.1</td>
<td>72.0±38.5</td>
<td>56±65</td>
<td>9±12</td>
</tr>
<tr>
<td>Desk</td>
<td>7.6±6.7</td>
<td>60.6±45.7</td>
<td>71±72</td>
<td>26±27</td>
</tr>
<tr>
<td>Misc</td>
<td>10.2±5.4</td>
<td>129.6±69.8</td>
<td>46±55</td>
<td>19±20</td>
</tr>
<tr>
<td>Sofa</td>
<td>3.2±3.1</td>
<td>30.8±25.5</td>
<td>75±76</td>
<td>50±51</td>
</tr>
<tr>
<td>Table</td>
<td>6.5±5.6</td>
<td>67.7±57.8</td>
<td>60±62</td>
<td>16±17</td>
</tr>
<tr>
<td>Tool</td>
<td>10.8±8.6</td>
<td>140.3±118.6</td>
<td>31±30</td>
<td>11±14</td>
</tr>
<tr>
<td>Wardrobe</td>
<td>5.9±3.5</td>
<td>39.9±29.3</td>
<td>65±68</td>
<td>54±55</td>
</tr>
<tr>
<td>Mean</td>
<td>9.7±7.2</td>
<td>81.7±56.2</td>
<td>57±65</td>
<td>25±28</td>
</tr>
</tbody>
</table>

Table 5. REFINEment gain for large domain shifts, namely Pix3D reconstructions by ShapeNet trained networks. REFINE achieves gains under all metrics and for all networks. REFINE is even able to improve on classes not seen during training (asterisked).
References


