UNIVERSITY OF CALIFORNIA SAN DIEGO

Understanding Learned Visual Invariances Through Hierarchical Dataset Design and Collection

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Electrical Engineering
(Machine Learning & Data Science)

by

Brandon Leung

Committee in charge:

Professor Nuno Vasconcelos, Chair
Professor Xiaolong Wang
Professor Pengtao Xie

2022

Copyright
Brandon Leung, 2022
All rights reserved.

The thesis of Brandon Leung is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my parents and grandparents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First I would like to thank my advisor, Nuno Vasconcelos. He has supported me greatly ever since I joined SVCL as a Sophomore, and taught me how to look at research analytically and critically. He also taught me how to properly convey research findings – accurately and complete, yet with elegance and simplicity. I would also like to thank the other members of my committee, Professors Xiaolong Wang and Pengtao Xie. I am also grateful for Professors Janine Tiefenbruck and Christine Alvarado for their amazing support throughout the years. There are also two other inspirational educators who, through their pedagogy, have profoundly changed how I looked at the world: Charles Beilin and Professor David Quarfoot. I hope to follow in your footsteps when I teach mathematics one day.

I would like to thank my colleagues at SVCL, including Chih-Hui (John) Ho, Pedro Morgado, Bo Liu, Tz-Ying Wu, Yunsheng Li, Yi Li, Pei Wang, Zhaowei Cai, and Jiacheng Cheng. In particular I would like to thank Bo and Pedro for their guidance and mentorship when I was initially beginning research, and John for many insightful discussions and supportive conversations. I consider him a great friend. I also collaborated with many others during my time at UCSD which I am thankful for, including Amir Persekian, David Orozco, Erik Sandström, Yen Chang, Sean Kamano, Igor Vivcharenko, Huang Po Hsian, Arth Dharaskar, and Allen Cheung.

Finally, I would like to thank my family: my younger brother Darron, and my parents Peggy and Jimmy. As a first-generation college student, their support means everything. I would also like to thank my grandparents, who took great care of me as a child. My uncle Wilson, who first made me interested in computer science. And my partner of 6 years, Barbara, who has been source of wonderful inspiration, warmth, and joy in my life.

Chapter 2 is, in full, based on the submission for publication of “Black-Box Test-Time Shape REFINEment for Single View 3D Reconstruction”, Brandon Leung, Chih-Hui Ho, Nuno Vasconcelos, currently in peer review. The thesis author was the primary investigator and author of this material.

Chapter 3 is, in full, based on the publication of “Catastrophic Child’s Play: Easy to Perform, Hard to Defend Adversarial Attacks”, Brandon Leung, Chih-Hui Ho, Erik Sandström, Yen Chang, Nuno Vasconcelos, as it appears in Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. The thesis author was a primary investigator and author of this material, along with Chih-Hui Ho.

I am appreciative of support given to me from the National Science Foundation (NSF) in the form of a GRFP fellowship given to me, and additionally NSF support given to my advisor which partially funded my work. I am also grateful for support from a Alfred P. Sloan Foundation fellowship, a UCSD STARS fellowship, and UC LEADS.

ABSTRACT OF THE THESIS

**Understanding Learned Visual Invariances
Through Hierarchical Dataset Design and Collection**

by

Brandon Leung

Master of Science in Electrical Engineering
(Machine Learning & Data Science)

University of California San Diego, 2022

Professor Nuno Vasconcelos, Chair

While CNNs have enabled tremendous progress in computer vision for a variety of tasks, robust generalization across domain, viewpoint, and class remains a significant challenge. This thesis therefore centers around a new hierarchical multiview, multidomain image dataset with 3D meshes called 3D-ODDS. Data was collected in several ways, involving turntable setups, flying drones, and in-the-wild photos under diverse indoor/outdoor locations. Experiments were subsequently conducted on two important vision tasks: single view 3D reconstruction and image classification. For single view 3D reconstruction, a novel postprocessing step involving test-time self-supervised learning is proposed to help improve reconstructed shape robustness. For image classification, we consider an adversarial attack framework using perturbations which are semantically imperceptible based on human subject surveys. For both tasks, experiments show that 3D-ODDS is a challenging dataset for state of the art methods and is useful in measuring class, pose, and domain invariance. We believe that the dataset will remain relevant moving forward, inspiring future works towards robust and invariant methods in computer vision.

Chapter 1

Introduction

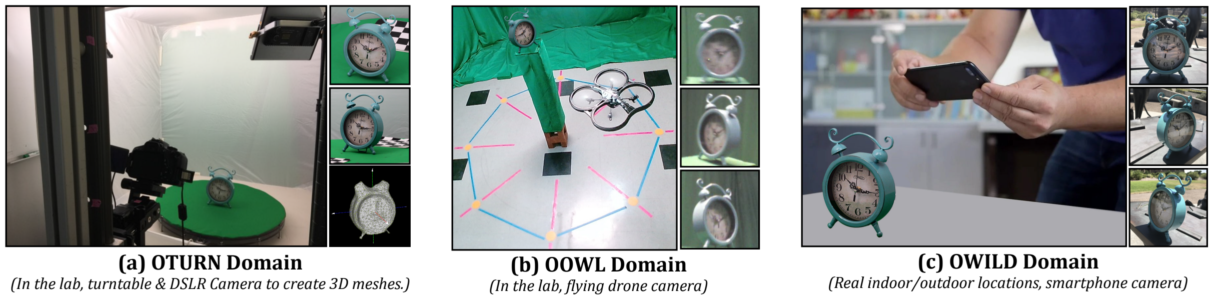

(a) OTURN Domain
(In the lab, turntable & DSLR Camera to create 3D meshes.)
(b) OOWL Domain
(In the lab, flying drone camera)
(c) OWILD Domain
(Real indoor/outdoor locations, smartphone camera)

Figure 1.1: Data collection for 3D-ODDS domains. OTURN uses a high resolution camera turntable setup, for generating 3D meshes with structure-from-motion. OOWL is captured using a drone camera, mid-flight. OWILD depicts objects in diverse indoor/outdoor locations, captured using smartphone cameras.

Over the past decade, convolutional neural networks (CNNs) [47, 82, 28] have led to tremendous progress in computer vision. Significant advances have been made for many tasks (e.g. classification, regression, retrieval, segmentation, detection, 3D reconstruction, and generation [86, 122, 127, 108]), under many machine learning paradigms (e.g. fully supervised, self supervised, unsupervised, and low-shot [38, 107, 67]). This is in large part enabled by recent advances in hardware (especially GPUs [80]) and large scale datasets collected on the internet [12]. However, generalizability and robustness is still a significant challenge. For example, although state of the art methods can now achieve up to 90.2% top-1 and 98.8% top-5 classification accuracy [74] on large scale datasets like ImageNet, they are vulnerable to adversarial attacks [48, 71, 66, 87], indicating a lack of robustness. All datasets are also to some degree subject to particular biases [96, 95]. Thus, methods trained on them can incur "blind spots", making them insufficiently generalized to images encountered in the real world. Alarmingly, these weaknesses can be "hidden" under misleadingly high test set performance – a dataset's test set is insufficient to reveal generalizability weaknesses, since it is ultimately drawn from the same distribution as the training set. In this thesis, it is hypothesized that an external dataset with controlled, hierarchical levels of variability would effectively reveal these generalizability blind spots in state of the art methods. A new dataset satisfying these characteristics is presented, several experimental results are described, and methods are proposed to improve robustness and generalizability.

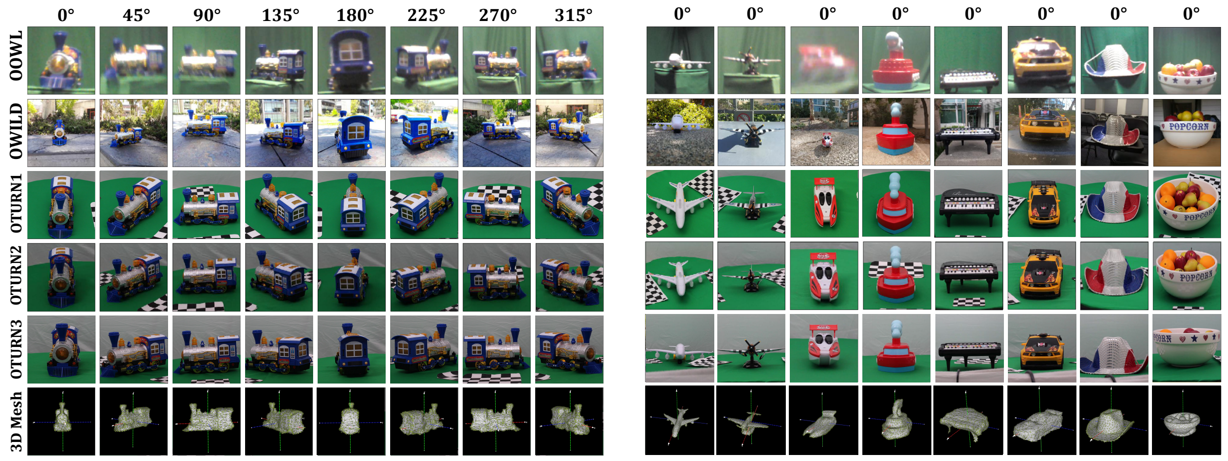


Figure 1.2: The proposed 3D-ODDS dataset incorporates many objects from 16 classes. Along with 3D mesh scans, real-world images are systematically captured from 3 domains and 8 azimuth angles.

1.1 Robustness and Invariance in Computer Vision

In the context of computer vision, a model is considered *robust* if its predictions are correct and consistent (*invariant*) regardless of appearance-based changes in its input. That is, to a reasonable extent (i.e. that of the human visual system [41, 36]), external factors affecting image appearance like camera quality, viewpoint, lighting, color, rotation, background, scale, and spatial translation should not affect the accuracy of the prediction. This has been a fundamental challenge in computer vision, from its origins to present day. For example, the Harris corner detector [25] is a classical method for image feature extraction, whose corner-based features are generally invariant to translation, rotation, and illumination. SIFT [62] features built on top of this by also enabling scale invariance and some local affine invariance, through a pyramidal difference of Gaussians approach. Overall, these classical methods are handcrafted and can generally only enable invariance with respect to low-level, geometrical properties like scale, affine transformation, and rotation.

More recently, CNNs have been incredibly popular and are trained using large datasets, allowing for more abstract and complicated invariances like viewpoint, lighting, and background. Here, mappings to the same output are learnable through exposure to multiple copies of transformed training inputs. Architectural components also enable geometric invariances, i.e. max pooling layers for scale invariance and sliding convolutional filters for translation invariance [84, 43]. Additionally, techniques exist to enhance robustness, such as data augmentation [81]. While CNNs' data-driven approach provides significant advantages, there are also some consequences to contend with. First, CNNs can be vulnerable to adversarial attack exploits [48, 71, 66, 87] which leverage the fact that they are data-driven models whose parameters are only approximately optimized through gradient decent. A subset of these attacks can be defended against [49, 97, 35], but generally require access to the underlying data-generation mechanism for additional training. Second, relying on datasets will inevitably incur certain types of bias. For example, large scale datasets do not have a uniform distribution of viewpoints with which objects are depicted [113], leading to pose bias. Additionally, issues can exist at inference time if images' general appearance do not match with the images seen during training [124, 102], in the form of domain bias. These biases lead to blind spots in trained networks, prohibiting true invariance.

1.2 Contributions of the Thesis

To tackle the aforementioned issues, this thesis centers around a new dataset called the 3D Object Domain Dataset Suite (3D-ODDS). This dataset contains over 100,000 images of real objects collected under controlled viewpoints, along with their scanned 3D meshes. As illustrated in Figures 1.1 and 1.2, each object is depicted in three different domains: OTURN, OOWL, and OWILD. OTURN images were collected with a turntable DSLR camera setup, OOWL using drone cameras during flight, and OWILD with smartphones in diverse indoor/outdoor locations. To show that 3D-ODDS can effectively diagnose for class, pose, and domain invariance, we use it to test two key vision tasks: 3D reconstruction and image classification. Through numerous experiments, it is shown that 3D-ODDS is an extremely challenging dataset for state of the art methods. To address this, novel approaches are proposed to help improve robustness and generalizability. We believe that the 3D-ODDS dataset will remain relevant moving forward, inspiring future works towards domain generalizable [124, 102] and invariant methods in computer vision.

1.2.1 Robustness for Single View 3D Reconstruction

Chapter 2 introduces the 3D-ODDS dataset, and explores robustness in the context of single view 3D reconstruction. In this task, the goal is to reconstruct the 3D shape of an object from an image of it. Besides being a fundamental research question for artificial intelligence (humans can easily envision 3D structure at a glance [79]), it is also important in areas such as autonomous vehicle 3D environment mapping [121], robotic object manipulation/grasping [51], and virtual/augmented reality interaction [119]. Much recent progress has been made, but due to the difficulty of collecting large datasets in the wild with 3D ground truth, it still 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 postprocessing 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.

1.2.2 Robustness for Classification

Chapter 3 explores robustness in the context of image classification, from an adversarial attack [48, 71, 66, 87] point of view. In particular, the 3D-ODDS dataset enables us to consider attacks that are trivial to perform but difficult to defend. A framework for the study of such attacks is proposed, using real world object manipulations. Unlike most works in the past, this framework supports the design of attacks based on both small and large image perturbations, implemented by drone camera shake and pose variation from OOWL. A setup is proposed for the collection of such perturbations and determination of their perceptibility. It is argued that the latter depends on context, and a distinction is made between imperceptible and semantically imperceptible perturbations. While the former survive image comparisons, the latter are perceptible but have no impact on human object recognition. A procedure is proposed to determine the perceptibility of perturbations using Amazon Mechanical Turk [69] experiments, and a dataset of both perturbation classes which enables replicable studies of object manipulation attacks, is assembled. Experiments using defenses based on many datasets, CNN models, and algorithms from the literature elucidate the difficulty of defending these attacks – in fact, none of the existing defenses is found effective against them. Better results are achieved with real world data augmentation, but even this is not foolproof. These results confirm the hypothesis that current CNNs are vulnerable to attacks implementable even by a child, and that such attacks may prove difficult to defend.

Chapter 2

Improved 3D Reconstruction Robustness via Test-Time Shape REFINEment

2.1 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 [63, 104, 117], or only applicable to a specific object class per network, such as birds [40, 53], 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 2.1 and 2.4, the application of an SVR network trained on ShapeNet [7] 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 [93], and make slight adjustments that are usually not enough to recover intricate shape details. As shown in Figure 2.2, 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) [76] 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 reconstruction across many object classes and different image domains, while providing a dense coverage of object views. In this work, we leverage this observation and propose a new hierarchical multiview, multiclass, multidomain dataset called the *3D Object Domain Dataset Suite (3D-ODDS)*, containing 71,496 real images of objects collected

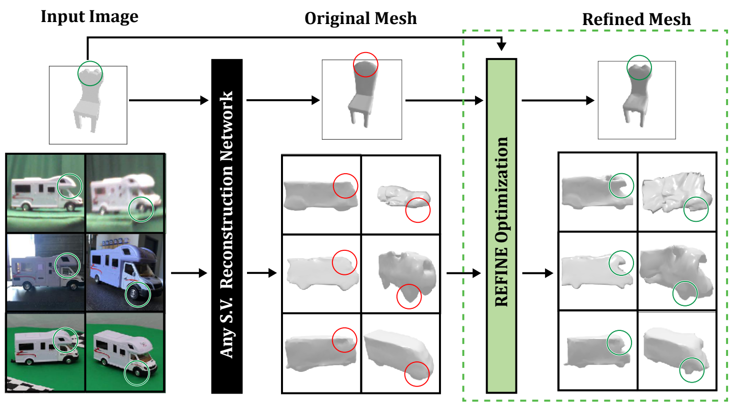


Figure 2.1: **bbTTSR:** Reconstructions by a SVR network trained on ShapeNet are fed to the proposed *external* shape REFINEment network at test-time. **Evaluation:** Images from the training domain (top, from ShapeNet) are combined with a *new* dataset, to evaluate how bbTTSR enhances accuracy and robustness.

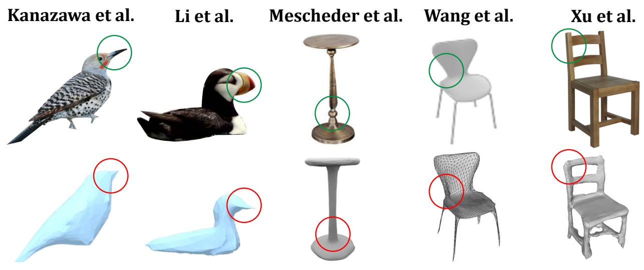


Figure 2.2: Important image details (circled in green) are frequently lost by state of the art 3D reconstruction methods (circled in red).

under many different controlled poses and domains, along with their scanned 3D meshes (Figures 2.1, 2.8). 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 [76] 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 2.1, 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 [73] or OccNet [63] which use implicit functions, Mesh R-CNN [20] or Pix2Vox [117] which have voxel-based components, and AtlasNet [23] 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 *REFine 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 [7, 10, 88] 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 the concept of bbTTSR, i.e. the use of external post-processing networks 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, and enhances the performance of many SVR networks under many experimental conditions.

2.2 Related Work

Single View 3D Reconstruction. While many SVR methods have been proposed, they all suffer from the inconsistencies of Figure 2.2, 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 [117, 10]. Octrees can enable higher voxel resolution [92, 105]. Pointclouds have been explored as an alternative to voxels [16, 54] but usually require voxel or mesh conversion for use by downstream tasks. Among mesh methods, some learn to displace vertices on a sphere [42, 104] or a mean shape [40] to reconstruct. Current state of the art methods rely on an intermediate implicit function representation to describe shape [63, 118, 73, 19, 68], mapped into a mesh by marching cubes [61].

Methods also vary by their level of supervision. Most are fully supervised, requiring a large dataset of 3D shapes such as ShapeNet [7]. Recently, weakly-supervised methods have also been introduced, using semantic keypoints [40] or 2.5D sketches [109] as supervision. Alternatively, [53] has proposed a fully unsupervised method, combining part segmentation and differentiable rendering. Few-shot is considered in [100, 64] where classes have limited training data. Domain adaptation was explored in [75], 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, [93] 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 2.1 and 2.2, 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 [93]. It also makes REFINE complementary to reconstruction methods and applicable as a postprocessing stage to any of them.

Test-Time Optimization. Test-time training [89] or optimization usually exploits inherent structure of the data in a self-supervised manner, as no ground truth labels are available. In [89], an auxiliary self-supervised rotation angle prediction task is leveraged to reduce domain shift in object classification. The same goal is achieved in [101] by test-time entropy minimization. Meanwhile, [98] uses self-supervision at test time to improve human motion capture. Additionally, interactive user feedback serves to dynamically optimize segmentation predictions [78, 85, 37].

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 [52, 55]. [52] requires the unsupervised part-based video reconstruction architecture proposed by the authors and [128] optimizes over a shape space specific to their architecture for zebra images. Among white-box methods, the approach closest to REFINE is that of [76], 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 [73]) 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 [76] for DeepSDF networks. In

Table 2.1: REFINE improves reconstruction by introducing only a small number of parameters, relative to popular networks.

	REFINE	OccNet [63]	MeshSDF [73]	Pix2Mesh [104]	AtlasNet [23]
Params. (M)	0.9	12.7	13.2	18.8	20.3

summary, unlike prior approaches, REFINE is a black-box technique that can be universally applied to improve reconstruction accuracy, a posteriori.

2.3 Black-Box Test-Time Shape Refinement

2.3.1 Formulation and Inputs

Single view 3D reconstruction methods reconstruct a 3D object shape from a single image of the object. This is implemented with a mapping

$$S : \mathbb{R}^{W \times H \times 3} \to \mathcal{M} \in \mathcal{V} \times \mathcal{E}, \tag{2.1}$$

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 $V \in \mathcal{V} \subset \mathbb{R}^{N \times 3}$ is a set of vertices and $E \in \mathcal{E} \subset \mathbb{B}^{\binom{N}{2}}$ a set of edges. $\mathbb{B}$ is a boolean domain specifying mesh connectivity. $S(x)$ is usually a coarse shape estimate, whose details do not match the input image, as shown in Figure 2.2. Performance further degrades when x is sampled from an image distribution different from that used during training [75].

In [76], 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 2.1. We denote the approach as REFINE and R 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 2.3.3.

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 pointcloud reconstruction methods, by using mesh conversions [45, 44, 5, 3]. This property is important, as different methods are better suited for different downstream applications. For example, implicit methods [63, 73] tend to produce the best reconstructions but can have slow inference [73]. Meanwhile, AtlasNet [23] 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 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 2.1, the REFINE network is at least 10 times smaller than most currently popular reconstruction networks.

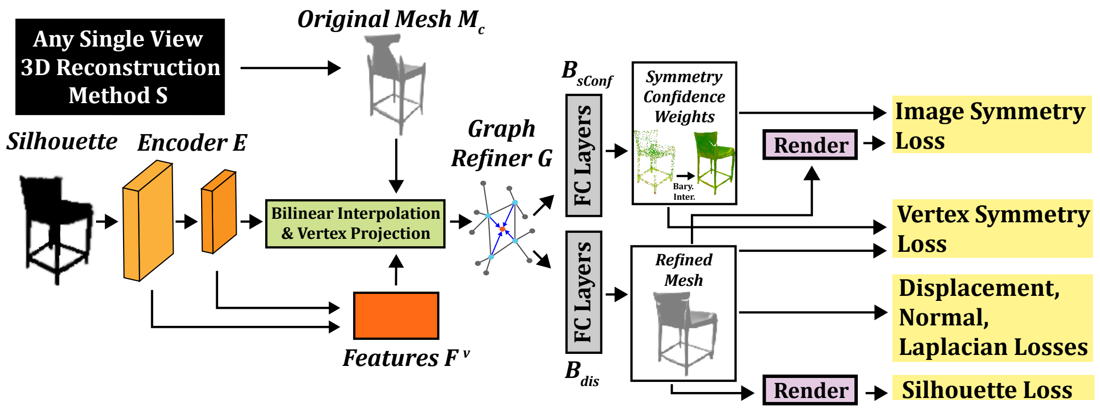


Figure 2.3: A feature map encoder, graph convolutions, and fully connected layers are used to output vertex REFINEments needed to make the mesh consistent with an input image. Losses, including confidence based 3D symmetry, prevent degenerate solutions. Optimization over single examples, at test time.

2.3.2 Architecture

Figure 2.3 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 x_s as input, and outputs a set of L feature maps $F(x_s;\Theta_l = \{\theta_j\}_{j=1}^{l}) \in \mathcal{R}^{W_l \times H_l \times C_l}$, of width W_l, height H_l and C_l channels. In our implementation, E is based on ResNet [28]; L is set to 2, where $C_1 = 64$ and $C_2 = 128$.

Given feature map $F(x_s;\Theta_l)$, the feature vector f_l^v corresponding to a vertex v in M_c is computed by projecting the vertex position onto the feature map [20, 104],

$$f_l^v = Proj(v; F(x_s;\Theta_l), p) \in \mathcal{R}^{C_l}, \tag{2.2}$$

where p is the camera viewpoint and $Proj$ a perspective projection with bilinear interpolation. Vertices are represented at different resolutions, by concatenating the feature vectors of different layers into $F^v = (f_1^v, \dots, f_L^v)^T$. The set $\{F^v\}_{v=1}^{N}$ of concatenated feature vectors extracted from all vertices is then processed by a graph convolution [46] refiner G, of parameters ϕ, to produce an improved set of feature vectors $\{H^v\}_{v=1}^N = G(\{F^v\}_{v=1}^N; \phi)$. Finally, this set is mapped into the displacement vector V_{dis}

$$V_{dis} = B_{dis}(\{H^v\}_{v=1}^N; \psi_{dis}), \tag{2.3}$$

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

$$V_{sConf} = B_{sConf}(\{H^v\}_{v=1}^N; \phi); \psi_{sConf}) \tag{2.4}$$

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

$$R(x_s, M_c; \{\theta_i\}, \phi, \psi_{dis}, \psi_{sConf}, p) = \{V_{dis}, V_{sConf}\}. \tag{2.5}$$

2.3.3 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 [42, 56] that maps mesh $M \in \mathcal{M}$ into its image captured by a camera of parameters p. We also define sets V_r^s, V_{dis}^s, and V_{sConf}^s of size N, constructed with the rows of V_r, 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

$$L_{Sil} = L_{BCE}(x_s, \gamma(M_r^p)), \tag{2.6}$$

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

$$L_{BCE}(a,b) = \sum_{ij} a_{ij} \log(b_{ij}) + (1 - a_{ij}) \log(1 - b_{ij}). \tag{2.7}$$

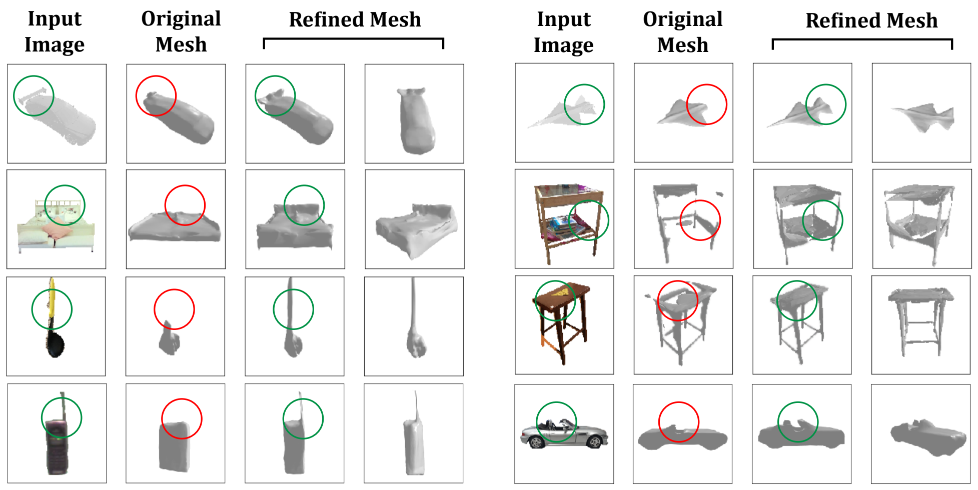


Figure 2.4: REFINEd reconstructions from an OccNet trained on ShapeNet. Results for objects from ShapeNet (top row), Pix3D (middle rows), & 3D-ODDS (bottom row). REFINE can correct small details as well as generate entirely new parts.

Displacement Loss: Discourages overly large vertex deformations, with

$$L_{Dis} = \sum_{v_i \in V^s_{dis}} ||v_i||_2^2. \tag{2.8}$$

Normal Consistency & Laplacian Losses: L_{Nc} and L_{Lp} are widely used [104, 13] and encourages mesh smoothness.

A second set of losses is introduced to avoid degenerate solutions, namely overfitting to the input view during bbTTSR. 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 [58]. Many methods (e.g. [63, 104]) learn symmetry implicitly from the training data. Since datasets like Shapenet [7] are composed primarily of symmetric objects, a learned bias towards symmetry is almost impossible to avoid. Symmetry can also be explicit, e.g. [125] predicts planes of symmetry given 2D images to improve monocular depth estimation, or used to regularize learning, e.g. with horizontal flips during training [111].

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. These leverage horizontal flips, vertex symmetry, and camera symmetry. Reflections are implemented with transformation $T = I - 2\vec{n}\vec{n}^{\intercal}$, where $\vec{n} \in \mathbb{R}^3$ is the unit normal vector of the reflection plane $\mathcal{Z}$. While there are methods to predict planes of object symmetry [18, 125] we have found that most reconstruction networks output aligned meshes, where $\vec{n} = [0,0,1]^{\intercal}$. To prevent the symmetry prior from overwhelming (2.6) if some asymmetry is present, confidence scores σ_i are learned during the REFINE optimization per vertex v_i. This enables local deviations from 3D symmetry when appropriate. The symmetry losses are as follows.

Vertex-Based Symmetry Loss: Encourages symmetric mesh vertices according to

$$L_{Vsym} = \frac{1}{N}\sum_{i=1}^{N} \sigma_i \min_{v_j \in V_r^s} \left\|Tv_i - v_j\right\|_2^2 + \lambda_{SymB} \ln\left(\frac{1}{\sigma_i}\right), \tag{2.9}$$

where $v_i \in V_r^s$ are the mesh vertices and $\sigma_i \in V_{sConf}^s$ the associated symmetry confidence scores. The first term penalizes distances between each vertex and its nearest neighbor upon reflection about $\mathcal{Z}$. This is weighted by the confidence score σ_i, which is low for vertices that should be asymmetric based on the object silhouette. The second term penalizes small confidence scores, preventing trivial solutions. The trade-off between the two terms is controlled by hyperparameter $\lambda_{SymB} \in [0,\infty)$. As shown in Figure 2.6, scores σ_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 $P_{Isym} = \{p_1, ..., p_m\}$, T is used to obtain differentiably rendered pairs from symmetric camera viewpoints $\{(M_r^{p_1}, M_r^{Tp_1}), ..., (M_r^{p_m}, M_r^{Tp_m})\}$, as shown in the rows of Figure 2.5. The loss is defined as

$$L_{Isym} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j,k}\left[\sigma_{j,k}||\gamma(h(M_r^{p_i}))_{j,k} - \gamma(M_r^{Tp_i})_{j,k}||_2^2 + \lambda_{SymB}\ln\left(\frac{1}{\sigma_{j,k}}\right)\right], \tag{2.10}$$

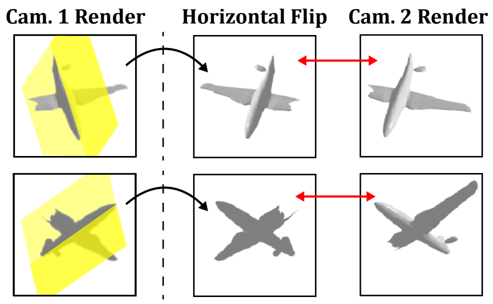


Figure 2.5: To enforce symmetry, a mesh is differentiably rendered by two cameras; the viewpoint of camera 2 is obtained by reflecting that of camera 1 about the mesh's plane of symmetry (yellow). The second render is compared to the horizontal flip of the first.

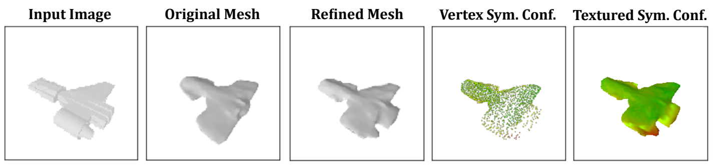


Figure 2.6: Left to right: image, original mesh, REFINEd mesh, and vertex confidence weights (shown as points or colors on the REFINEd mesh). Green shades indicate higher confidence; red lower, relaxing the symmetry prior on asymmetric object parts.

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 $M_r^{p_i}$ and the render $M_r^{Tp_i}$ 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 (2.9). However, they are not relearned, but derived from the vertex confidences $\sigma_i, i \in \mathcal{V}_{j,k}$, of (2.9) by barycentric interpolation on the mesh faces, where $\mathcal{V}_{j,k}$ are mesh face vertices projected into pixel j,k.

Overall Loss: REFINE is trained with a weighed combination of the six losses

$$L_{total} = \lambda_{Sil}L_{Sil} + \lambda_{Isym}L_{Isym} + \lambda_{Vsym}L_{Vsym} + \lambda_{Dis}L_{Dis} + \lambda_{Nc}L_{Nc} + \lambda_{Lp}L_{Lp}. \tag{2.11}$$

L_{Sil} is the main driving factor to ensure input silhouette consistency, while other losses serve as regularizers to prevent degenerate solutions. Figure 2.7 shows that REFINEd shape quality

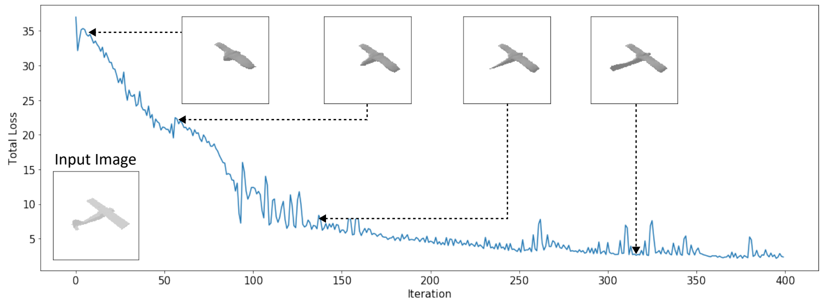


Figure 2.7: Mesh shape improves as REFINEment optimizes.

tracks the evolution of this loss, for an airplane whose body has been truncated in the original reconstruction. As the REFINE loss steadily decreases, the mesh progressively becomes more faithful to the input image; this is seen in the elongated body and corrected wing shape.

2.3.4 Implementation Details

Several details of our implementation are worth noting. In all experiments we used P_{Isym} of 6 viewpoints, with azimuths in $\{15°, 45°, 75°\}$ and elevations in $\{-45°, 45°\}$. The learning rate is 0.00007, $\lambda_{Sil} = 10$, $\lambda_{Isym} = 80$, $\lambda_{Vsym} = 20$, $\lambda_{SymB} = 0.0005$ $\lambda_{Dis} = 100$, $\lambda_{Nc} = 10$, and $\lambda_{Lp} = 10$. Also, REFINE 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 Appendix.

2.4 Multiview, Multidomain 2D & 3D Dataset

SVR is usually benchmarked on synthetic CAD datasets [7, 112] because these, albeit unrealistic, allow renders of images from many viewpoints. While real data can also be collected [50, 31, 9, 83], this has various difficulties resulting in datasets with different limitations. For example, Pascal3D [114] contains diverse real indoor/outdoor settings, but meshes are only approximations manually chosen from a CAD library. Pix3D [88] 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 reconstruction across *both* controlled viewpoints and domains.

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 Appendix). Crucially, 232 of the objects can also be found in two real-world, multiview image datasets: OOWL [29] and OWILD [30]. 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 2.8, 3D-ODDS combines OTURN (our collected turntable images and 3D meshes) with OOWL and OWILD images to create a uniquely challenging hierarchical dataset of real images with 3 disentangled factors of variation: pose, object class, and domain. This results in the first real-world dataset to provide both 3D meshes of objects and their images under controlled viewpoints and domains. We believe that 3D-ODDS (to be released publicly) will be an important testing ground to evaluate the real world robustness of SVR methods.

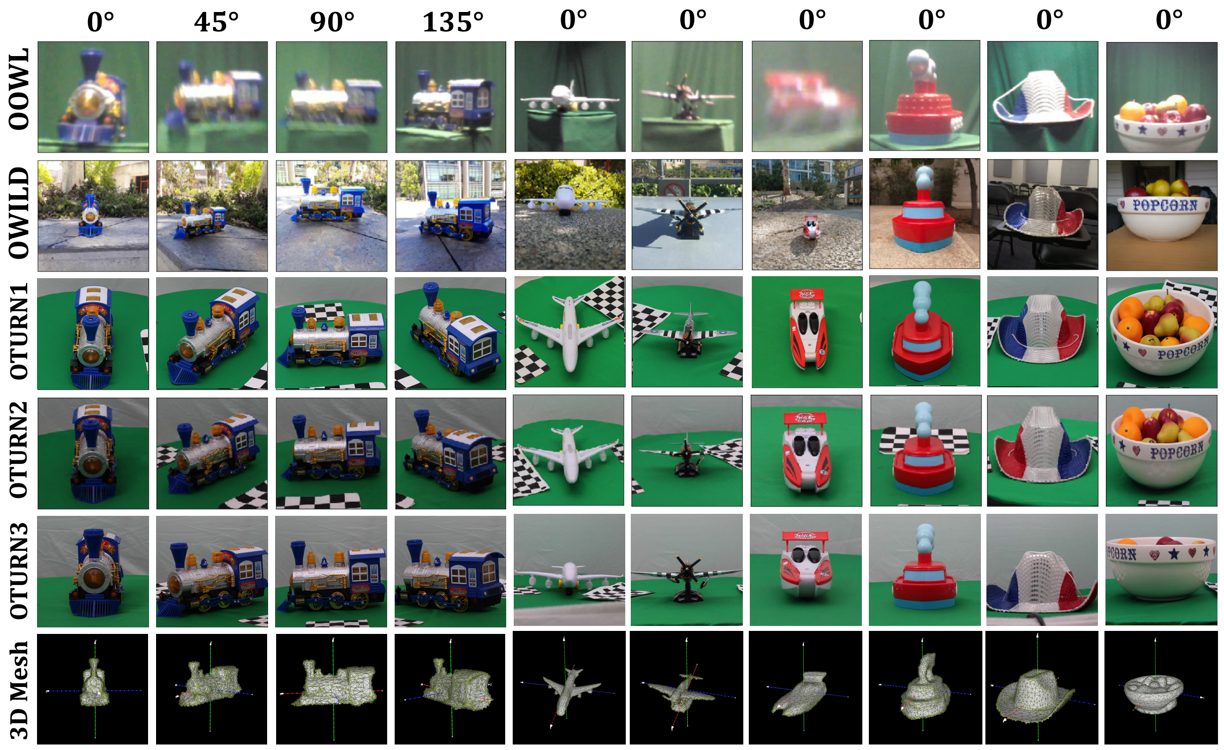


Figure 2.8: The proposed 3D-ODDS dataset contains 3D meshes and images from 3 domains, 8 azimuth angles, and 16 classes.

2.5 Experiments

2.5.1 Experimental Setup

Metrics: To evaluate bbTTSR performance, the original mesh is first reconstructed by a baseline SVR method, and the reconstruction accuracy is measured. REFINE is then applied to the mesh and its accuracy is measured again. Several metrics of 3D accuracy [93] are used: EMD, $l2$ Chamfer Distance, F-Score, and 3D Volumetric IoU. Lower is better for EMD and Chamfer, while higher is better for IoU and F-Score; for details please refer to the Appendix.

Datasets: Five datasets are considered, to carefully study domain shift. All baseline models are trained on the synthetic *ShapeNet* dataset [7], with images rendered by [10] using Blender [11]. We also re-rendered the 3D models in the test set of [10] using Pytorch3D [39]. This second dataset, called *RerenderedShapeNet* is designed to create a domain gap due to significant differences in shading, viewpoint, and lighting. The third dataset is motivated by our observation that about 97% of ShapeNet is symmetrical, in the sense that each mesh has a symmetry loss $L_{Isym} < 0.01$ for $\lambda_{SymB} = 1$ and $\sigma_{j,k} = 1$. To ablate how asymmetry affects reconstruction quality, we introduce a subset of RerenderedShapeNet, denoted as *ShapeNetAsym*, containing 1259 asymmetric meshes. Fourth, we use the *Pix3D* dataset [88], which contains real images and their ground truth meshes, to test for large domain shifts. Finally, we use *3D-ODDS* to study invariance to pose and image domain. For bbTTSR experiments, we use a subset of 3D-ODDS consisting of objects with high quality 3D mesh scans and images of 45° increment azimuth angles found in OOWL, OWILD, and OTURN's middle elevation. In total, this subset consists of 212 objects, 3 domains, and 8 viewpoints, for a total of $212 * 3 * 8 = 5088$ images and 212 meshes.

Table 2.2: Ablation study of REFINE. L_{total} indicates that all losses are used; an asterisk indicates results averaged over ShapeNetAsym instead of RerenderedShapeNet.

Configuration	EMD↓	CD-l_2 ↓	F-Score↑	Vol. IoU↑	2D IoU
OccNet [63]	4.3	34.0	80	33	69
L_{Sil}	12.2	154.8	51	16	87
$L_{Sil,Dis,Nc,Lp}$	3.7	26.2	80	31	85
$L_{Sil,Dis,Nc,Lp,Vsym}$	3.7	25.8	81	32	86
$\boldsymbol{L_{total}}$	**3.3**	**22.5**	**84**	**35**	85
E & G removed, L_{total}	3.5	24.5	82	33	87
E removed, L_{total}	3.4	24.1	82	34	87
E rand. init, L_{total}	3.4	23.1	83	35	85
OccNet* [63]	11.0	123.3	48	10	53
L_{total}, $\lambda_{SymB} = 1.0$*	8.9	89.1	52	10	72
$\boldsymbol{L_{total}}$*	**7.8**	**85.9**	**55**	**12**	76

2.5.2 Ablation Studies

Ablations were performed for different components of REFINE. Here we also measure consistency between the reconstructed mesh's render and the input image silhouette, using 2D IoU. While not necessarily indicative of 3D accuracy, we use it in the following discussion to better understand REFINE's behavior.

The top section of Table 2.2 shows the effect of different REFINEments of Rerendered-ShapeNet meshes originally reconstructed by OccNet. The first row is not refined. The second row shows that, using the silhouette loss only ($\lambda_{Sil} = 10$, all other $\lambda = 0$) improves input image consistency (from 69 to 87 2D IoU), but the refined mesh severely overfits to the input viewpoint, leading to decreased 3D accuracy. Adding the popular regularizers (third row, $\lambda_{Dis} = 100$, $\lambda_{Nc} = 10$, $\lambda_{Lp} = 10$) improves 3D reconstruction, but the gains over the baseline are small. The fourth row shows that enforcing vertex symmetry ($\lambda_{Vsym} = 20, \lambda_{SymB} = 0.0005$) has marginal improvements by itself. However, when combined with render-based image symmetry (row five, which further adds the image symmetry loss with $\lambda_{Isym} = 80$) it enables significant gains in all metrics (e.g. from 34 to 22.5 CD-l_2).

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

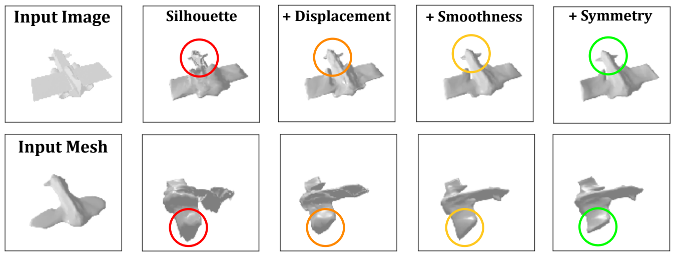


Figure 2.9: Leftmost column: input image and mesh. Other columns: REFINEment improves with an increasingly larger set of losses (left to right).

Table 2.3: 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* $\rightarrow$ *after* refinement, with gain shown in parenthesis.

		EMD $\downarrow$	CD-l_2 $\downarrow$	F-Score $\uparrow$	Vol. IoU $\uparrow$
SVR	AtlasNet [23]	8.0	13.0	89	30
	Mesh R-CNN [20]	4.2	10.3	90	52
	Pix2Mesh [104]	3.4	8.0	93	48
	DISN [118]	2.6	9.7	91	57
TTSR	MeshSDF [76]	3.0→2.5 (-0.5)	12.0→7.8 (-4.2)	91→95 (+4)	-
	REFINEd OccNet [63]	2.9→**2.3** (-0.6)	12.2→**7.5** (-4.7)	91→**96** (+5)	57→**59** (+2)

no network), removing only E, and randomly initializing E. These refinements improve on the original mesh, but underperform the implementation of REFINE 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 [99, 24].

The bottom three rows of Table 2.2 use ShapeNetAsym to study the effect of asymmetry on REFINE performance. The sixth row is not refined. The seventh row shows that when the confidence scores of (2.9) and (2.10) are removed (by setting $\lambda_{SymB} = 1$, 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, $\lambda_{SymB} = 0.0005$). It can also be seen that, when confidence scores are used, the reconstruction quality is significantly superior to

Table 2.4: REFINEment in the presence of mild domain shift, namely RerenderedShapeNet reconstructions by ShapeNet trained networks. Gains occur under all networks, classes, and metrics.

	EMD $\downarrow$	CD-l_2 $\downarrow$	F-Score $\uparrow$	Vol. IoU $\uparrow$
REFINEd OccNet [63]	4.3 $\rightarrow$ **3.3** (**-1.0**)	34.0 $\rightarrow$ **22.5** (**-11.5**)	80 $\rightarrow$ **84** (**+4**)	33 $\rightarrow$ **35** (**+2**)
REFINEd Pix2Mesh [104]	4.8 $\rightarrow$ **3.5** (**-1.3**)	38.0 $\rightarrow$ **23.1** (**-14.9**)	67 $\rightarrow$ **78** (**+11**)	22 $\rightarrow$ **27** (**+5**)
REFINEd AtlasNet [23]	6.2 $\rightarrow$ **4.9** (**-1.3**)	62.5 $\rightarrow$ **32.9** (**-29.6**)	56 $\rightarrow$ **72** (**+16**)	8 $\rightarrow$ **13** (**+5**)
REFINEd Pix2Vox [117]	4.5 $\rightarrow$ **3.3** (**-1.2**)	37.3 $\rightarrow$ **21.8** (**-15.5**)	70 $\rightarrow$ **80** (**+10**)	27 $\rightarrow$ **34** (**+7**)

that of the original meshes. In summary, the proposed confidence mechanism enables effective REFINEment of non-symmetric objects.

Figure 2.9 illustrates the contribution of each loss. The leftmost column shows the input airplane image (top) and mesh (bottom). From the second column, 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 λ_{SymB} can be increased.

2.5.3 bbTTSR Results

We next consider the robustness of REFINE 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 [10]. Table 2.3 compares different reconstruction networks and TTSR methods (full per-class results in Appendix). Since the weights used in the state of the art method of [76] are not publicly available, we instead REFINEd OccNet[1] [63]. The

[1]OccNet and the unrefined version of MeshSDF are comparable (both implicit based) and have nearly identical performance prior to refinement.

REFINE+OccNet combination beats the state of the art, despite a somewhat unfair comparison, since REFINE performs black-box TTSR and is applicable to any network while the MeshSDF refinement of [76] is specific to its network.

Several experiments were next conducted to evaluate the effectiveness of REFINEment in the presence of domain gap. Table 2.7 gives reconstruction accuracy for RerenderedShapeNet reconstructions, before and after REFINEment, of ShapeNet pretrained networks. Four representatives of different reconstruction strategies are considered: OccNet (implicit functions [63]), Pixel2Mesh (ellipsoid deformation [104]), AtlasNet (surface atlas elements [23]), and Pix2Vox (voxel outputs, converted to mesh [117, 61]). A larger table with per-class results is presented in the Appendix; REFINE provides gains for all classes. The pre-refinement results of Table 2.7 are generally worse than those of Table 2.3. While the methods perform well on the training domain, they struggle to generalize to out-of-distribution data. However, REFINEment 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 2.5 shows that on Pix3D, REFINE gains are qualitatively identical to those of Table 2.7. A comparison to the TTSR method of [76] on “Chair” shapes (the only class considered in [76]) again shows that REFINE substantially improves on the state of the art. This occurs even though performance prior to refinement is actually worse for OccNet 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 2.10 (larger version in Appendix). Averaged

Table 2.5: 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).

		EMD $\downarrow$	CD-l_2 $\downarrow$	F-Score $\uparrow$	Vol. IoU $\uparrow$
MeshSDF [76]	Chair	11.9 $\rightarrow$ 9.8 (-2.1)	102.0 $\rightarrow$ 89.0 (-13.0)	-	-
REFINEd OccNet [63]	Chair	11.0 $\rightarrow$ **8.5**	110.7 $\rightarrow$ **74.5**	57 $\rightarrow$ **62**	18 $\rightarrow$ **20**
	Bed*	7.5 $\rightarrow$ **6.1**	70.1 $\rightarrow$ **47.9**	57 $\rightarrow$ **62**	22 $\rightarrow$ **23**
	Bookcase*	7.4 $\rightarrow$ **4.1**	72.0 $\rightarrow$ **38.5**	56 $\rightarrow$ **65**	9 $\rightarrow$ **12**
	Desk	7.6 $\rightarrow$ **6.7**	60.6 $\rightarrow$ **43.7**	71 $\rightarrow$ **72**	26 $\rightarrow$ **27**
	Misc*	10.2 $\rightarrow$ **5.4**	129.6 $\rightarrow$ **69.8**	46 $\rightarrow$ **55**	19 $\rightarrow$ **20**
	Sofa	3.2 $\rightarrow$ **3.1**	30.8 $\rightarrow$ **25.5**	75 $\rightarrow$ **76**	50 $\rightarrow$ **51**
	Table	6.5 $\rightarrow$ **5.6**	67.7 $\rightarrow$ **57.8**	60 $\rightarrow$ **62**	16 $\rightarrow$ **17**
	Tool*	10.8 $\rightarrow$ **8.6**	140.8 $\rightarrow$ **118.6**	51 $\rightarrow$ **60**	11 $\rightarrow$ **14**
	Wardrobe*	5.9 $\rightarrow$ **3.7**	49.9 $\rightarrow$ **29.3**	65 $\rightarrow$ **68**	54 $\rightarrow$ **55**
	Mean	7.9 $\rightarrow$ **5.8** (**-2.1**)	81.4 $\rightarrow$ **56.2** (**-25.2**)	59 $\rightarrow$ **65** (**+6**)	23 $\rightarrow$ **28** (**+5**)

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 2.11 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. OOWL (noisiest due to drone camera shake) is the hardest domain on average, followed by OWILD 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[17], with a blocked design to account for object-specific variability. Details are given in the Appendix; all factors were found statistically significant and total variability was decomposed into 13% class, 2% pose, 1% domain,

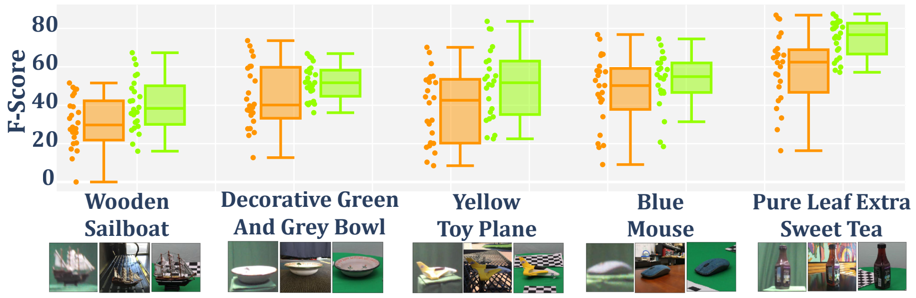


Figure 2.10: 3D-ODDS objects have 24 images (3 domains, 8 viewpoints). Reconstruction accuracies plotted before (after) REFINE as orange (green). REFINE improves performance & invariance.

19% object instance, and 17% from interaction effects between pose/class/domain.

Overall, Tables 2.3 2.7, 2.5, and Figures 2.10, 2.11 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, RerenderedShapeNet, Pix3D, OTURN, OWILD, and OOWL. These improvements are illustrated in Figure 2.4. 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 successful even for *classes on which the original reconstruction method was not trained*, leading to poor original meshes. This includes the spoon and bed in Figure 2.4; unseen classes marked by an asterisk in Table 2.5 and Figure 2.11. Additional examples provided in the Appendix.

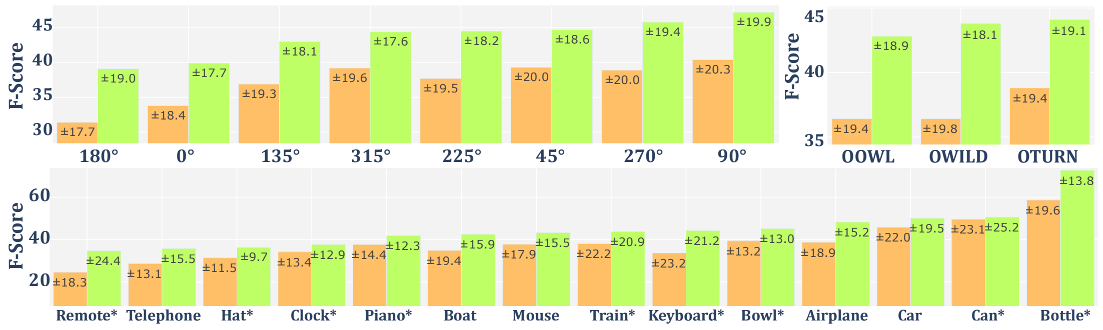


Figure 2.11: Performance & standard deviation on 3D-ODDS across viewpoint, domain, & class (asterisked unseen during training). Compared to original scores (orange), REFINEing (green) generally improves accuracy while decreasing variability.

2.6 Discussion

In this paper, we demonstrated the effectiveness and versatility of black-box test-time shape refinement (bbTTSR) for single view 3D reconstruction. The proposed REFINE method enforces regularized input image consistency, and can be applied 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. We believe that the bbTTSR paradigm and the 3D-ODDS dataset will remain relevant as future reconstruction networks are introduced, inspiring further work towards robust and accurate reconstruction methods.

2.7 Acknowledgements

Chapter 2 is, in full, based on the submission for publication of “Black-Box Test-Time Shape REFINEment for Single View 3D Reconstruction”, Brandon Leung, Chih-Hui Ho, Nuno Vasconcelos, currently in peer review. The thesis author was the primary investigator and author of this material.

2.8 Appendix

2.8.1 Additional Notes on REFINE

2.8.1.1 Extended Figures and Tables

Several figures and tables are given in this Appendix to complement the main paper. Figure 2.12 illustrates that although OccNet [63], Pix2Mesh [104], and AtlasNet [23] produce very different failure cases and artifacts, REFINE improves the both the input image consistency and 3D accuracy of all methods. For detailed per-class results on ShapeNet, please refer to Table 2.6. For per-class results on RerenderedShapeNet, refer to Tables 2.7 and 2.8. Figures 2.21 and 2.22 provides performance measurements visualized as a heatmap for 3D-ODDS across the class/angle and domain/angle factors. Figure 2.30 plots reconstruction accuracies on a per-object basis, for all objects in 3D-ODDS.

Figures 2.23 and 2.24 show more REFINE examples on real-world images (Pix3D and 3D-ODDS). Figure 2.25 is on RerenderedShapeNet, while Figure 2.26 is on ShapeNet. All these figures use an OccNet to reconstruct the original mesh. For REFINEment examples using the AtlasNet, Pix2Mesh, and Pix2Vox reconstruction methods, please refer to Figures 2.27, 2.28, and 2.29 respectively.

2.8.1.2 Scope, Limitations, and Future Work

Test-time shape refinement explores whether or not reconstructions can be improved by the use of additional auxiliary test-time information. In the work of [76], this was performed by optimizing the parameters of a SVR network given a coarsely reconstructed mesh, object silhouette, and pose. We also follow this input setting, which allows us to focus on studying REFINE independently without confounding factors. Research in automatic image segmentation [77, 26, 126, 2] and pose estimation [115, 94, 33] is beyond the scope of this paper, and advancement in those tasks is left for future research. Additionally, we believe that the REFINE

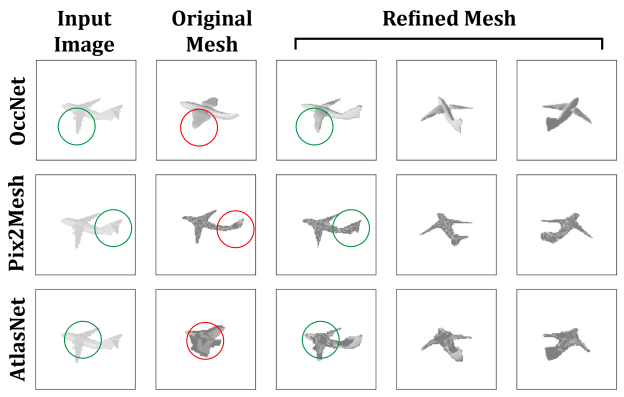


Figure 2.12: An airplane reconstructed by three different methods [63, 104, 23]. Since the methods differ greatly, they exhibit very different failure cases and artifacts. Nevertheless, REFINE improves all reconstructions.

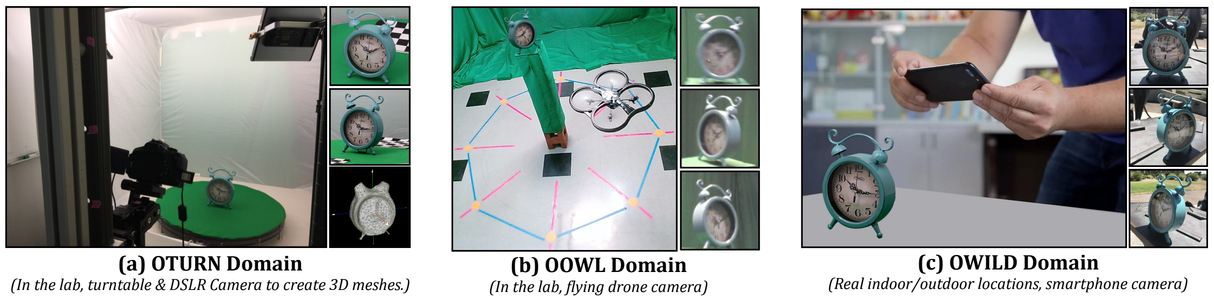

(a) OTURN Domain
(In the lab, turntable & DSLR Camera to create 3D meshes.)
(b) OOWL Domain
(In the lab, flying drone camera)
(c) OWILD Domain
(Real indoor/outdoor locations, smartphone camera)

Figure 2.13: Data collection differs for each domain of 3D-ODDS. OTURN uses a high resolution camera turntable setup, generating 3D meshes using structure-from-motion. OOWL is captured using a drone camera, mid-flight. OWILD depicts objects in diverse indoor/outdoor locations with smartphone cameras.

paradigm and 3D-ODDS dataset provide an excellent foundation for future improvements in test-time refinement and generalizable reconstruction. For example, it may be worthwhile to explore more complex architectures, high level learned priors, topological modifications, and generative/adversarial formulations. They may lead to more powerful refinements, but also significantly increased challenges in avoiding degenerate solutions.

2.8.1.3 Potential on Societal Impact

REFINE is a relatively lightweight instance-based, class-agnostic postprocessing step. It does not rely on any dataset to train on; its effectiveness is due its formulation, designed architecture, and proposed loss functions. Thus, we do not anticipate immediate negative environmental, fairness, or privacy concerns directly resulting from REFINE. However, it requires a black-box separate single view reconstruction network S which reconstructs the original meshes. In real world deployments we encourage understanding the design and training procedure of S, especially its potential biases and security/privacy concerns which may be problematic in some neural networks [106, 103].

2.8.1.4 REFINE Architecture

The feature map encoder is based on the first two convolutional layers of ResNet-18 [28]. The dimension of all 8 graph convolution layers used is 128, and each is followed by a ReLU nonlinearity. V_{dis} is predicted with a single fully connected layer, while V_{sConf} is predicted with fully connected layers of sizes 32, 16, and 1 (a ReLU follows each except for the last, which uses sigmoid). The feature map encoder is initialized using ImageNet [12] classification pretrained weights, while all other weights are randomly initialized; no weights are frozen during optimization. We use the PyTorch3D differentiable renderer [39], which is implemented based on [56]. All hyperparameters were tuned with a small portion of RerenderedShapeNet, disjoint from the test set.

2.8.1.5 Loss Functions

The weights chosen for the loss functions are $\lambda_{Sil} = 10$, $\lambda_{Isym} = 80$, $\lambda_{Vsym} = 20$, $\lambda_{SymB} = 0.0005$ $\lambda_{Dis} = 100$, $\lambda_{Nc} = 10$, and $\lambda_{Lp} = 10$. We found that this configuration works well overall in practice; however, they are not overly sensitive and changing them by $\pm 25\%$ didn't change results significantly. Beyond this range, we observed that these weights operate intuitively as one would

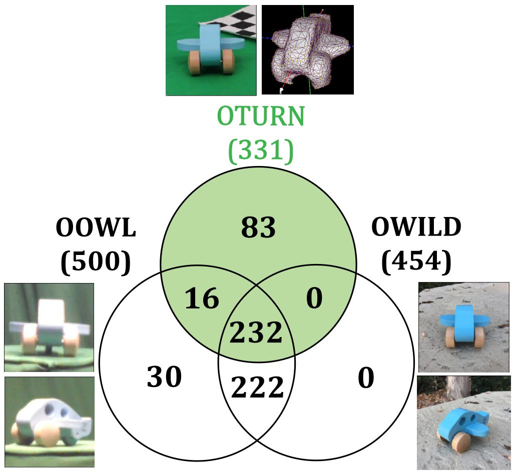


Figure 2.14: A venn diagram of objects that can be found in the 3D-ODDS dataset's three domains: OTURN, OOWL, and OWILD. 232 objects can be found simultaneously in all three domains.

expect (as illustrated in Figure 9 of the main paper). In general, they are not difficult to tune and practitioners can modify them accordingly with their use case. For example, one might increase the weight of λ_{dis} if they are confident that in their use case, input reconstructions are already of relatively high quality. This would effectively apply a stronger prior towards minimizing the displacements' magnitudes. Alternatively, if only symmetric objects are considered λ_{SymB} can be increased.

Additionally for the symmetry losses, there are methods to predict planes of object symmetry [18, 125] but we found them to be unnecessary since most reconstruction methods output semantically aligned meshes for objects of the same class. In general, the objects are aligned so that $\mathcal{Z}$ is the vertical plane with $\vec{n} = [0,0,1]^\intercal$. We adopt this convention in all our experiments. For the image rendering based symmetry loss, we also tried to use differentiably rendered normal maps and depth maps instead of only silhouettes. However, we found that this increased the computational complexity, and resulted in nearly the same performance.

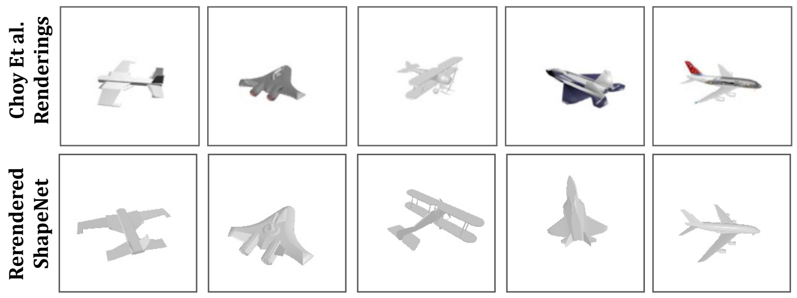


Figure 2.15: Comparisons between ShapeNet renderings from Choy Et al. [10] and Rerendered-ShapeNet. Both use the same 3D models, but a domain gap is intentionally created through viewpoint, lighting, and rasterization implementation differences in the rendering process.

2.8.1.6 Time Efficiency

Ideally, test-time shape refinement postprocessing should support any mesh and be fast. REFINE intrinsically satisfies the first requisite, since it is black-box, class-agnostic, and allows variable number of vertices per mesh. Optimization from scratch converges in relatively few iterations, approximately 400 (i.e. 400 forward and backward passes). This requires about 90 seconds on a GTX 1080Ti GPU. Moreover, because instances are treated independently, the refinement is trivially parallelizable. Since 4 instances fit on a GPU, a two GPU server trivially achieves a per-instance refinement time of $90/(4*2) \approx 11$ seconds, which is effective in terms of the second requisite.

2.8.2 Dataset Additional Details

Three new datasets are proposed in this paper: 3D-ODDS, RerenderedShapeNet, and ShapeNetAsym. All datasets will be publicly released upon publication. More details about these datasets are provided as follows.

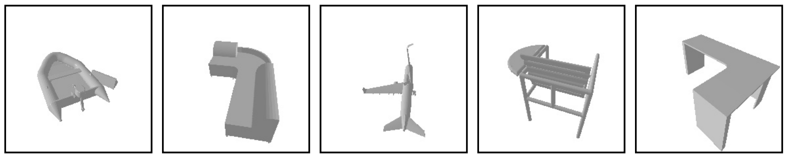


Figure 2.16: Example images from the ShapeNetAsym dataset.

2.8.2.1 3D-ODDS

The proposed 3D-ODDS dataset contains multiview objects in 3 domains, as illustrated in Figure 2.13. The first domain is the contribution of this paper, OTURN, which is taken in the lab using a turntable and DSLR camera. 331 objects were imaged with dense pose coverage; 3 elevation angles and 72 azumuth angles (5° increments), for $331 * 3 * 72 = 71,496$ total images which are of high resolution with simple backgrounds. All the OTURN images for an example airplane object is provided in Figure 2.20. These images were then used to reconstruct a mesh for each object, using structure-from-motion software [60]. The second domain is OOWL [29], with multiview (45° azimuth increments) images of objects collected in the lab using a drone camera. These images have a green background and can be blurry, due to camera shake from flight. The third domain is OWILD [30], which contains multiview images of objects (also 45° azimuth increments) in diverse real world indoor/outdoor locations. These images are taken with a smartphone camera. A venn diagram of the objects found in these domains is given in Figure 2.14; 232 objects can be simultaneously found in OTURN, OOWL, and OWILD. More example objects in the 3D-ODDS dataset are shown in Figures 2.18 and 2.19.

Note that some imperfections are present in the mesh scans, as a natural consequence of real-world 3D data collection. This can be due to absence of texture or difficult material reflectance properties. To account for this, we manually annotated each mesh's quality; there are 101 excellent quality meshes, 198 high quality meshes, and 32 low quality meshes. High quality meshes are characterized by overall geometrical resemblance to the true shape; some small superficial noise artifacts may exist. In all experiments, we only considered objects found in OTURN, OOWL, and OWILD with excellent/high quality meshes; this subset consists of 212 objects.

2.8.2.2 RerenderedShapeNet and ShapeNetAsym

RerenderedShapeNet matches the ShapeNet [7] models in the test set given by [10]. However, a small domain gap is intentionally induced compared to the images from [10] through differences in the rendering process. This allows us to measure the robustness of SVR methods to domain gaps of various sizes between training on [10] and inference (on RerenderedShapeNet, ShapeNetAysm, Pix3D, or 3D-ODDS). In particular, [10] is rendered textured with Blender's Eevee engine [11] at distance 0.8, uses 2 sun light sources 180 degrees rotated from one another, with specular and diffuse shading disabled. Meanwhile, RerenderedShapeNet is rendered textureless with PyTorch3D's Hard Phong shading [39] at distance 1, uses a point light source at $(0,5,-10)$, with ambient intensity 0.3, specular intensity 0.2, and diffuse intensity 0.3. Images in both have an elevation of 40° and randomly samples azimuths uniformly. An illustration of differences between RerenderedShapeNet and renderings from [10] is shown in Figure 2.15. In total, RerenderedShapeNet contains 8629 images and meshes.

We also introduced an asymmetric subset of RerenderedShapeNet called ShapeNetAsym containing 1259 images and meshes. The meshes are all asymmetrical, in the sense that each mesh has a symmetry loss $L_{Isym} < 0.01$ for $\lambda_{SymB} = 1$ and $\sigma_{j,k} = 1$. Some examples from ShapeNetAsym can be found in Figure 2.16.

2.8.3 Evaluation Details

2.8.3.1 Analysis of Variance Results

Analysis of Variance (ANOVA) [17] is a commonly used statistical model and hypothesis testing framework for splitting observed variability into systemic factors and random error.

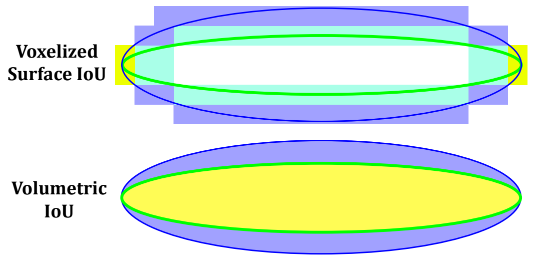


Figure 2.17: A 2D illustration of why a surface-based coarsely voxelized IoU metric (top) can be inaccurate, compared to the standard volumetric IoU (bottom). Surface-based voxelized IoU heavily underrepresents the intersection-based similarity of the two shapes compared to the volume based approach.

In particular, it can be used to model the influence of categorical independent variables (i.e. "factors") on a continuous dependent variable, to check if they are statistically significant. Due to its hierarchical structure, 3D-ODDS has 3 factors: class (14 levels, one for each class), domain (3 levels in OTURN, OOWL, OWILD) and pose (8 levels from 45° viewpoint azimuth increments). This suggests a 3-way ANOVA with blocked design, to account for object-based variability and dependencies (i.e. each object comprises a block). Our dependent variable in this case is F-Score after REFINEment of an OccNet. All factors, pairwise interaction effects, and triplet interaction effects were found to be statistically significant at the $\alpha = 0.05$ level. total variability was decomposed into 13% class, 2% pose, 1% domain, 19% object instance. Interaction effects between (class,domain), (class, angle), (domain, angle), and (class, domain, angle) were found to be 7.6% , 6.8% , 0.3%, and 2.5% respectively, for 17.2% in total attributed to interaction effects between the factors.

Note that ANOVA has several assumptions. The dependent variable should be additively influenced, and ideally errors should be independent, homoscedastic, and Gaussian (though ANOVA is considered relatively robust to some departures [59, 21], due to the central limit theorem). In light of this, we suggest viewing these ANOVA results as a simple summary heuristic useful for gaining further intuition and insight into 3D-ODDS, rather than dogma.

2.8.3.2 Metrics

We detail the metrics used in the main paper below. For Pix3D, we follow the practice of [76] and exclude images where the object is truncated resulting in 5325 test instances. The meshes used in this work have approximately 1500 vertices, although REFINE can handle much larger meshes (the only limitation being GPU memory).

The **Earth Mover Distance (EMD)** measures distance between point clouds S_1, S_2 sampled from two meshes, by solving the assignment optimization problem given by

$$d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2, \tag{2.12}$$

where ϕ is an optimal bijection. Because exactly computing EMD is too expensive, we utilize the approximation given by [16]. Like [76], we sample 2048 points from the reconstructed mesh and target mesh, scaled so it fits in a sphere of radius 1. For this metric, lower is better.

Chamfer-l_2 Distance (CD-l_2) is a widely used metric in the 3D literature [63, 23, 104, 16]. It computes the average nearest neighbor distance between points sampled from two meshes. Given these two sampled point clouds S_1, S_2, their Chamfer-l_1 distance is

$$d_{CD-l_2}(S_1, S_2) = \sum_{p \in S_1} \min_{q \in S_2} \|p - q\|_2^2 + \sum_{q \in S_2} \min_{p \in S_1} \|p - q\|_2^2. \tag{2.13}$$

Just like EMD, we follow [76] and sample 2048 points from the reconstructed mesh and target mesh, scaled so it fits in a sphere of radius 1. For this metric, lower is better.

F-score is formulated as the harmonic mean between precision and recall at a distance threshold between two shapes. Precision involves the number of points on the reconstruction which lie a certain distance to ground truth; recall measures completeness by the number of points on the ground truth which lie within a certain distance to the reconstruction. Like [76], we set this distance threshold to be 0.05 and sample 10000 points after rescaling to a sphere of radius 1.

Table 2.6: Extended, per-class results for reconstruction accuracy with no domain shift.

Metric	Method	Airplane	Bench	Cabinet	Car	Chair	Display	Lamp	Speakers	Rifle	Sofa	Table	Telephone	Watercraft	Mean
EMD $\downarrow$	AtlasNet [23]	6.3	7.9	9.5	8.3	7.8	8.8	9.8	10.2	6.6	8.2	7.8	9.9	7.1	8.0
	Mesh R-CNN [20]	4.5	3.7	4.3	3.8	4.0	4.6	5.7	5.1	3.8	4.0	3.9	4.7	4.1	4.2
	Pix2Mesh [104]	3.8	2.9	3.6	3.1	3.4	3.3	4.8	3.8	3.2	3.1	3.3	2.8	3.2	3.4
	DISN [118]	2.2	2.3	3.2	2.4	2.8	2.5	3.9	3.1	1.9	2.3	2.9	1.9	2.3	2.6
	MeshSDF [76]	3.3→2.5	2.5→2.1	3.2→3.0	2.2→2.0	2.8→**2.4**	3.0→2.4	4.2→**3.2**	3.5→2.9	2.6→**1.9**	2.7→2.4	3.1→2.7	1.9→1.7	2.9→**2.3**	3.0→2.5
	REFINEd OccNet [63]	3.0→**2.4**	2.4→**2.0**	3.1→**1.9**	2.3→**1.9**	2.8→2.6	2.4→**2.3**	5.4→3.4	4.8→**2.7**	2.5→2.4	2.8→**1.7**	3.4→**2.3**	1.3→**1.2**	2.9→2.4	2.9→**2.3**
CD-l_2 $\downarrow$	AtlasNet [23]	10.6	15.0	30.7	10.0	11.6	17.3	17.0	22.0	6.4	11.9	12.3	12.2	10.7	13.0
	Mesh R-CNN [20]	13.3	8.3	10.5	7.2	9.8	10.9	16.4	14.8	6.9	8.7	10.0	6.9	10.4	10.3
	Pix2Mesh [104]	12.4	5.5	8.2	5.6	6.9	8.2	12.3	11.2	6.0	6.8	7.9	4.7	7.9	8.0
	DISN [118]	6.3	6.6	11.3	5.3	9.6	8.6	23.6	14.5	4.4	6.0	12.5	5.2	7.8	9.7
	MeshSDF [76]	10.6→**6.3**	9.5→5.4	8.8→7.8	4.2→**3.5**	8.2→**5.9**	12.4→**7.3**	25.9→14.9	20.4→12.1	8.9→**3.4**	11.5→7.8	14.6→10.7	6.2→3.9	17.1→10.0	12.0→7.8
	REFINEd OccNet [63]	7.5→6.5	8.5→**5.3**	7.4→**5.2**	5.3→4.9	13.1→8.1	18.7→11.7	30.2→**13.1**	18.5→**10.5**	5.9→3.9	10.0→**7.1**	11.7→**8.8**	7.6→**3.5**	11.9→**9.1**	12.2→**7.5**
F-Score $\uparrow$	AtlasNet [23]	91	86	74	94	91	84	81	80	96	91	91	90	90	89
	Mesh R-CNN [20]	87	91	90	95	90	89	83	85	93	92	90	95	91	90
	Pix2Mesh [104]	88	95	94	97	94	92	89	89	95	96	93	97	94	93
	DISN [118]	94	94	89	96	90	92	78	85	96	96	87	96	93	91
	MeshSDF [76]	92→**96**	95→**97**	92→94	98→**98**	94→**97**	91→95	85→91	86→91	96→**98**	94→**96**	91→94	95→**98**	93→**95**	91→95
	REFINEd OccNet [63]	94→**96**	95→**97**	94→**97**	93→95	90→94	89→**96**	81→**92**	86→**92**	95→95	92→95	92→**95**	96→96	90→94	91→**96**
Vol. IoU $\uparrow$	AtlasNet [23]	39	34	21	22	26	36	21	23	45	28	23	43	28	30
	Pix2Mesh [104]	42	32	66	55	40	49	32	60	40	61	40	66	40	48
	DISN [118]	58	53	52	74	**54**	**56**	35	55	**59**	66	48	**73**	**56**	57
	REFINEd OccNet [63]	57→**59**	49→**55**	73→**73**	73→**74**	50→51	47→49	37→**43**	65→**65**	47→49	68→**69**	51→**52**	72→72	53→54	57→**59**

For more details, please refer to [93]. For this metric, higher is better.

Volumetric IoU is a standard metric [93] computed by the volume of two meshes' union divided by the volume of their intersection. Like [63], we obtain an unbiased estimate by randomly sampling 100k points in the bounding volume and checking if points are inside the meshes (scaled to radius 1). A higher score is better. Non-watertight meshes were made watertight with ManifoldPlus [34]. Note that MeshSDF [76] reports scores for their non-standard version of the 3D IoU which only accounts for coarsely $30 \times 30 \times 30$ voxelized 3D surface, not internal volume. As this can be highly misleading (see Figure 2.17), we instead use the conventional definition of 3D IoU in all experiments.

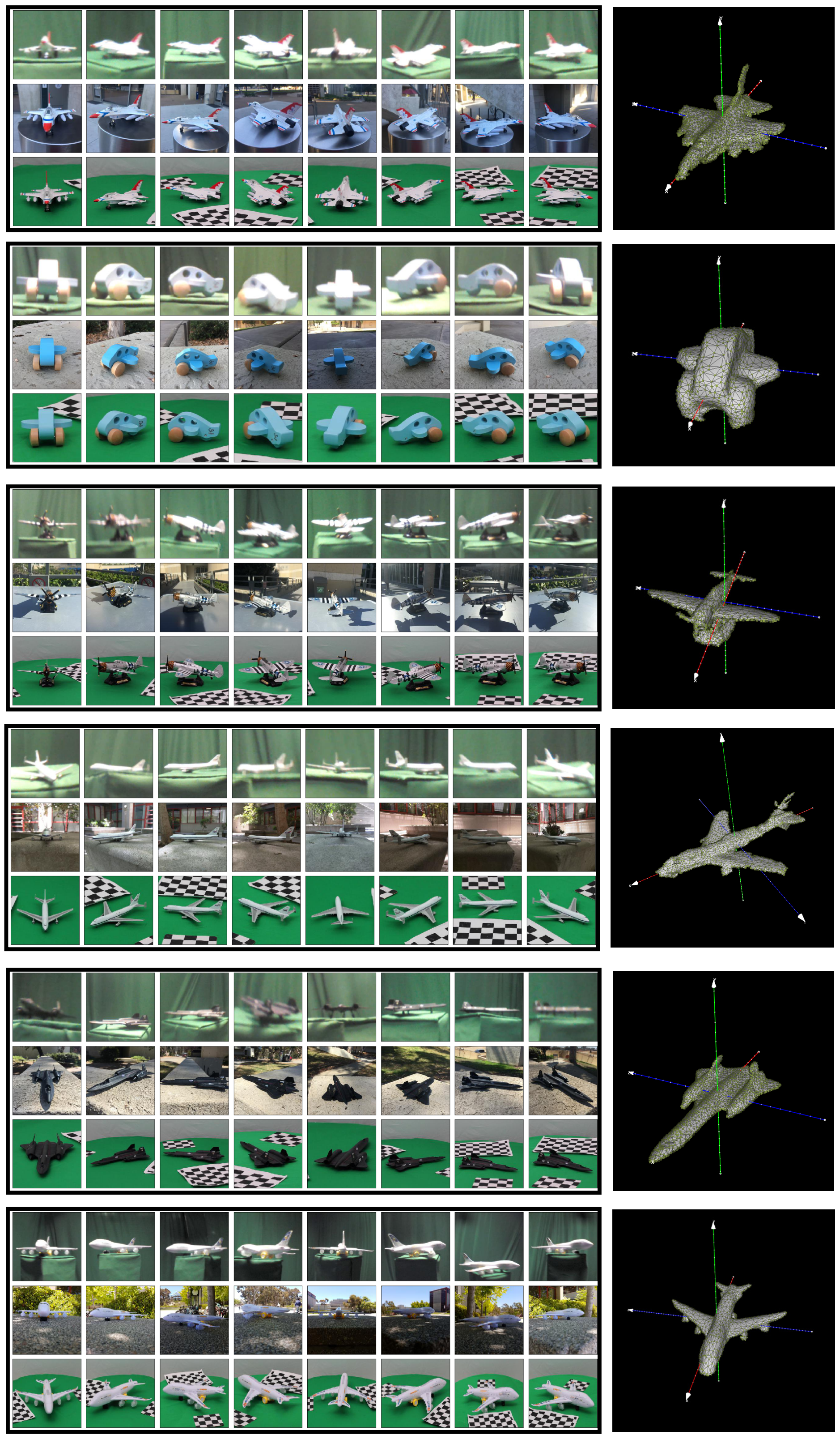


Figure 2.18: Additional example images and mesh for objects in the Airplane class of 3D-ODDS.

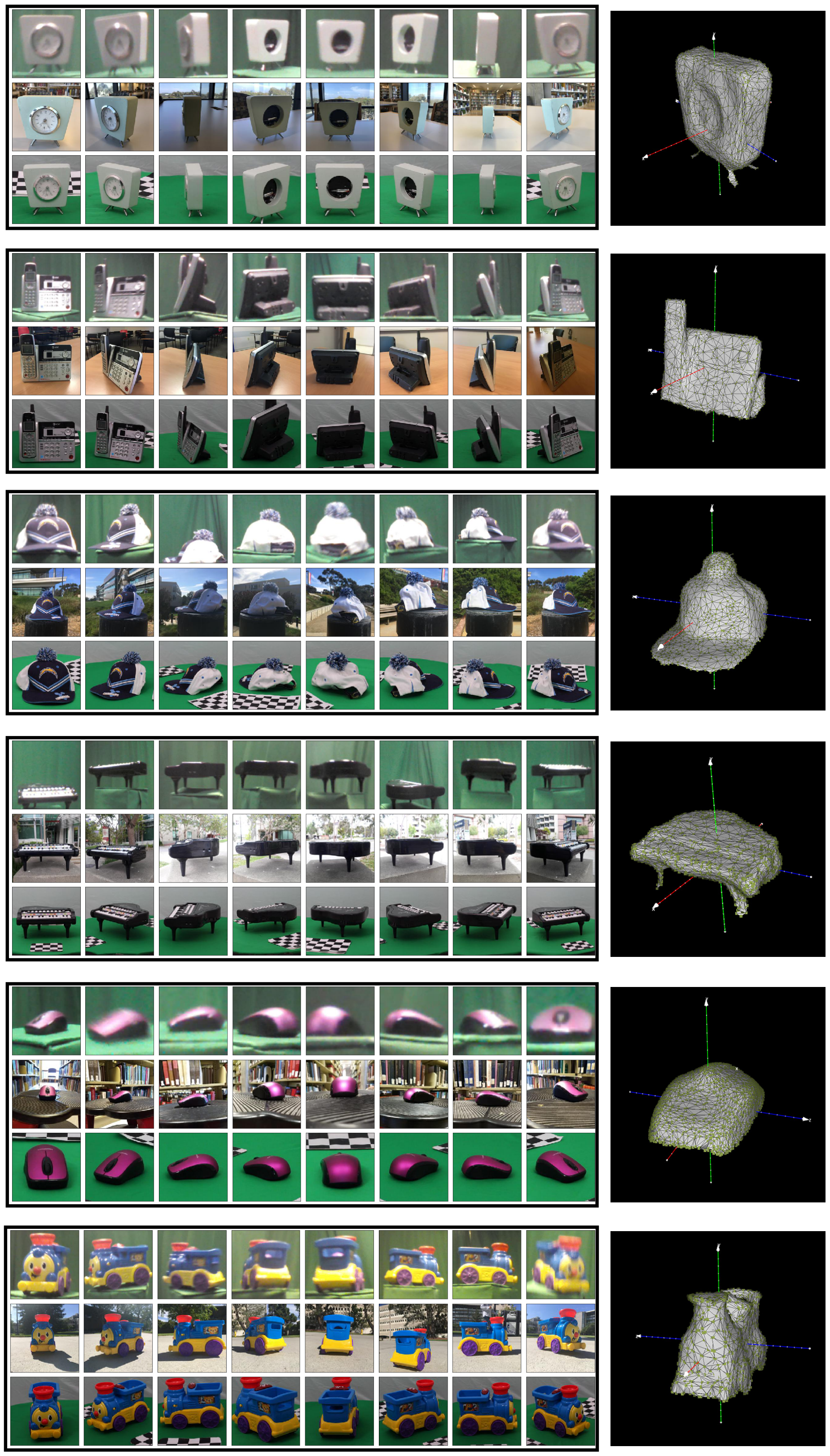


Figure 2.19: Additional example images and mesh for objects in 3D-ODDS.

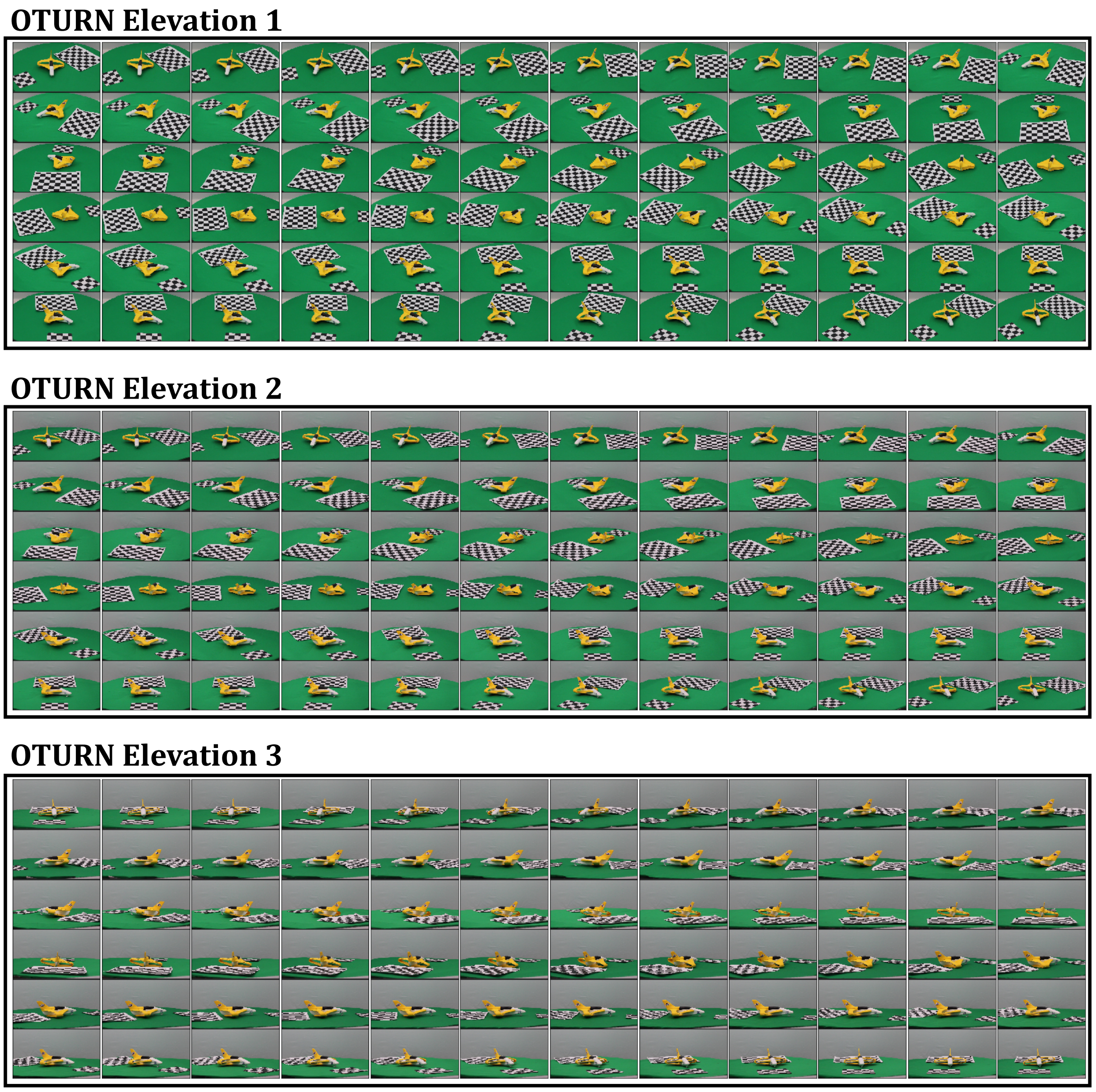


Figure 2.20: All 216 images for an airplane object in the OTURN domain of 3D-ODDS. There are 72 azimuth angles (increments of 5°) for 3 elevation angles.

	0°	45°	90°	135°	180°	225°	270°	315°	All
Airplane	38.4 → 45.5 (18.1 → 15.0)	38.5 → 46.6 (17.9 → 14.1)	41.2 → 53.8 (18.7 → 15.6)	37.5 → 46.7 (17.3 → 13.3)	37.6 → 48.0 (20.6 → 19.6)	39.7 → 48.5 (20.3 → 12.6)	38.1 → 50.0 (20.6 → 15.9)	38.2 → 47.3 (18.9 → 14.1)	38.7 → 48.3 (18.9 → 15.2)
Boat	25.0 → 33.9 (14.6 → 12.9)	41.9 → 46.1 (18.6 → 15.0)	41.5 → 49.1 (19.9 → 15.3)	36.1 → 45.0 (17.0 → 16.0)	22.4 → 30.2 (11.6 → 12.4)	36.7 → 43.4 (21.3 → 17.1)	38.7 → 48.6 (22.3 → 15.7)	36.5 → 43.5 (18.9 → 12.4)	34.9 → 42.5 (19.4 → 15.9)
Bottle	59.0 → 71.6 (18.3 → 8.8)	58.1 → 70.1 (25.6 → 17.1)	60.7 → 73.2 (14.4 → 13.9)	56.6 → 74.4 (23.0 → 14.3)	59.8 → 75.7 (19.4 → 14.5)	60.0 → 74.6 (20.5 → 12.5)	56.0 → 71.6 (21.3 → 16.2)	59.0 → 70.1 (16.0 → 13.9)	58.7 → 72.7 (19.6 → 13.8)
Bowl	40.5 → 46.4 (14.0 → 13.9)	40.0 → 46.5 (13.7 → 12.2)	40.8 → 48.0 (13.1 → 12.2)	39.0 → 43.3 (13.5 → 12.7)	36.8 → 42.0 (13.0 → 13.5)	38.6 → 43.7 (14.2 → 13.8)	38.8 → 46.0 (11.6 → 12.2)	40.4 → 46.1 (13.2 → 13.1)	39.4 → 45.2 (13.2 → 13.0)
Can	52.8 → 49.8 (24.5 → 24.8)	48.6 → 51.7 (19.7 → 24.1)	51.8 → 53.9 (24.2 → 25.6)	47.6 → 47.9 (22.9 → 25.5)	46.3 → 47.6 (22.2 → 24.8)	49.8 → 52.8 (22.8 → 26.5)	49.0 → 50.6 (24.8 → 24.6)	50.5 → 50.3 (24.5 → 27.1)	49.6 → 50.6 (23.1 → 25.2)
Car	30.0 → 33.2 (11.5 → 11.9)	53.5 → 54.5 (22.3 → 17.2)	54.7 → 62.6 (21.6 → 16.8)	51.3 → 52.8 (20.8 → 17.1)	26.9 → 33.6 (13.0 → 14.9)	47.0 → 50.3 (20.5 → 17.5)	51.6 → 61.5 (23.6 → 20.1)	51.7 → 52.3 (19.9 → 16.0)	45.8 → 50.1 (22.0 → 19.5)
Clock	35.0 → 40.1 (12.9 → 13.1)	33.0 → 36.8 (11.5 → 11.8)	33.2 → 36.1 (14.4 → 13.3)	32.7 → 36.0 (12.7 → 13.3)	34.2 → 37.1 (13.9 → 13.7)	35.2 → 37.7 (12.8 → 10.6)	36.7 → 39.5 (17.5 → 14.9)	33.7 → 38.1 (10.9 → 12.3)	34.2 → 37.7 (13.4 → 12.9)
Mouse	25.3 → 29.8 (13.3 → 11.3)	45.9 → 50.7 (16.4 → 12.8)	44.8 → 54.3 (18.0 → 12.3)	36.5 → 40.1 (16.4 → 13.5)	26.9 → 31.3 (15.1 → 15.3)	42.7 → 48.0 (19.3 → 13.8)	39.9 → 47.1 (16.2 → 13.7)	40.5 → 45.9 (16.8 → 12.5)	37.8 → 43.3 (17.9 → 15.5)
Hat	31.7 → 37.2 (11.5 → 9.5)	31.3 → 33.6 (11.4 → 8.6)	30.8 → 35.4 (11.0 → 10.1)	30.1 → 35.5 (12.7 → 9.8)	30.9 → 37.7 (11.1 → 9.1)	29.7 → 37.4 (10.8 → 11.1)	33.7 → 38.5 (11.5 → 10.8)	33.2 → 35.2 (11.9 → 8.3)	31.4 → 36.3 (11.5 → 9.7)
Keyboard	38.5 → 55.4 (26.2 → 21.1)	35.6 → 44.7 (21.6 → 21.1)	26.6 → 33.0 (17.7 → 17.0)	34.2 → 43.9 (24.8 → 22.3)	30.7 → 51.8 (23.7 → 26.4)	33.9 → 43.2 (22.3 → 18.7)	33.9 → 37.9 (21.0 → 16.6)	36.9 → 45.8 (27.1 → 18.1)	33.7 → 44.3 (23.2 → 21.2)
Piano	41.5 → 44.8 (16.3 → 12.3)	34.6 → 38.9 (16.9 → 13.3)	36.2 → 40.1 (11.7 → 11.5)	36.6 → 41.8 (14.2 → 10.4)	40.6 → 46.2 (15.9 → 14.1)	37.3 → 42.5 (16.8 → 13.0)	36.9 → 39.4 (10.0 → 10.8)	37.6 → 41.8 (11.8 → 11.8)	37.7 → 41.9 (14.4 → 12.3)
Remote	19.7 → 30.3 (14.7 → 21.9)	22.7 → 35.4 (18.6 → 25.2)	32.9 → 38.6 (23.7 → 27.9)	22.9 → 34.1 (18.6 → 24.2)	18.5 → 30.7 (11.1 → 19.1)	24.5 → 37.5 (14.8 → 25.0)	29.0 → 35.2 (20.4 → 26.5)	26.4 → 36.0 (18.8 → 25.6)	24.6 → 34.7 (18.3 → 24.4)
Telephone	28.5 → 36.3 (13.5 → 13.9)	28.1 → 36.2 (12.3 → 17.0)	31.6 → 37.9 (14.1 → 18.1)	28.4 → 34.0 (14.2 → 13.7)	26.4 → 34.5 (11.9 → 14.4)	27.8 → 36.7 (12.0 → 15.6)	29.7 → 35.0 (14.6 → 17.9)	29.5 → 35.5 (12.2 → 13.2)	28.7 → 35.8 (13.1 → 15.5)
Train	25.7 → 28.7 (12.8 → 12.9)	46.9 → 49.0 (23.8 → 21.2)	48.5 → 56.8 (24.4 → 20.8)	36.9 → 43.7 (20.4 → 18.5)	22.4 → 28.0 (12.2 → 12.6)	38.3 → 45.3 (20.7 → 20.3)	41.6 → 51.1 (22.3 → 20.9)	44.5 → 47.5 (23.5 → 19.5)	38.1 → 43.8 (22.2 → 20.9)
All	33.8 → 39.9 (18.4 → 17.7)	39.3 → 44.7 (20.0 → 18.6)	40.4 → 47.2 (20.3 → 19.9)	36.9 → 43.0 (19.3 → 18.1)	31.4 → 39.1 (17.7 → 19.0)	37.7 → 44.5 (19.5 → 18.2)	38.9 → 45.8 (20.0 → 19.4)	39.2 → 44.4 (19.6 → 17.6)	37.2 → 43.6 (19.6 → 18.7)

Figure 2.21: F-score performance and standard deviation (in parenthesis) on the 3D-ODDS dataset across the class and angle factors, before → after REFINEment. Colors correspond to accuracy after REFINEment, normalized across the table. Red indicates lower accuracy, green indicates higher.

	OOWL	OTURN	OWILD	All
0°	32.8 → 39.1 (17.0 → 16.7)	33.8 → 39.1 (19.6 → 19.4)	34.9 → 41.6 (18.6 → 16.9)	33.8 → 39.9 (18.4 → 17.7)
45°	38.9 → 44.2 (20.7 → 19.4)	41.2 → 45.8 (19.5 → 18.6)	37.8 → 44.2 (19.7 → 17.7)	39.3 → 44.7 (20.0 → 18.6)
90°	40.5 → 47.5 (20.5 → 19.8)	42.1 → 47.5 (19.2 → 20.1)	38.7 → 46.6 (21.1 → 19.9)	40.4 → 47.2 (20.3 → 19.9)
135°	34.5 → 40.8 (17.7 → 17.6)	39.4 → 44.4 (19.1 → 19.0)	36.8 → 43.6 (20.7 → 17.5)	36.9 → 43.0 (19.3 → 18.1)
180°	30.9 → 37.7 (17.8 → 18.5)	32.9 → 39.6 (19.0 → 19.4)	30.4 → 39.9 (16.1 → 19.0)	31.4 → 39.1 (17.7 → 19.0)
225°	37.6 → 43.9 (19.5 → 19.2)	40.3 → 46.1 (19.2 → 18.5)	35.3 → 43.4 (19.5 → 16.8)	37.7 → 44.5 (19.5 → 18.2)
270°	38.9 → 46.5 (20.5 → 20.5)	39.6 → 44.4 (19.0 → 18.5)	38.3 → 46.3 (20.6 → 19.2)	38.9 → 45.8 (20.0 → 19.4)
315°	37.1 → 42.4 (19.1 → 17.3)	41.5 → 45.8 (18.9 → 18.2)	39.1 → 44.9 (20.7 → 17.1)	39.2 → 44.4 (19.6 → 17.6)
All	36.4 → 42.8 (19.4 → 18.9)	38.8 → 44.1 (19.4 → 19.1)	36.4 → 43.8 (19.8 → 18.1)	37.2 → 43.6 (19.6 → 18.7)

Figure 2.22: F-score performance and standard deviation (in parenthesis) on the 3D-ODDS dataset across the domain and angle factors, before → after REFINEment. Colors correspond to accuracy after REFINEment, normalized across the table. Red indicates lower accuracy, green indicates higher.

Table 2.7: REFINEment in the presence of mild domain shift, namely RerenderedShapeNet reconstructions by ShapeNet trained networks OccNet, Pix2Mesh, and AtlasNet. REFINE achieves gains under all networks, classes, and metrics.

	REFINEd OccNet [63]				REFINEd Pix2Mesh [104]				REFINEd AtlasNet [23]			
	EMD ↓	CD-l_2 ↓	F-Score ↑	Vol. IoU ↑	EMD ↓	CD-l_2 ↓	F-Score ↑	Vol. IoU ↑	EMD ↓	CD-l_2 ↓	F-Score ↑	Vol. IoU ↑
Airplane	3.5 → **2.2**	20.6 → **11.4**	86 → **91**	38 → **40**	3.7 → **2.3**	22.3 → **11.0**	65 → **88**	12 → **22**	5.3 → **3.8**	41.9 → **18.2**	60 → **82**	5 → **13**
Bench	2.9 → **2.2**	28.6 → **17.0**	84 → **86**	20 → **20**	3.6 → **2.6**	28.0 → **19.9**	65 → **76**	9 → **11**	4.9 → **4.6**	50.0 → **37.7**	58 → **68**	5 → **8**
Cabinet	3.4 → **2.7**	17.0 → **14.8**	83 → **85**	45 → **46**	3.6 → **3.0**	20.2 → **16.4**	74 → **78**	37 → **39**	4.3 → **4.1**	30.7 → **19.9**	59 → **75**	14 → **17**
Car	2.9 → **2.5**	19.9 → **12.9**	86 → **87**	30 → **31**	2.7 → **2.3**	10.8 → **7.8**	85 → **90**	24 → **27**	7.6 → **4.8**	98.8 → **27.0**	44 → **72**	6 → **12**
Chair	6.5 → **5.4**	48.5 → **39.4**	72 → **76**	29 → **32**	6.3 → **4.5**	35.4 → **25.2**	60 → **73**	17 → **22**	6.8 → **5.0**	49.5 → **27.3**	53 → **71**	8 → **13**
Display	3.5 → **2.7**	30.8 → **18.1**	76 → **83**	31 → **37**	4.2 → **3.0**	28.0 → **17.4**	72 → **81**	25 → **32**	4.9 → **4.5**	43.1 → **30.0**	61 → **71**	10 → **14**
Lamp	8.9 → **6.3**	90.5 → **59.1**	68 → **73**	22 → **23**	9.2 → **7.0**	71.6 → **40.6**	50 → **66**	11 → **14**	10.2 → **7.5**	102.4 → **51.1**	44 → **62**	5 → **10**
Speakers	4.4 → **3.6**	29.8 → **22.3**	73 → **76**	43 → **44**	4.3 → **3.8**	31.4 → **25.5**	65 → **70**	36 → **38**	5.4 → **4.7**	46.6 → **27.7**	55 → **69**	13 → **17**
Rifle	6.5 → **3.9**	37.7 → **14.6**	86 → **91**	30 → **30**	3.5 → **3.4**	18.1 → **10.1**	76 → **91**	12 → **21**	6.3 → **4.5**	61.4 → **28.6**	70 → **84**	7 → **14**
Sofa	3.0 → **2.7**	23.8 → **17.9**	83 → **85**	48 → **49**	4.3 → **3.2**	24.8 → **21.8**	71 → **79**	34 → **40**	5.3 → **4.7**	48.0 → **31.1**	63 → **73**	15 → **19**
Table	4.5 → **3.9**	40.6 → **34.3**	72 → **77**	17 → **20**	9.3 → **6.2**	159.3 → **81.8**	30 → **44**	6 → **8**	8.9 → **7.4**	129.6 → **82.7**	36 → **47**	4 → **8**
Telephone	2.3 → **2.0**	10.9 → **8.0**	90 → **92**	48 → **50**	2.2 → **1.8**	10.9 → **8.2**	89 → **92**	40 → **44**	3.4 → **3.3**	33.6 → **20.8**	66 → **79**	11 → **16**
Watercraft	4.3 → **2.9**	42.4 → **23.5**	80 → **86**	32 → **36**	5.0 → **2.7**	32.7 → **14.0**	71 → **86**	16 → **27**	7.1 → **4.3**	76.8 → **25.5**	55 → **79**	6 → **15**
Mean	4.3 → **3.3** **(-1.0)**	34.0 → **22.5** **(-11.5)**	80 → **84** **(+4)**	33 → **35** **(+2)**	4.8 → **3.5** **(-1.3)**	38.0 → **23.1** **(-14.9)**	67 → **78** **(+11)**	22 → **27** **(+5)**	6.2 → **4.9** **(-1.3)**	62.5 → **32.9** **(-29.6)**	56 → **72** **(+16)**	8 → **13** **(+5)**

Table 2.8: REFINEment in the presence of mild domain shift, namely RerenderedShapeNet reconstructions by a ShapeNet trained Pix2Vox Network. REFINE achieves gains under all classes and metrics.

	REFINEd Pix2Vox [117]			
	EMD $\downarrow$	CD-l_2 $\downarrow$	F-Score $\uparrow$	Vol. IoU $\uparrow$
Airplane	4.5 $\rightarrow$ **2.3**	19.7 $\rightarrow$ **7.3**	71 $\rightarrow$ **93**	19 $\rightarrow$ **38**
Bench	2.9 $\rightarrow$ **2.5**	25.3 $\rightarrow$ **16.5**	72 $\rightarrow$ **80**	12 $\rightarrow$ **16**
Cabinet	2.8 $\rightarrow$ **2.8**	17.0 $\rightarrow$ **15.5**	79 $\rightarrow$ **80**	43 $\rightarrow$ **43**
Car	3.2 $\rightarrow$ **2.5**	26.3 $\rightarrow$ **14.6**	80 $\rightarrow$ **85**	29 $\rightarrow$ **32**
Chair	5.3 $\rightarrow$ **3.5**	30.2 $\rightarrow$ **18.4**	64 $\rightarrow$ **79**	23 $\rightarrow$ **32**
Display	3.9 $\rightarrow$ **3.2**	33.1 $\rightarrow$ **20.4**	71 $\rightarrow$ **80**	28 $\rightarrow$ **34**
Lamp	9.6 $\rightarrow$ **6.1**	78.0 $\rightarrow$ **44.6**	53 $\rightarrow$ **65**	18 $\rightarrow$ **23**
Speakers	3.5 $\rightarrow$ **3.5**	27.5 $\rightarrow$ **22.0**	72 $\rightarrow$ **75**	42 $\rightarrow$ **44**
Rifle	4.8 $\rightarrow$ **3.0**	23.2 $\rightarrow$ **12.2**	83 $\rightarrow$ **92**	25 $\rightarrow$ **35**
Sofa	4.1 $\rightarrow$ **3.2**	32.4 $\rightarrow$ **20.2**	72 $\rightarrow$ **82**	43 $\rightarrow$ **50**
Table	6.8 $\rightarrow$ **5.4**	121.3 $\rightarrow$ **62.5**	35 $\rightarrow$ **51**	8 $\rightarrow$ **11**
Telephone	2.1 $\rightarrow$ **2.1**	20.3 $\rightarrow$ **14.6**	79 $\rightarrow$ **85**	34 $\rightarrow$ **38**
Watercraft	5.1 $\rightarrow$ **2.7**	30.3 $\rightarrow$ **15.2**	75 $\rightarrow$ **87**	26 $\rightarrow$ **42**
Mean	4.5 $\rightarrow$ **3.3** (**-1.2**)	37.3 $\rightarrow$ **21.8** (**-15.5**)	70 $\rightarrow$ **80** (**+10**)	27 $\rightarrow$ **34** (**+7**)

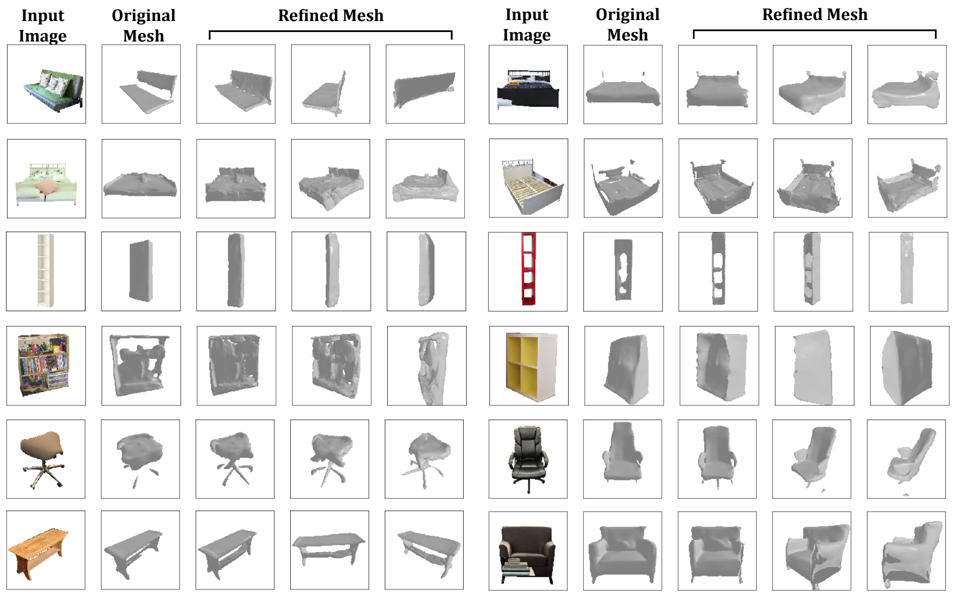


Figure 2.23: Occupancy Network mesh REFINEments for Pix3D images in the bed, bookcase, and chair classes.

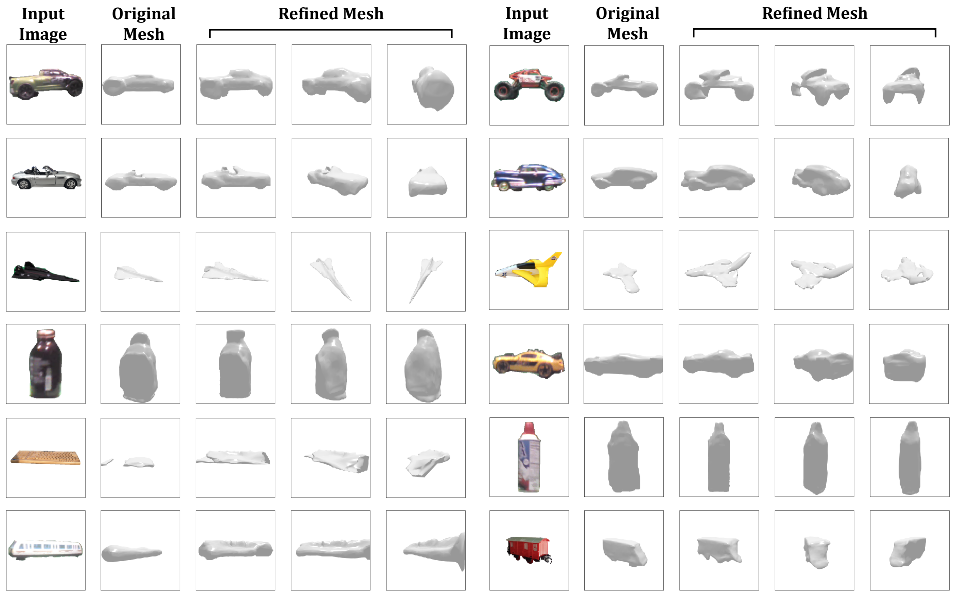


Figure 2.24: Occupancy Network mesh REFINEments for example 3D-ODDS images.

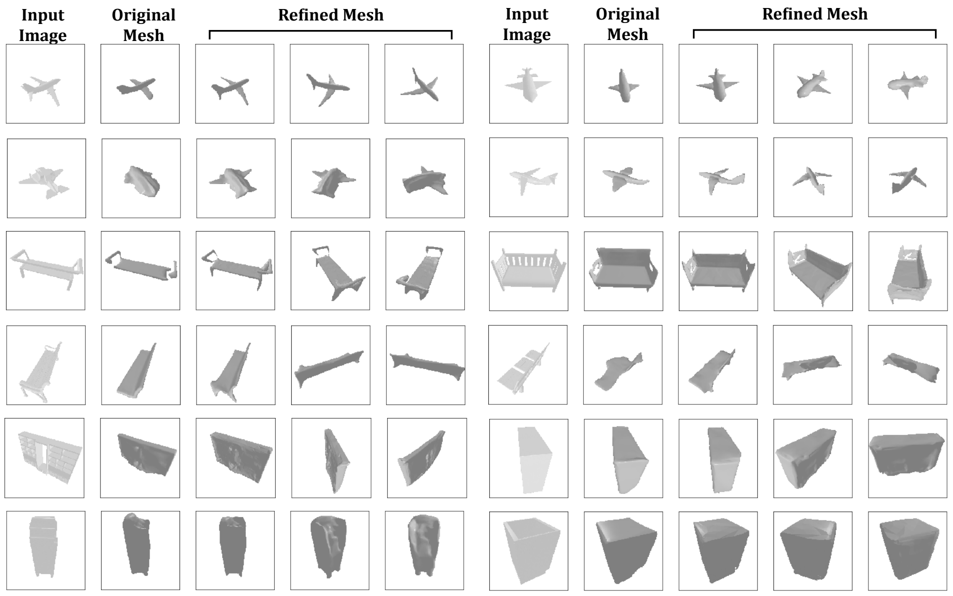


Figure 2.25: Occupancy Network mesh REFINEments for RerenderedShapeNet images in the airplane, bench, and cabinet classes.

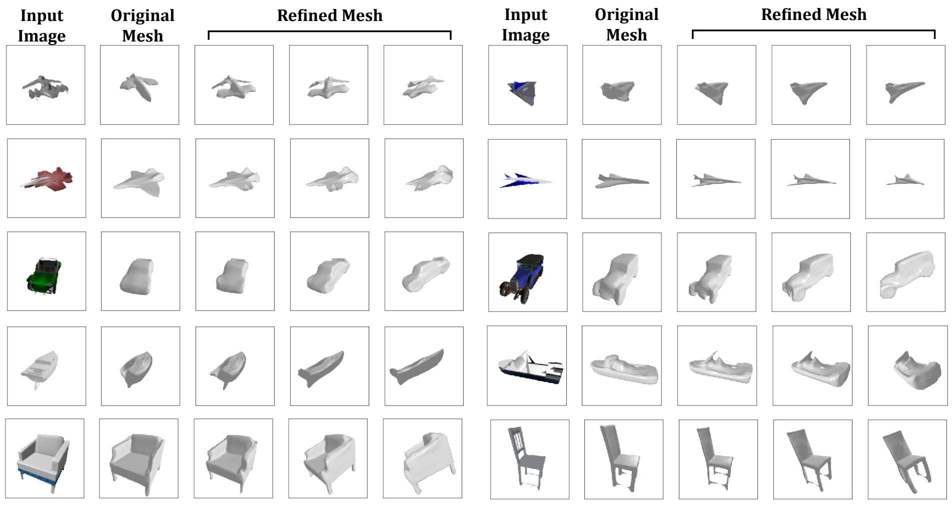


Figure 2.26: Occupancy Network mesh REFINEments for several ShapeNet images.

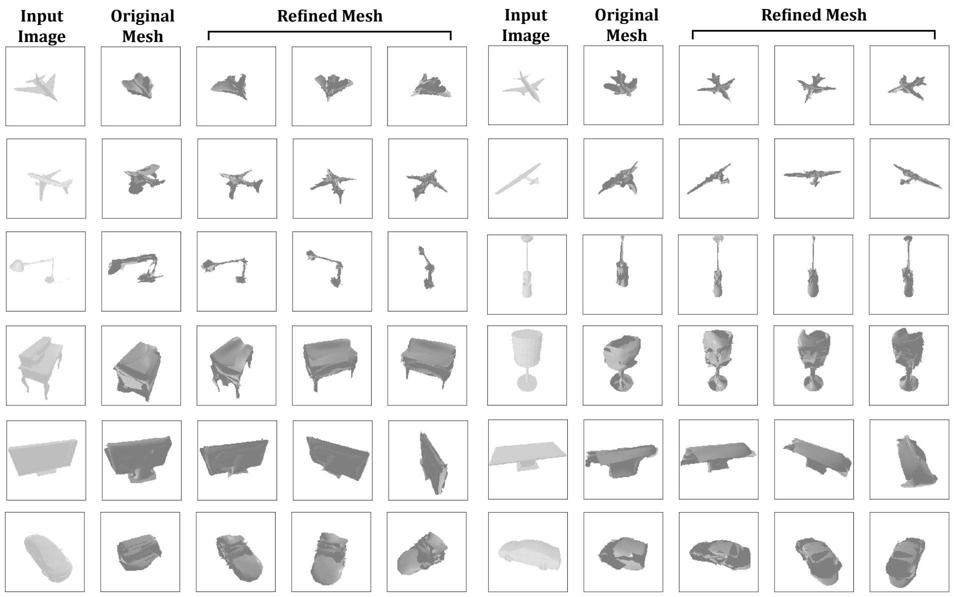


Figure 2.27: AtlasNet mesh REFINEments for several RerenderedShapeNet images.

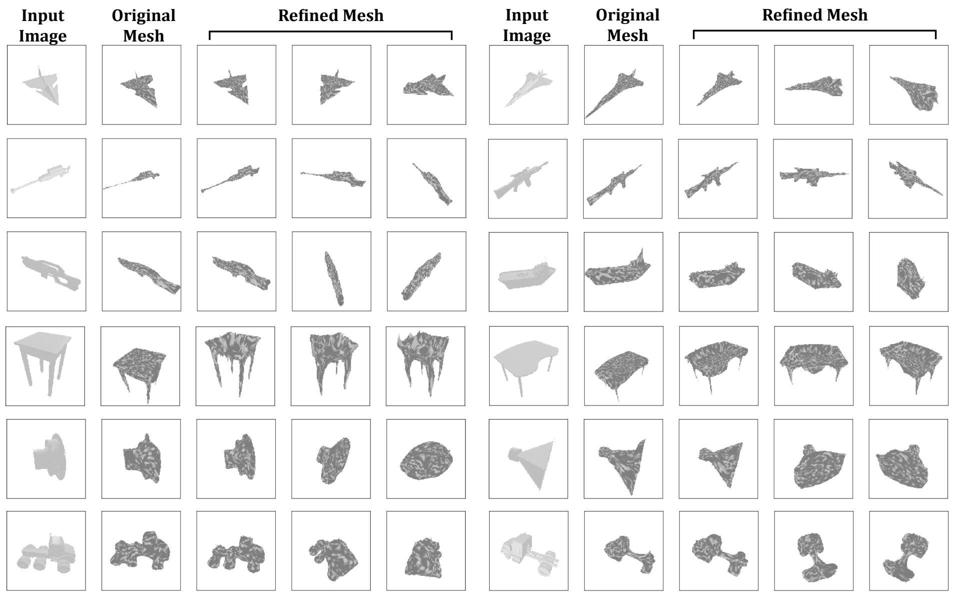


Figure 2.28: Pix2Mesh mesh REFINEments for several RerenderedShapeNet images.

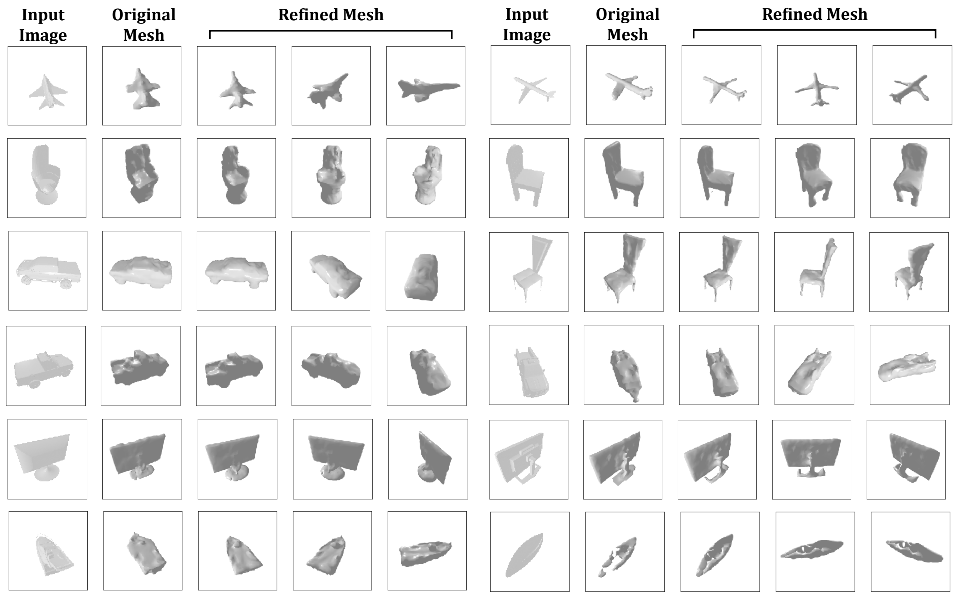


Figure 2.29: Pix2Vox mesh REFINEments for several RerenderedShapeNet images.

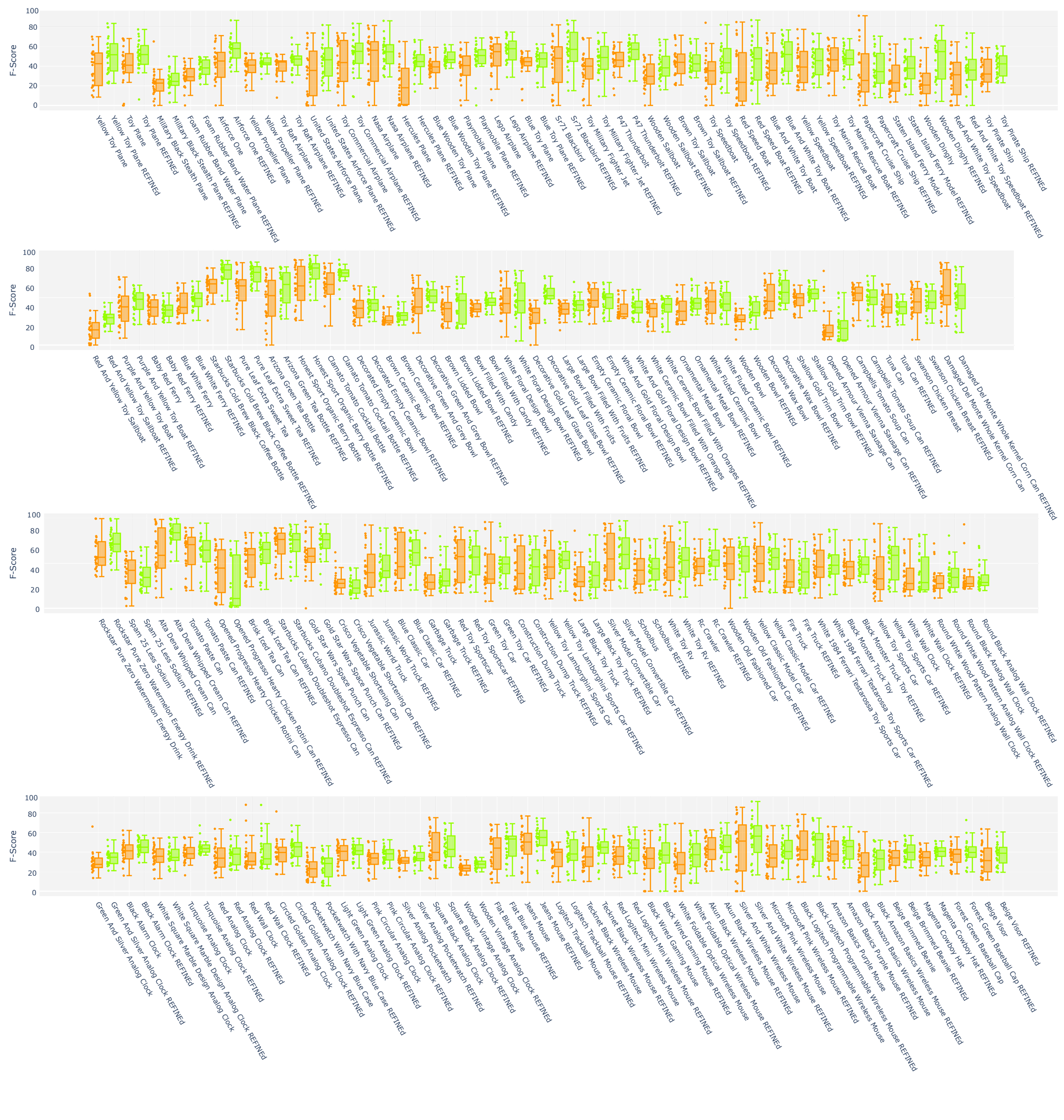


Figure 2.30: 3D-ODDS objects have 24 images (3 domains, 8 viewpoints). Reconstruction accuracies plotted before (after) REFINE as orange (green). Generally, REFINE improves performance invariance. Extended version of Figure 10 in the main paper.

Chapter 3

Evaluating Classification Robustness Through Adversarial Attacks

3.1 Introduction

Convolutional neural networks (CNNs) trained on large corpora such as ImageNet [12] have enabled significant advances in computer vision in recent years. While initially popular for recognition, these models have shown to be remarkably easy to train and transfer across vision tasks, and are now almost universally used across computer vision. Recently, however, this robustness has been questioned by some puzzling findings derived from adversarial CNN attacks [48, 71, 66, 6, 65, 87]. Although CNNs have excellent, even superhuman [27], recognition performance on randomly internet-collected test images, it is quite easy to generate images where they fail dramatically [91, 22]. In fact, most images that a CNN classifies correctly can be transformed into images that it cannot classify, by the addition of a very small adversarial perturbation. Most interestingly, this perturbation can usually be made so small as to be imperceptible, i.e. impossible to detect, by a human. This suggests that the space of images correctly classified by most CNNs is, at most, a countable dense subset of the space of images recognizable by humans, e.g. similar to the relationship between rational and real numbers.

This problem is of great concern for many applications. For example, smart cars depend on CNNs to make decisions that could have life or death consequences, security and surveillance systems rely on CNNs for identity verification, etc. The existence of many images capable of fooling CNNs poses a significant challenge to such applications. This has spurred interest in adversarial attacks [15, 48] and a literature has emerged in the area, with many variants of the problem being proposed. In result, there are at least four fundamental dimensions along which adversarial algorithms can differ: they can be 1) "white" [91, 22, 48, 71, 66, 6, 65] or "black"-box [8, 123, 87, 14, 32], depending on whether knowledge of the CNN model to attack is required, 2) "targeted" or "non-targeted," depending on whether the goal is to induce the network to make specific errors [91, 71, 6] or to simply make an error [22, 48, 66, 65, 123], 3) "digital" or "real-world" depending on whether the examples used in the attack are produced by an algorithm [91, 22, 87] vs. object manipulation in the real world [48, 15, 1], and 4) "single model" or "universal" depending on whether they aim to fool a single network [48, 66, 6, 123] or many models [65]. Interestingly enough, the relative difficulty of these problems does not always correlate with what would be intuitively expected. For example, it appears that most of the attacks designed to fool a particular CNN, e.g. AlexNet [47], also fool most other CNNs [70, 97, 49, 57, 120], e.g. VGG [82], Inception [90], or ResNet [28]. Similarly, some "black-box" attacks involving simple image transformations [14, 32] appear to be much more effective than "white-box" methods that require access to the CNN and optimization based on backpropagation style of algorithms.

In general however, it can be quite difficult to compare the merits of different algorithms. This is due to two main problems. First, most methods use perturbations that cannot be easily compared. While many authors rely on the standard of an image that is "perceptually indistinguishable from the original" to define a valid attack, it is not clear what the boundaries of "indistinguishable" are and no attempts have been made to define this concept. Instead, the standard is usually met by adoption of a very conservative attack strategy, e.g. the use of an "infinitesimally small step along some gradient direction". It is frequently unclear if the use of larger perturbations would enable the same algorithm to produce more successful attacks. Second, most adversarial works do not even attempt to compare performance with previous approaches. This is unlike most other areas of computer vision, where the ability to compare algorithms is considered critical to evaluate progress.

Recently, some works have started to address the second problem through a strategy that we denote as the "arms race". This exploits the fact that any attack procedure can be transformed into a defense, by 1) augmenting the training set, e.g. ImageNet, with examples produced by the procedure and 2) fine-tuning the network. While not guaranteeing full robustness against the attack [49, 97, 35], this defense strategy renders most attacks much less effective. Under the "arms race" paradigm, a new attack strategy is considered state of the art if it fools a network that implements defenses to previously known attacks [116]. The “arms race” captures the fact that, for practical applications, the only significant attacks are those for which no defenses are available. However, while knowing the attack procedure enables a defense, not all attacks are equally easy to defend. An important variable is the defense’s cost. For example, attacks that require more computation to defend against are more costly than attacks than can be thwarted with little computation. Similarly, attacks have different costs. For example, white box attacks can be rendered impractical by the simple use of a proprietary CNN. Overall, the most concerning attacks are those easiest to execute and hardest to defend against.

In this work, we consider the design of such attacks. We argue that the most successful attacks are those that leverage the limitations of computer vision, namely those based on perturbations that are easily produced by people but cannot be replicated by computers. This exploits the large imbalance between the cost of attack and defense in terms of the number of required examples. While an attack requires a few well chosen examples, its defense requires augmenting the training set with an extensive number of examples. Hence, while attacks can be generated manually, those that cannot be defended with computer generated examples are impractical to defend against. We then consider a set of image perturbations based on *variation of object pose.* This is an operation that can be implemented by a child (simply by rotating an object) but is very hard to defend against, due to the well known difficulty of synthesizing objects under different poses [72, 110]. We consider attacks using both small and large perturbations, due to *camera shake* (CS) and *pose variations* (PV). However, the study of such attacks requires a definition of which perturbations are valid. After all, extreme poses can confuse even humans. Unfortunately, common definitions, such as “infinitesimal gradient steps” or imperceptibility on side-by-side image comparisons, are not suitable for *large* perturbations. We argue that these can only be declared imperceptible given an attack context and seek definitions of imperceptibility suited for the object recognition context. This suggests a distinction between imperceptible perturbations (IPs), which survive image comparisons, and semantically imperceptible perturbations (SIPs), which are perceptible on image comparisons but have no impact on human ability to recognize objects.

Overall, this work makes three contributions to the study of adversarial attacks on CNNs. The first is a dataset of images of multiple object classes under camera shake and pose variation. The object classes are a subset of ImageNet, to enable the attack of ImageNet trained CNNs, and each object is imaged with extensive coverage of both small (camera shake) and large (pose variation) view variability. The second contribution is a procedure to determine which perturbations are imperceptible to humans, using Amazon Turk experiments. The procedure is designed to support many attack contexts and could be used to characterize many other types of attacks. We consider two contexts, image and object retrieval, that enable the differentiation between imperceptible perturbations and semantically imperceptible perturbations for object recognition; these can be thought of as small vs. large perturbations. A dataset containing camera shake and pose variation perturbations of the two types is finally assembled, to support the study of recognition attacks. A final contribution is an extensive experimental study of camera shake and pose variation attacks' performance, against multiple CNN models, trained on multiple datasets, and augmented with multiple defenses from the literature. This shows that pose attacks are highly successful against existing CNNs, previous defenses are ineffective against them, and even data collection can have limited effectiveness. Thus, while easy to perform, pose attacks can be difficult to defend.

3.2 Prior work

There is now a significant literature on adversarial attacks. The most popular setting is a non-targeted white-box digital attack of a single model [22, 48, 66]. The attack is usually an image perturbation based on an infinitesimal step along the gradient of the loss used to train the model, evaluated at the image [22, 48]. The simplest attacks reduce to one back-propagation iteration, computing derivatives with respect to the input image, and require a forward and backward pass through the network [22]. Many variants have been proposed, including different algorithms [71, 6, 57] or slight variations on the problem. For example, [65] proposed similar techniques for universal attacks, i.e. perturbations that fool many models, and [91, 71, 6] considered targeted attacks. These aim to induce specific errors, e.g. the classification of “apples” as “oranges”, using a somewhat more sophisticated optimization. All these methods are digital and can, in principle, be defended against by using the attack algorithm to generate augmentation data to retrain the CNN.

More recently, there has been interest in attacks based on object manipulation in the real world [48, 15, 1]. Some of these address specific applications, such as recognition by smart cars. For example, [15] investigated attacks based on the addition of stickers to traffic signs. This is much less general than the attacks now proposed, which can be applied to any object. Others have investigated the manipulation of images in the world, or the fabrication of objects with certain properties. For example, [1] devised an interesting procedure to fabricate objects that can consistently fool CNNs irrespective of viewing angle. While having some similarities to the attacks now proposed, this setup is substantially more complex than the one presented, which does not require object fabrication. Fabrication raises the cost of attack, by requiring access to knowledge of object fabrication, and drastically reduces the cost of defense, since it relies on algorithms that can be leveraged to produce defenses digitally. For example, because the objects fabricated by [1] have digital textures, their images can be rendered by computer. This is unlike real objects and textures, which are well known to be difficult to capture and render accurately under pose variation [110].

Perhaps most related to this work are previous efforts based on image transformations. For example, [14] has shown that black box attacks by simple image rotation can fool CNNs more effectively than white-box attacks based on gradient optimization. A recent extension of this idea uses spatial transformer networks to synthesize image transformation attacks more general than rotations [116]. This work again showed that image transformations are successful even on networks that implement defenses against gradient attacks. However, all these methods implement digital attacks, using algorithms that can in turn be exploited to defend against them. We propose a setting that generalizes these procedures, relying on real world image manipulation. This is much harder to defend against.

3.3 Using pose to attack recognition networks

There are several challenges to the study of adversarial attacks. A meaningful attack requires two images: a *true positive* x, i.e. a successfully classified image, and a perturbation x'. A first difficulty is that x' should be, in some sense, "identical" to x. Otherwise, it is illogical to ask the classifier to assign it to the same class, and the attack is ill-defined. We refer to this as the problem of *attack validity.* Consider the popular framework of attacks based on additive perturbations, $x' = x + \eta\delta$, where δ is a function of the gradient of the classification loss with respect to x [49]. In the absence of a criterion to test whether x and x' are "identical", validity is sought by constraining η to be very small, so as to make x' *visually indistinguishable* from x. However, this is not a full guarantee of validity, since a person with infinite time can frequently identify the perturbed image. There can also be moiré-like interference patterns that easily give the perturbation away. Some methods attempt to address the problem by thresholding the gradient, but this can produce salt-and-pepper artifacts. In general, it is difficult to guarantee that x and x' are indistinguishable.

For these methods, the validity problem follows from the lack of realism in the perturbations used for the attack. We refer to this as the *realism problem.* The difficulty is that δ *is not* a natural image. Hence, the methods above simply produce images at the "edge" of the space of natural images. While overly large steps along δ produce completely unrealistic images, a small enough η guarantees they are acceptable. Yet, because the perturbed images do not occur

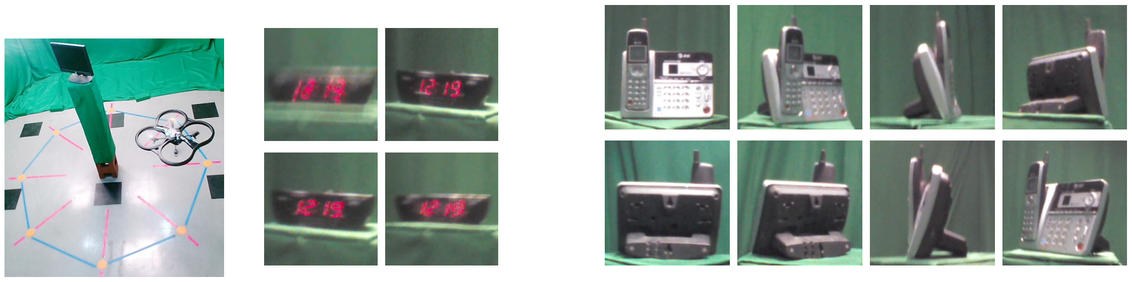

(a) (b) (c)

Figure 3.1: (a) Drone capturing images during flight. (b) Examples of varying levels of camera shake as the drone hovers. (c) Example images collected per viewing angle.

in the real world, the perturbations must be very small for the attack to remain valid. This leads to a third problem, which is the *small perturbation problem,* i.e. exclusion of attacks that are not immediate neighbors of the true positive. For most applications, such attacks are a much stronger concern than infinitesimal steps towards the edge of image space. For example, the shake and pose attacks proposed in this work can occur *naturally* during the operation of a vision system. This also implies that they are much easier to perform and thus much more likely to be executed – imagine a world where any child can hack a robot simply by showing it familiar objects in strange poses.

In summary, because there is lack of realism, validity can only be guaranteed by small perturbations. This has motivated a recent emergence of perturbations $x' = f(x)$ where f is no longer additive. Various functions have been proposed, from affine transformations [14] to affixing stickers on images [15, 4], to building 3D objects [1]. Because they are more realistic, the perturbations can be larger. On the other hand, large realistic perturbations exacerbate the difficulty of the validity problem since it is even harder to define an "indistinguishable" transformation. For example, a simple in-plane rotation can turn a '6' into a '9'. Similarly, if one is allowed to affix fur to a traffic sign, or repaint it, it will eventually stop being a traffic sign. While most works make an effort to select perturbations indistinguishable from the true positive in some form, this is never quantified. Beyond potentially compromising the significance of these studies, this makes it difficult to compare attacks.

In this work, we avoid these problems by introducing a new attack strategy based entirely on real-world object manipulations. This automatically eliminates the realism problem, since all attacks are based on natural images. We then propose a protocol to *guarantee* the validity of all attacks, by verifying that all perturbations are imperceptible to humans. Finally, we consider a domain (view transformations) that enables the characterization of the *size* of a perturbation. This enables the study of *both* small and large perturbations. We next discuss these contributions in detail.

3.3.1 Camera shake and pose manipulations

The ultimate goal of this work is to explore the space of attacks that are easy to perform (sometimes even arising naturally from real-world vision systems) but difficult to defend. The idea is to exploit image transformations that can be easily performed in the real world but are hard to replicate by computer. This leverages the fact that while an attack may be performed with a single example, most attacks can only be defended by training the classifier with many examples. When example collecting is costly, the defense becomes impractical. In this context, digital attacks which use algorithms to produce examples are easier to defend than real world attacks involving image manipulations not replicable by computer. Despite significant advances in photo-realistic rendering, it is still not possible to synthesize truly realistic examples from most object classes, at least without a significant investment in a sophisticated computer graphics infrastructure, rendering experts, etc. Hence, attacks with examples of objects under novel views or novel imaging conditions are difficult to defend. An additional benefit of these attacks is that they make it relatively easy to manipulate perturbation size, which correlates with the degree of view change. We illustrate this by introducing a family of attacks ranging from small transformations due to "camera shake" (CS – small variations of camera position) to larger transformations due to "pose variation" (PV – changes in viewing angle). These attacks are also particularly important because they are trivial to perform. For example, a child can shake a camera or rotate an object. In fact, they are inevitable in certain domains of computer vision, such as robotics, where objects can appear in many 3D orientations and the vision system is subject to nuisances such as shaking due to robot movement.

The first contribution of this work is a dataset to enable replicable studies of camera shake and pose variation attacks. For this, we relied on a drone-based imaging setup. A drone was flown around an object, as illustrated in Figure 3.1a, using markings on the ground to define picture taking stops at regularly spaced intervals. By collecting images at these stops under alignment with the markings on the ground, the drone assembled a set of views of the object corresponding to different orientations of the object in 3D. We refer to these as "object poses". Examples of multiple poses of an object are shown in Figure 3.1c. Within each stop, the drone was allowed to hover and collect several images of the object, as shown in Figure 3.1b. Due to the small hovering motion, many of these images are indistinguishable to the inattentive eye. They show the same pose of the same object, varying by very small translations of the camera and some amount of motion blur. We refer to this as "camera shake". The procedure was repeated for 20 objects per class from 23 different object classes. To facilitate attacks on existing object recognizers, these are classes represented in the ImageNet dataset, where the recognizers are trained. Overall, the dataset contains 30 camera shake images per pose and 8 poses for 460 objects, totaling 110,400 images. It is split into a *defense dataset* containing 16 objects per class and an *attack dataset* containing 4 objects per class. The defense dataset can be used to learn defenses against the proposed attacks. Each object is furthermore assigned a "frontal" pose by visual inspection, e.g. the frontal pose of the telephone in Figure 3.1c is that in the upper left corner. It should be noted that this setup is only necessary to enable *replicable studies* of the proposed attacks and to collect data for *defense purposes.* The attacks themselves can be performed by simply rotating objects.

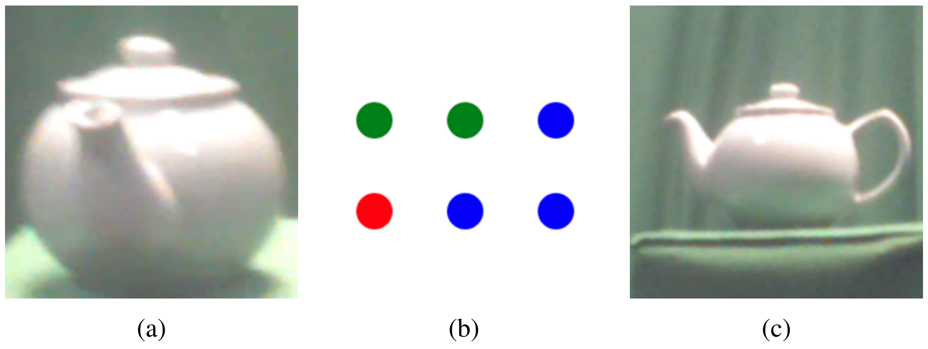


Figure 3.2: Turk experiment. (a) True positive (TP) x, shown for 750 ms. (b) Distractor task: count dots of some color. (c) After correctly counting, x' shown for 750 ms. Then, turkers asked if the image (object) has changed.

3.3.2 Characterizing indistinguishable perturbations

A difficulty of real world attacks, especially those involving larger perturbations, is to guarantee their validity. After all, under extreme viewing angles, objects can be hard to recognize even for humans. The second contribution of this work is a replicable procedure, based on Amazon Turk experiments, to characterize *indistinguishable perturbations* (IP). We start by proposing that perturbations can only be declared indistinguishable within a certain application context. For object recognition, we identify two contexts of interest. The first is *image retrieval* (IR). This addresses the question of whether a person can distinguish two images. However, we do not pursue the forensic definition of distinguishability commonly used in the literature. Instead, we limit the resources available to the person, by asking them to compare the perturbation to an image they have committed to memory. This is more closely related to object recognition than forensic comparisons.

The setup is illustrated in Figure 3.2. A turker is shown the true positive x for 750 ms (see Appendix for details) and asked to memorize it. The object disappears and the turker is asked to count the number of dots of some color in a 2x3 grid. This is a distractor task to prevent a purely iconic matching of image details. A second image x' is finally shown for 750 ms, and the turker is asked to indicate if x' was the *image* seen earlier. The second image can be of four types: the *true positive* x (15% of the time), a perturbation of x due to camera shake (35%), a perturbation of x due to a pose variation (35%), or an image of a *different object* (15%). All images used in this experiment are from the attack dataset of the previous section. From 30 frontal pose images, examples randomly sampled (with replacement) are used as the *true positive* per object. Camera shake perturbations are also "frontal" poses. Given a true positive, an *image pair* is created by sampling one of the 29 remaining frontal poses of the object. This procedure was used to produce 70 camera shake pairs per object. Pose variation perturbations use images from all object views. The set of perturbed images was created by randomly sampling 10 images from each pose, excluding frontal. Finally, for *different object,* examples were sampled randomly from frontal poses of other objects which belong to different classes. A probability of error was recorded per type of x'. These are denoted p^{TP}, p^{CS}, p^{PV}, and p^{DO}, respectively. The values of p^{CS} and p^{PV} are used as *indistinguishable perturbation rates* (IPR) for camera shake and pose variation perturbations. A high indistinguishable perturbation rate implies that turkers are not able to tell apart the perturbed x' from the true positive x. p^{TP} and p^{DO} are upper and lower bounds for the indistinguishable perturbation rate, respectively. For increased accuracy, each image-perturbation pair was evaluated by 3 turkers. A final quality control threshold was imposed per turker: those who scored above 10% for p^{DO} and those who did not score at least 90% for p^{TP} were excluded, as evaluation is trivial in these cases. For the remaining image pairs, those with less than two identical evaluations were eliminated. The indistinguishable perturbation rate was finally determined from the remaining evaluations, by majority vote.

So far, the experiments test if turkers can distinguish perturbed images. This is informative for image retrieval, but ultimately not the goal of object recognition. For example, the rotation of a digit by 30^o is a very perceptible image transformation. While most humans can easily tell the image has been rotated, this makes little difference for recognition. An equally large percentage of the population will be able to effortlessly recognize the rotated digit. In other words, recognition is invariant to the perturbation. We argue that, for recognition, it is also important to define the notion of *semantically indistinguishable perturbations* (SIPs). These are perturbations that may be noticeable but do not alter the image semantics. Semantically indistinguishable perturbations differ from indistinguishable perturbations in that they are tied to the semantics of interest for an application. For example, while a smart car only cares about the presence/absence of pedestrians on the road, a face recognizer aims to determine the person's identity. Hence, replacing the true positive by an image of another person is a semantically indistinguishable perturbation for pedestrian detection but not for face recognition.

An interesting property of the experimental setup above is that it can be extended to semantically indistinguishable perturbations by simply modifying the *context* of the experiment. This is done by changing the *question* asked to the turkers. Rather than asking them if x' is the same *image* as x, they can be asked if it is an image of the same *person, object, animal, scene,* or whatever the semantics of interest are for the application. In this work, we consider the generic *object recognition* (OR) context, asking the turkers if the two images are of the same object. The probabilities p^{CS} and p^{PV} then become *semantically indistinguishable perturbation rates* (SIPR) for camera shake and pose variation perturbations. They capture the degree to which the perturbations are imperceptible for OR. Note that a large pose transformation, clearly perceptible for image retrieval can easily be imperceptible for object recognition. This difference is captured by the two questions (*is this the same image?* vs. *is this the same object?*) that set different contexts for the experiment.

In summary, the probabilities p^{CS} and p^{PV} can be indistinguishable perturbation rates (IPR) or semantically indistinguishable perturbation rates (SIPR), depending on the context (image retrieval or object recognition respectively). Table 3.1 summarizes the rates observed on the Turk experiments. Several conclusions can be taken from the table. First, turkers' scores were excellent when spotting replicas of the true positive (IPR $> 99\%$) or rejecting images from different objects (SIPR $\leq 1\%$). This suggests that the experimental protocol is robust. Second, all rates were lower for pose variation than for camera shake. This was expected, because pose variation induces larger image variations. These results confirm the hypothesis that camera shake is a *small* perturbation, while pose variation is a *larger* perturbation. Note that only 7% of the pose variation perturbations were indistinguishable perturbations, while this held for 72% of the camera shake perturbations. Finally, it is clear that indistinguishability depends on context. While only 72% of the camera shake perturbations were indistinguishable perturbations, 92% were semantically indistinguishable perturbations. Similarly, while only 8% of the pose variation perturbations were indistinguishable perturbations, 82% were semantically indistinguishable perturbations.

3.3.3 Attacks and defenses

The third contribution of this work is a study of the difficulty of defending attacks based on real-world object manipulations. This is based on the image pairs declared as indistinguishable by the Turk experiment [1]. While experiments were performed for both indistinguishable perturbations and semantically indistinguishable perturbations, we report semantically indistinguishable perturbations only, since these are the most relevant perturbations for object recognition. Indistinguishable perturbation results are discussed in the Appendix. Three datasets were used to implement all defenses: 1) a subset of ImageNet containing all object classes used for attacks, denoted "ImageNet," 2) a subset of the defense dataset of Section 3.3.1 containing only frontal pose images, denoted "Frontal," and 3) the entire defense dataset, denoted "All". Every attack was performed on AlexNet [47], ResNet34 [28], and VGG16 [82].

To evaluate the impact of different object manipulations, the attacks were implemented with both camera shake and pose variation semantically imperceptible perturbations. For each true positive x, the associated perturbation x' was fed to the classifier and recognition rates (RR) r^{CS} and r^{PV} are recorded. Defenses were evaluated under the "arms race" strategy, by synthesizing examples with different attack methods and retraining the network on a dataset augmented with

[1]All data collected in this work is available at `http://www.svcl.ucsd.edu/projects/OOWL/CVPR2019_adversarial.html`.

Table 3.1: Turker imperceptibility rates for true positive (TP), camera shake (CS), pose variation (PV), and different object (DO) pairs. For image retrieval task, *indistinguishable perturbation rates* (IPR) is considered, while for object recognition task, *semantic IPR* (SIPR) is used.

	IPR (%) (Image Retrieval)	SIPR (%) (Object Recognition)
p^{TP}	99.7	99.8
p^{CS}	72.4	91.6
p^{PV}	7.5	81.5
p^{DO}	0.2	1.0

these examples. We considered methods from the two broad categories discussed above: 1) transformation based and 2) gradient based.

1. **Transformation based**

    - **Affine:** Random affine transformations with rotation less than 15 degrees.
    - **Blur:** Gaussian blur kernel with random standard deviation in $[0, 0.6]$.
    - **Blur-Affine:** Combination of affine and blur.
    - **Worst-of:** The worst-of-K method of [14]. Ten affine transformations are randomly sampled and the one of highest loss is selected.
    - **Color Jitter:** Image saturation and hue transformation according to [32].

2. **Gradient based**

    - **FGSM:** The fast gradient sign method of [49].
    - **ENS:** The ensemble adversarial training method of [97].
    - **IFGSM:** The iterative fast gradient sign method of [49].

3. **Standard** training is also experimented as baseline for comparison. The standard training method included random cropping and random horizontal flipping. The learning rate was set to 0.001.

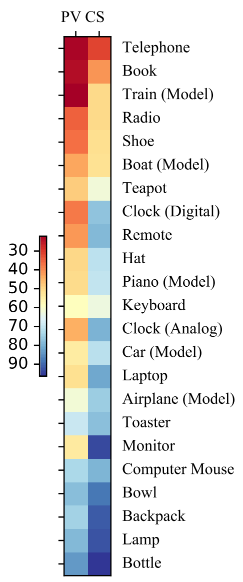


Figure 3.3: Per class RR for CS/PV SIP attacks.

3.4 Experiments

3.4.1 Implementation

All experiments were conducted with Pytorch. For training process, we found that at most 20 epochs were enough for the classifier to converge, and the maximum number of epochs was set to this value. Vanilla SGD was used as the optimizer and momentum was set to 0.9 for all classifiers.

3.4.2 Qualitative results

Attack and defense efficiency: A preliminary observation was that the attacks had similar effect on the three networks. While some models have higher accuracy than others, the relative drop in accuracy due to the attacks were nearly identical. Hence, for brevity, we only discuss average accuracy of the three models here. More detailed, per-model, results are given in the Appendix. Table 3.2 presents the recognition rates of camera shake and pose variation manipulation attacks, for networks with various defenses. Each defense was implemented on

Table 3.2: Recognition rates for camera shake and pose variation semantically indistinguishable perturbation attacks, under different defense methods and datasets. Recognition rates are averaged over AlexNet, ResNet34 and VGG16.

		Attack							
		CS	PV	CS	PV	CS	PV	CS	PV
Defense		ImageNet		Frontal		All		Avg	
	None	73.7	47.2	82.0	63.7	87.1	**79.1**	80.9	**63.3**
Transformation	Affine	71.8	45.1	83.4	58.8	85.2	76.5	80.1	60.1
	Blur	74.2	45.2	84.8	**64.1**	86.9	78.3	82.0	62.5
	Blur-Affine	75.4	47.5	83.5	60.0	**88.0**	76.6	82.3	61.3
	Worst-of	73.0	47.1	83.8	63.0	86.4	76.1	81.0	62.0
	Color Jitter	74.5	45.5	**86.4**	61.6	87.1	**79.1**	**82.7**	62.0
	Avg	73.8	46.1	84.4	61.5	86.7	77.3	81.6	61.6
Gradient	FGSM	72.9	**49.2**	84.7	61.1	83.2	74.3	80.3	61.5
	ENS	**75.7**	46.3	83.6	58.1	81.9	72.8	80.4	59.0
	IFGSM	71.8	47.0	82.8	55.5	83.3	70.0	79.3	57.5
	Avg	73.5	47.5	83.7	58.2	82.8	72.3	80.0	59.3

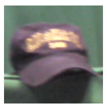
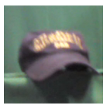
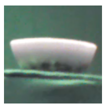
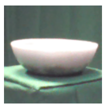
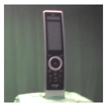
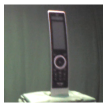
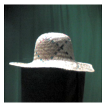
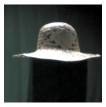
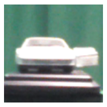
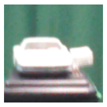
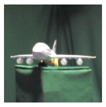
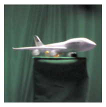
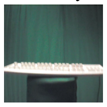
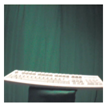
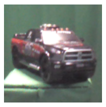
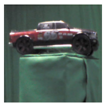
the three defense datasets and recognition rates are presented per defense method and dataset. Since all perturbations have been declared semantically indistinguishable perturbations by turkers, the human recognition rate is 1 on these experiments (under the assumption that turkers could correctly classify the true positive).

Various conclusions can be drawn. First, as expected, *pose variation is the more dangerous attack.* For standard ImageNet classifiers the recognition rate drops to almost half (from 70s to 40s), independently of the defense implemented. Second, *no defense method stands out.* While gradient methods achieve best performance for ImageNet training, transformations have superior performance for Frontal and All training. Within each category, relative performance varies with dataset and perturbation type. On average (as seen in the last column of the table), Color Jitter is the top defense against camera shake. Third, none of the defense algorithms improves significantly on no defense. In fact, *the absence of defense is the best defense against pose variation,* and close to the best (80.9 vs. 82.7) against camera shake, on average. Fourth, *data collection is a much more effective defense than algorithms.* Independently of the algorithm, recognition rates increase significantly from ImageNet to Frontal (10+ points) and increase further from Frontal to All (2 points). However, even the collection of data with camera shake and pose variation perturbations fails to fully defend against real-world object manipulations. The best performance against pose variation (none) has a recognition rate of only 63.3%. For camera shake the top recognition rate is 82.7%. All these observations support the hypothesis that real-world manipulations are a very effective tool to attack CNNs. Besides being trivial to perform, they can be very hard to defend. Since the collection of real data fails to produce a foolproof defense, it is questionable that digital defenses could fully neutralize these attacks. Clearly, simple digital defenses such as Affine or Blur transformations are ineffective.

In-depth comparisons of the table also challenge some common notions in the adversarial literature. One striking effect is the reversal of performance between gradient and transformation based methods with the defense dataset. Gradient methods work better on ImageNet, but are not effective when camera shake and pose variation perturbations are added to the defense set. This supports the hypothesis that they mostly push examples to the edge of the natural image space. Better coverage of these regions, by camera shake and more camera views, eliminates these methods' gains. For example, the average recognition rate of the gradient methods on the All defense dataset is 4 to 7 points weaker than using no defense algorithm at all. Transformation based methods perform significantly better on this dataset. In the adversarial literature, IFGSM and ENS have also been claimed to outperform FGSM. This is because IFGSM generates stronger adversarial examples and ENS decouples the adversarial example generator for the defender (by adding adversarial examples from a third party to the training set). However, this is nearly the opposite of the results on Table 3.2. On average, FGSM outperforms IFGSM and ENS. Again, this is likely due to the real world nature of the attacks. The fact that IFGSM and ENS are better defenses against digital attacks, seems to translate into no benefits for real world attacks.

Objects: It is also pertinent to ask which types of objects lead to more successful attacks. Figure 3.3 shows the recognition rate of camera shake and pose variation perturbations per object class. While camera shake leads to higher recognition rates for all objects, the recognition rate trend is similar for camera shake and pose variation. This suggests that attack efficiency is indeed

Table 3.3: Examples of IPs and SIPs, for CS and PV perturbations, that fool many classifiers. In all cases, TP is left, perturbation right. Also shown is ground truth class and # of classifiers fooled (out of 81, see Appendix).

IPs			
CS		**PV**	
TP: *Hat*	Fools *16*	TP: *Bowl*	Fools *36*
			
TP: *Remote*	Fools *21*	TP: *Hat*	Fools *20*
			
SIPs			
CS		**PV**	
TP: *Car*	Fools *13*	TP: *Plane*	Fools *20*
			
TP: *Keyb.*	Fools *13*	TP: *Car*	Fools *32*
			

determined by object properties. Finally, a "lack of symmetry" seems to be the object property most predictive of successful attacks – symmetric objects, such as bottles, lamps, and bowls are less effective attackers than less symmetric objects like telephones, radios, and trains.

Universal attacks: A final question is which attacks fool a large number of classifiers. Table 3.3 shows some examples of the most successful perturbations from this point of view (more in Appendix). Some interesting observations can be made. First, perturbations that are clearly noticeable under a forensic comparison (side-by-side images, infinite time) can become indistinguishable under the memory recall paradigm of Figure 3.2. Take the "bowl" and "hat" examples for instance, which were deemed indistinguishable perturbations by the turkers; the fact that these perturbations were deemed the same *image* as the true positive shows that the standard practice of determining attack validity by forensic comparisons is poorly suited for object recognition. Second, it appears that perturbations of all sizes can fool a large number of models. Note that the perturbations shown range from "insignificant" (almost imperceptible even

Table 3.4: Classifier accuracy for crafted vs random attack examples.

Dataset	ImageNet		Frontal	
Attack	CS	PV	CS	PV
Random	73.7	47.2	82.0	63.7
Crafted	51.3	33.0	66.2	49.3

on a forensic comparison, e.g. "remote") to "large" (significant pose variations, e.g. "car" on the bottom right). Overall, it appears that even very elementary natural perturbations can fool state-of-the-art classifiers.

Crafted attacks: Since the proposed attack happens naturally in the real world, one might criticize that this is different from L_p norm based attacks, which can be designed and crafted. Inspired by [14], which generates attacks by rotating the image and proposes a worst-of-K method to pick the most adversarial transformation , we implemented this for our attacks with $K = 5$ similar to the set up in [14]. Table 3.4 presents the results for no defense algorithm on random and crafted attack examples. These CS/PV attacks are intentionally crafted, by picking the CS/PV instance most likely to fool the network. They are more effective than the random attacks as expected.

3.5 Discussion

This work makes several contributions to the study of adversarial attacks that are easy to execute but difficult to defend, using a new setup based on real-world object manipulations. Unlike the standard practice in the literature, we considered both small and large perturbations, generated by camera shake and pose variation, and introduced a procedure for systematic collection of such perturbations. This was complemented by a replicable procedure to measure the imperceptibility of perturbations, using Turk experiments. These contributions enabled the creation of a dataset of small and large perturbations, imperceptible under two contexts of interest for object recognition. Experimental results comparing defenses based on many datasets, CNN models, and algorithms from the literature elucidated the difficulty of defending these attacks. None of the existing defenses were effective against them, and while better results were achieved with real world data augmentation, this is costly and not foolproof. These results suggest that more research is needed on defenses against "easy to perform" attacks and that the data now assembled can play an important role in this regard.

3.6 Acknowledgements

Chapter 3 is, in full, based on the publication of "Catastrophic Child's Play: Easy to Perform, Hard to Defend Adversarial Attacks", Brandon Leung, Chih-Hui Ho, Erik Sandström, Yen Chang, Nuno Vasconcelos, as it appears in Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. The thesis author was a primary investigator and author of this material, along with Chih-Hui Ho.

3.7 Appendix

3.7.1 Amazon Turk test times

One of the variables in the experimental protocol used to measure perturbation perceptibility is the time for which images are shown to the subjects. Several works have shown that neural activity exhibits signs of object recognition within about 200 ms of an image stimulus, with reaction times about 150 ms later. Preliminary experiments with a 350 ms viewing time showed that this was too little, at least for Turk experiments. Turkers only identified the TP as being the same image 55 percent of the time. While they did much better at rejecting different objects, this time was considered overall too aggressive. Subsequent experiments with a longer limit of 750 ms suggested that this was enough time. The IPRs obtained with the two settings are shown in Table 3.5.

Table 3.5: Preliminary Amazon Turk A/B testing results; turkers given 350 ms or 750 ms to remember images. Average imperceptibility scores in the the context of image recognition reveal similar trends relative to their respective upper bound p^{TP} and lower bound p^{DO}.

	p^{TP}	p^{CS}	p^{PV}	p^{DO}
350 ms	0.551	0.404	0.258	0.059
750 ms	0.977	0.798	0.106	0.010

3.7.2 Recognition rates (RR) for IP and SIP

Tables 3.6-3.7 summarize the RRs of IP and SIP attacks per model, defense dataset, and defense algorithm. While, in general, ResNet outperformed the other models, the effect of the attacks on the three models was quantitatively similar. Defense algorithms were more effective for IPs than SIPs. This is not surprising, since the former tend to be smaller perturbations. The largest gains were obtained by using defense datasets augmented with CS and PV perturbations ('All').

3.7.3 Example adversarial samples

Additional adversarial samples of CS-IP, PV-IP, CS-SIP and PV-SIP are provided in Tables 3.8, 3.9, 3.10 and 3.11 respectively.

Table 3.6: Recognition rate (RR) for IP.

		ImageNet				Frontal				All			
Defense		Alex	ResNet	VGG	Avg	Alex	ResNet	VGG	Avg	Alex	ResNet	VGG	Avg
Camera shake attack													
	None	76.0	78.5	69.2	74.6	82.3	88.2	88.5	86.3	83.0	92.3	92.5	89.3
Aug.	Affine	77.8	76.6	74.5	76.3	85.4	86.2	89.8	87.1	84.0	93.8	88.0	88.6
	Blur	76.5	80.1	72.5	76.4	83.0	84.2	90.5	85.9	81.4	94.9	89.8	88.7
	Blur-Affine	78.0	72.8	76.1	75.7	86.8	86.0	87.2	86.7	83.7	91.2	88.2	87.7
	Worst	68.0	77.2	72.0	72.4	88.7	88.8	88.5	88.7	84.0	91.8	90.0	88.6
	Color Jitter	77.3	78.5	70.3	75.4	87.4	90.4	91.9	89.9	89.9	92.6	88.4	90.3
	Avg	75.5	77.1	73.1	75.2	86.3	87.1	89.6	87.7	84.6	92.9	88.9	88.8
Adv.	FGSM	76.1	83.0	70.7	76.6	84.3	90.9	84.5	86.6	86.1	84.8	91.0	87.3
	ENS	74.1	82.0	78.2	78.1	87.6	83.7	86.7	86.0	82.5	81.2	89.6	84.4
	IFGSM	70.7	77.1	73.6	73.8	85.1	88.1	88.3	87.2	82.8	86.7	88.0	85.8
	Avg	73.7	80.7	74.2	76.2	85.7	87.6	86.5	86.6	83.8	84.2	89.5	85.8
Pose variation attack													
	None	79.5	81.1	72.2	77.6	80.6	79.7	80.9	80.4	78.3	91.8	84.5	84.9
Aug.	Affine	62.2	83.0	54.5	66.6	89.5	67.8	81.0	79.4	83.1	88.7	85.9	85.9
	Blur	78.4	85.5	63.8	75.9	80.0	77.4	75.4	77.6	83.6	91.9	83.3	86.3
	Blur-Affine	71.8	80.4	61.7	71.3	70.0	83.6	80.0	77.9	87.7	81.9	86.8	85.5
	Worst	56.8	84.2	65.3	68.8	85.2	86.2	81.8	84.4	81.4	86.1	77.6	81.7
	Color Jitter	78.9	88.9	73.8	80.5	79.3	85.5	87.3	84.0	84.4	94.5	88.9	89.3
	Avg	69.6	84.4	63.8	72.6	80.8	80.1	81.1	80.7	84.0	88.6	84.5	85.7
Adv.	FGSM	83.8	90.7	57.8	77.4	83.6	82.3	84.1	83.3	80.7	83.1	83.3	82.4
	ENS	66.7	78.2	57.9	67.6	88.9	79.7	83.8	84.1	72.7	83.3	78.3	78.1
	IFGSM	34.4	71.8	60.0	55.4	72.2	76.9	75.0	74.7	85.7	88.1	78.5	84.1
	Avg	61.6	80.2	58.6	66.8	81.6	79.6	81.0	80.7	79.7	84.8	80.0	81.5

Table 3.7: Recognition rate (RR) for SIP

		ImageNet				Frontal				All			
Defense		Alex	ResNet	VGG	Avg	Alex	ResNet	VGG	Avg	Alex	ResNet	VGG	Avg
Camera shake attack													
	None	78.3	72.5	60.3	73.7	77.5	84.4	84.0	82.0	83.5	90.3	87.5	87.1
Aug.	Affine	68.7	75.5	71.4	71.8	80.4	83.1	86.7	83.4	82.1	89.0	84.5	85.2
	Blur	76.2	79.6	66.8	74.2	81.9	85.5	86.9	84.8	82.7	90.7	87.2	86.9
	Blur-Affine	79.2	72.4	74.4	75.4	81.1	85.1	84.4	83.5	84.5	90.4	89.0	88.0
	Worst	70.0	75.8	73.3	73.0	84.3	84.1	82.8	83.8	80.8	90.4	88.1	86.4
	Color Jitter	79.5	73.3	70.9	74.5	84.0	87.3	87.8	86.4	84.3	91.2	85.9	87.1
	Avg	74.7	75.3	71.4	73.8	82.3	85.0	85.7	84.4	82.9	90.3	87.0	86.7
Grad.	FGSM	75.4	77.1	66.2	72.9	81.4	87.1	85.5	84.7	79.6	81.7	88.2	83.2
	ENS	74.0	80.9	72.2	75.7	82.9	82.2	85.7	83.6	79.5	80.2	85.9	81.9
	IFGSM	66.2	75.4	73.8	71.8	78.1	85.1	85.3	82.8	81.4	84.5	83.9	83.3
	Avg	71.9	77.8	70.8	73.5	80.8	84.8	85.5	83.7	80.2	82.1	86.0	82.8
Pose variation attack													
	None	40.0	52.3	49.2	47.2	58.4	67.9	64.8	63.7	72.1	82.8	82.5	79.1
Trans.	Affine	36.5	52.2	46.7	45.1	53.3	61.1	62.1	58.8	68.5	81.6	79.6	76.5
	Blur	36.4	56.1	43.2	45.2	58.3	67.9	65.9	64.1	72.4	83.6	78.9	78.3
	Blur-Affine	41.5	54.7	46.4	47.5	53.0	65.3	61.8	60.0	69.1	80.1	80.4	76.6
	Worst	40.6	53.8	46.7	47.1	58.1	68.1	62.9	63.0	67.0	81.7	79.6	76.1
	Color Jitter	39.7	50.6	46.3	45.5	52.5	67.3	65.1	61.6	73.1	83.7	80.5	79.1
	Avg	38.9	53.5	45.9	46.1	55.0	65.9	63.6	61.5	70.0	82.1	79.8	77.3
Adv.	FGSM	41.3	59.5	46.8	49.2	55.4	65.4	62.6	61.1	70.5	72.8	79.6	74.3
	ENS	41.5	52.4	45.0	46.3	59.2	54.5	60.7	58.1	71.3	71.0	76.0	72.8
	IFGSM	47.6	49.9	43.4	47.0	54.3	51.6	60.6	55.5	66.7	68.8	74.4	70.0
	Avg	43.5	53.9	45.1	47.5	56.3	57.2	61.3	58.2	69.5	70.9	76.7	72.3

Table 3.8: Adversarial samples for CS IP

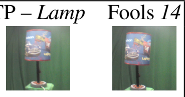
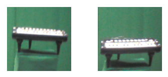
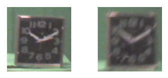
TP – *Hat* Fools *16*	TP – *Lamp* Fools *14* 
TP – *Piano* Fools *27* 	TP – *Remote* Fools *21*
TP – *Clock* Fools *19* 	TP – *Shoe* Fools *19*
TP – *Clock* Fools *15*	TP – *Backpack* Fools *12*

Table 3.9: Adversarial samples for PV IP

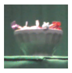
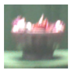
TP – *Bowl*	Fools *36*	TP – *Hat*	Fools *20*
TP – *Piano*	Fools *19*	TP – *Remote*	Fools *17*
TP – *Toaster*	Fools *15*	TP – *Bowl*	Fools *12*
			
TP – *Book*	Fools *21*	TP – *Bowl*	Fools *12*

Table 3.10: Adversarial samples for CS SIP

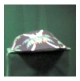
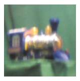
TP – *Book*	Fools *19*	TP – *Car*	Fools *13*
TP – *Keyboard*	Fools *13*	TP – *Boat*	Fools *12*
TP – *Clock*	Fools *47*	TP – *Remote*	Fools *20*
TP – *Hat*	Fools *13*	TP – *Radio*	Fools *11*
			

Table 3.11: Adversarial samples for PV SIP

TP – *Airplane*	Fools *20*	TP – *Car*	Fools *32*
TP – *Laptop*	Fools *19*	TP – *Train*	Fools *24*
			
TP – *Piano*	Fools *15*	TP – *Shoe*	Fools *14*
TP – *Bottle*	Fools *42*	TP – *Monitor*	Fools *22*

Chapter 4

Conclusion

In this thesis, we have detailed a new hierarchical multiview, multidomain image dataset with 3D meshes called 3D-ODDS. Its application was studied on two different vision tasks: single view 3D reconstruction and image classification.

In the case of single view 3D reconstruction, we rigorously showed that current methods were not sufficiently pose and domain invariant, both for 3D-ODDS and other datasets. To address this we proposed REFINE, a postprocessing 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.

For image classification, we considered adversarial attacks that are trivial to perform but difficult to defend. A framework for the study of such attacks was proposed, using real world object manipulations. Unlike most works in the past, this framework supports the design of attacks based on both small and large image perturbations, implemented by camera shake and pose variation from the 3D-ODDS dataset. Experiments using defenses based on many datasets, CNN models, and algorithms from the literature elucidate the difficulty of defending these attacks – in fact, none of the existing defenses is found effective against them.

Overall, these works show that robust generalization across domain, viewpoint, and class remains a significant challenge for computer vision. Furthermore, curated datasets with hierarchical levels of variability like 3D-ODDS are a powerful indicator and diagnostic tool. We believe that the 3D-ODDS dataset will remain relevant moving forward, inspiring future works towards robust and invariant methods in computer vision.

Bibliography

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.

[4] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017.

[5] Fatih Calakli and Gabriel Taubin. Ssd: Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, volume 30, pages 1993–2002. Wiley Online Library, 2011.

[6] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, and Hao Su. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 15–26, New York, NY, USA, 2017. ACM.

[9] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[10] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[11] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[13] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[14] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779, 2017.

[15] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017.

[16] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[17] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992.

[18] Lin Gao, Ling-Xiao Zhang, Hsien-Yu Meng, Yi-Hui Ren, Yu-Kun Lai, and Leif Kobbelt. Prs-net: Planar reflective symmetry detection net for 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3007–3018, 2020.

[19] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[20] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[21] Gene V Glass, Percy D Peckham, and James R Sanders. Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance. *Review of educational research*, 42(3):237–288, 1972.

[22] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[23] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[24] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), 2020.

[25] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] Chih-Hui Ho, Brandon Leung, Erik Sandstrom, Yen Chang, and Nuno Vasconcelos. Catastrophic child's play: Easy to perform, hard to defend adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9229–9237, 2019.

[30] Chih-Hui Ho, Pedro Morgado, Amir Persekian, and Nuno Vasconcelos. Pies: Pose invariant embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12377–12386, 2019.

[31] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[32] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. *CoRR*, abs/1804.00499, 2018.

[33] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3385–3394, 2019.

[34] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020.

[35] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

[36] Leyla Isik, Ethan M Meyers, Joel Z Leibo, and Tomaso Poggio. The dynamics of invariant object recognition in the human visual system. *Journal of neurophysiology*, 111(1):91–102, 2014.

[37] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5306, 2019.

[38] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[39] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, pages 1–1. 2020.

[40] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018.

[41] Hamid Karimi-Rouzbahani, Nasour Bagheri, and Reza Ebrahimpour. Invariant object recognition is a personalized selection of invariant features in humans, not simply explained by hierarchical feed-forward vision models. *Scientific reports*, 7(1):1–24, 2017.

[42] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[43] Eric Kauderer-Abrams. Quantifying translation-invariance in convolutional neural networks. *arXiv preprint arXiv:1801.01450*, 2017.

[44] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[45] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[46] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[47] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[48] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[49] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[50] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE, 2011.

[51] Rui Li and Hong Qiao. A survey of methods and strategies for high-precision robotic grasping and assembly tasks—some new trends. *IEEE/ASME Transactions on Mechatronics*, 24(6):2718–2732, 2019.

[52] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In *Advances in Neural Information Processing Systems*, 2020.

[53] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *European Conference on Computer Vision*, pages 677–693. Springer, 2020.

[54] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[55] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 969–978, 2019.

[56] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[57] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016.

[58] Yanxi Liu, Hagit Hel-Or, and Craig S Kaplan. *Computational symmetry in computer vision and computer graphics*. Now publishers Inc, 2010.

[59] Lisa M Lix, Joanne C Keselman, and Harvey J Keselman. Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance f test. *Review of educational research*, 66(4):579–619, 1996.

[60] Agisoft LLC. Agisoft metashape.

[61] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[62] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[63] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[64] Mateusz Michalkiewicz, Sarah Parisot, Stavros Tsogkas, Mahsa Baktashmotlagh, Anders Eriksson, and Eugene Belilovsky. Few-shot single-view 3-d object reconstruction with compositional priors. In *European Conference on Computer Vision*, pages 614–630. Springer, 2020.

[65] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.

[66] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[67] Iqbal Muhammad and Zhu Yan. Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3), 2015.

[68] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[69] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[70] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[71] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[72] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. *CoRR*, abs/1703.02921, 2017.

[73] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[74] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[75] Pedro O Pinheiro, Negar Rostamzadeh, and Sungjin Ahn. Domain-adaptive single-view 3d reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7638–7647, 2019.

[76] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22468–22478. Curran Associates, Inc., 2020.

[77] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[78] Tomas Sakinis, Fausto Milletari, Holger Roth, Panagiotis Korfiatis, Petro Kostandy, Kenneth Philbrick, Zeynettin Akkus, Ziyue Xu, Daguang Xu, and Bradley J Erickson. Interactive segmentation of medical images through fully convolutional neural networks. *arXiv preprint arXiv:1903.08205*, 2019.

[79] Roger N Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971.

[80] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking state-of-the-art deep learning software tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104. IEEE, 2016.

[81] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[82] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[83] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014.

[84] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018.

[85] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020.

[86] M Sornam, Kavitha Muthusubash, and V Vanitha. A survey on image classification and activity recognition using deep convolutional neural network architecture. In *2017 Ninth International Conference on Advanced Computing (ICoAC)*, pages 121–126. IEEE, 2017.

[87] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017.

[88] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018.

[89] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

[90] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[91] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[92] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[93] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.

[94] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[95] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer, 2017.

[96] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[97] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[98] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems*, pages 5236–5246, 2017.

[99] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[100] Bram Wallace and Bharath Hariharan. Few-shot generalization for single-image 3d reconstruction via priors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3818–3827, 2019.

[101] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[102] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[103] Mei Wang, Weihong Deng, Jiani Hu, Xunqiang Tao, and Yaohai Huang. Racial faces in the wild: Reducing racial bias by information maximization adaptation network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[104] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[105] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[106] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[107] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[108] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[109] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pages 540–550, 2017.

[110] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[111] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2020.

[112] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[113] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014.

[114] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[115] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[116] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *CoRR*, abs/1801.02612, 2018.

[117] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2690–2698, 2019.

[118] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502, 2019.

[119] Ming-Der Yang, Chih-Fan Chao, Kai-Siang Huang, Liang-You Lu, and Yi-Ping Chen. Image-based 3d scene reconstruction and exploration in augmented reality. *Automation in Construction*, 33:48–60, 2013.

[120] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017.

[121] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[122] Nida M Zaitoun and Musbah J Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015.

[123] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *CoRR*, abs/1710.11342, 2017.

[124] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *arXiv preprint arXiv:2103.02503*, 2021.

[125] Yichao Zhou, Shichen Liu, and Yi Ma. NeRD: Neural 3d reflection symmetry detector. In *CVPR*, 2021.

[126] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11. Springer, 2018.

[127] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

[128] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images “in the wild”. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5358–5367. IEEE, 2019.