Shape and Viewpoint without Keypoints

Shubham Goel, Angjoo Kanazawa, and Jitendra Malik

UC Berkeley

Abstract. We present a learning framework that learns to recover the 3D shape, pose and texture from a single image, trained on an image collection without any ground truth 3D shape, multi-view, camera viewpoints or keypoint supervision. We approach this highly under-constrained problem in a "analysis by synthesis" framework where the goal is to predict the likely shape, texture and camera viewpoint that could produce the image with various learned category-specific priors. Our particular contribution in this paper is a representation of the distribution over cameras, which we call "camera-multiplex". Instead of picking a point estimate, we maintain a set of camera hypotheses that are optimized during training to best explain the image given the current shape and texture. We call our approach Unsupervised Category-Specific Mesh Reconstruction (U-CMR), and present qualitative and quantitative results on CUB, Pascal 3D and new web-scraped datasets. We obtain state-of-the-art camera prediction results and show that we can learn to predict diverse shapes and textures across objects using an image collection without any keypoint annotations or 3D ground truth. Project page: https://shubham-goel.github.io/ucmr
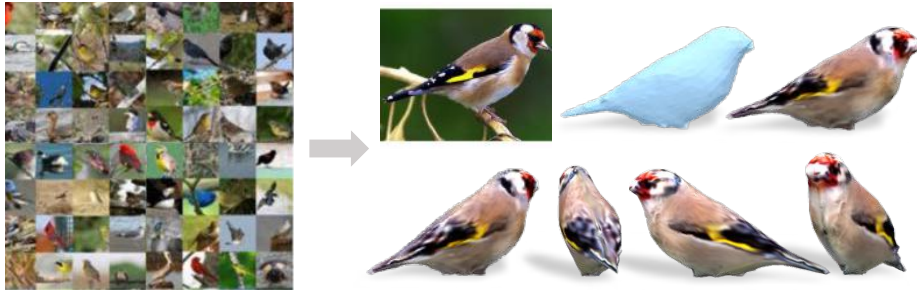
Fig. 1: Given an image collection of an object category, like birds, we propose a computational framework that given a single image of an object, predicts its 3D shape, viewpoint and texture, without using any 3D shape, viewpoints or keypoint supervision during training. On the right we show the input image and the results obtained by our method, shown from multiple views.

1 Introduction

There has been much progress in recent years in training deep networks to infer 3D shape from 2D images. These approaches fall into two major families

based on the supervisory signal used (a) 3D models, as available in collection of CAD models such as ShapeNet or (b) multiple views of the same object which permit deep learning counterparts of classical techniques such as shape carving or structure-from-motion. But do we need all this supervision?

It is easy to find on the internet large collections of images of objects belonging to particular categories, such as birds or cars or chairs. Let us focus on birds, for which datasets like CUB [35], shown in Figure 1 (left) exist. Note that this is a "Multiple Instance Single View" setting. For each bird instance we have only a single view, and every bird is a slightly different shape, even though the multiple instances share a family resemblance. Compared to classical SFM, where we have "Single Instance Multiple Views", our goal is to "3Dfy" these birds. From a single image, create a 3D model and its texture map, which can then be rendered from different camera viewpoints as shown in the rows of Figure 1 (right). This particular formulation was presented in the "Category-Specific Mesh Reconstruction" work of Kanazawa *et al.* [15], and their algorithm (CMR) is an inspiration for our work. Even earlier the work of Cashman and Fitzgibbon [3] working on analyzing images of dolphins showed how an "analysis by synthesis" paradigm with a deformable template model of a shape category could enable one to infer 3D shapes in an optimization framework.

It is under-appreciated that these approaches, while pioneering, do exploit some supervisory information. This includes (1) knowledge of a mean shape for the category (2) silhouettes for each instance (3) marked keypoints on the various instances (e.g. beak tip for each bird). Of these, the need for labeled keypoints is the most troublesome. Only one mean shape is needed for the entire category and sometimes a very generic initialization such as a sphere is good enough. Silhouettes could presumably be marked (perhaps in a category-agnostic way) by an instance segmentation network like Mask R-CNN. But keypoints are tedious to mark on every instance. This effort can be justified for a single important category like humans, but we cannot afford to do for the thousands of categories which we might want to "3Dfy". Yes, we could have the keypoints be marked by a network but to train that would require keypoint labels! There have been recent efforts in unsupervised keypoint-detection [13], however, so far, these methods learn viewpoint-dependent keypoints that often get mixed-up in the presence of 180-degree rotations.

In this paper we present an approach, U-CMR (for Unsupervised CMR) which enables us to train a function which can factor an image into a 3D shape, texture and camera, in a roughly similar setting to CMR [15], except that we replace the need for keypoint annotation with a single 3D template shape for the entire category. It turns out that keypoint annotations are needed for recovering cameras (using SFM like techniques) and if we don't have keypoint annotations, we have to solve for the camera simultaneously with shape and texture. Extending the "analysis by synthesis" paradigm to also recover cameras is unfortunately rather hard. Intuitively speaking, this is because of the discontinuous and multi-modal nature of the space of possible camera viewpoints. While shape and texture have a smooth, well behaved optimization surface that is amenable to gradient

descent optimization, this does not hold for the space of possible cameras. The two most likely camera explanations might lie on the opposite sides of the viewing sphere, where it is not possible to approach the optimal camera in an iterative manner as done when optimizing the energy landscape of a deep network. This typically causes the camera prediction to be stuck in bad local minima. Our solution to this problem is to maintain a set of possible camera hypotheses for each training instance that we call a **camera-multiplex**. This is reminiscent of particle filtering approaches with the idea being to maintain a distribution rather than prematurely pick a point estimate. We can iteratively refine the shape, texture as well as the camera-multiplex. More details are presented in Section 3.

We evaluate our approach on 3D shape, pose, and texture reconstruction on 4 categories: CUB-200-2011 birds [35], PASCAL-3D [38] cars, motorcycles and a new dataset of shoes we scraped from the Internet. We show that naively predicting shape, camera, and texture directly results in a degenerate solution where the shape is flat and the camera collapses to a single mode. We quantitatively evaluate the final camera prediction where the proposed camera-multiplex approach obtains the state of the art results under this weakly-supervised setting of no keypoint annotations. We show that despite the lack of viewpoints and keypoints, we can learn a reasonable 3D shape space, approaching that of the shapes obtained by a previous method that uses keypoint supervision [15].

2 Related Work

Recent deep learning based 3D reconstruction methods can be categorized by the required supervisory signal and the output of the system. This is illustrated in Table 1. Earlier methods formulate the problem assuming full 3D shape supervision for an image [5,9,7,31,10], which is enabled by synthetic datasets such as ShapeNet [37] and SunCG [27]. Some approaches generate synthetic datasets using data gathered from the real world to train their models [4,33,41]. However, requiring 3D supervision severely restricts these approaches, since ground truth 3D shape is costly or not possible to acquire, especially at a large scale. As such, follow up methods explore more natural forms of supervision, where multiple-views of the same object are available. Some of these approaches assume known viewpoints [39,16,32] akin to the setting of traditional MVS or visual hull. Other approaches explore the problem with unknown viewpoint setting [30,12,8]. These approaches assume that multi-view silhouettes, images, and/or depth images are available, and train their models such that the predicted 3D shapes reconstruct the images after projection or differentiable rendering. A variety of differentiable rendering mechanisms have been explored [21,18,20].

While multi-view images may be obtained in the real world, the vast amount of available visual data corresponds to the setting of unconstrained collection of single-view images, where no simultaneous multiple views of the same instance are available. This is also the natural setting for non-rigid objects where the shape may change over time. The traditional non-rigid structure from motion [29] also falls under this category, where the input is a tracked set of corresponding

Approach	Required Supervision per Image					Output			
	3D Shape	Multi-view	Cam	Keypoints	Mask	3D Shape	2.5D	Cam	Texture
MeshRCNN* [10]	✗					✓			
DeepSDF [23]	✗					✓			
Smalst [41]	✗		✗	✗	✗	✓		✓	✓
PTN [39]		✗	✗				✓		
MVC [30]		✗				✓	✓		
CMR [15]			✗	✗	✗	✓		✓	✓
CSM [19]					✗		✓		
Wu *et al.* [36]							✓	✓	✓
U-CMR					✗	✓		✓	✓

Table 1: A comparison of different approaches highlighting the differences between the input (during training) and the output (during inference). Our approach (U-CMR) uses only silhouette supervision but predicts full 3D shape, camera and texture. *MeshRCNN predicts shape in camera-coordinates instead of a canonical frame.

points [29,6] or 2D keypoints [34,22]. Earlier approaches fit a deformable 3D model [1,17,3,14] to 2D keypoints and silhouettes. Kanazawa *et al.* [15] propose CMR, a learning based framework where 3D shape, texture, and camera pose are predicted from a single image, trained under this setting of single-view image collections with known mask and keypoint annotations. While this is a step in the right direction, the requirement of keypoint annotation is still restrictive. More recently, Kulkarni *et al.* [19] bypass this requirement of keypoints to learn a dense canonical surface mapping of objects from a set of image collections with mask supervision and a template 3D shape. They focus on predicting the surface correspondences on images and learn to predict the camera viewpoints during the training, but do not learn to predict the 3D shape. While we tackle a different problem, we operate under the same required supervision. As such we quantitatively compare with CSM on the quality of the camera predicted, where our approach obtains considerably better camera predictions. Note that there are several recent approaches that explore disentangling images into 2.5D surface properties, camera, and texture of the visible regions from a collection of monocular images, without any masks [28,26,36]. However these approaches are mainly demonstrated on faces. In this work we recover a full 3D representation and texture from a single image.

3   Approach

3.1   Preliminaries

Shape Representation. We represent 3D shape as a mesh $M \equiv (V, F)$ with vertices $V \in \mathbb{R}^{|V| \times 3}$ and faces F. The set of faces F defines the connectivity of vertices in the mesh and we assume it remains fixed. We choose a mesh topology that is

Training with camera-multiplex

Probability of optimal camera

Training feed-forward camera

I

V

I^{uv}

Differentiable Renderer

$\tilde{I}_1$

$\tilde{I}_2$

$\tilde{I}_K$

f

g → π_i^*

I

Test time

V

f

g

I^{uv}

I

π
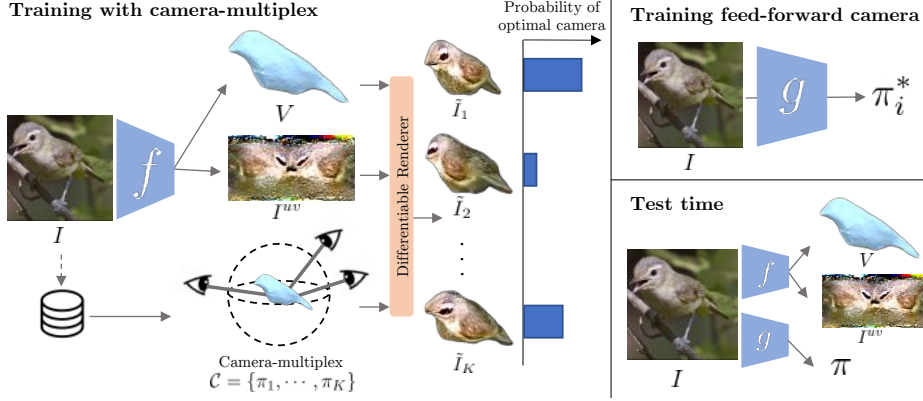
Camera-multiplex
$\mathcal{C} = \{\pi_1, \cdots, \pi_K\}$

Fig. 2: **Overview of the proposed framework.** We first train a shape and texture predictor f, while simultaneously optimizing the camera-multiplex $\mathcal{C}$, a set of K possible camera hypotheses maintained for every image. We render the predicted shape and texture from every camera in the multiplex, compute a per-camera reconstruction loss, treat it as negative log-likelihood of the camera, and update the f against the expected loss. We also update each camera in the multiplex against the loss incurred by it. After training f, we train a feed-forward model g to predict the best camera π^* in the multiplex from an image. As such, at test time our approach is able to predict all shape, texture, and camera from a single image.

homeomorphic to a sphere. We model the vertex positions of a deformable object as $V = \Delta_V + \bar{V}$, the summation of an instance-specific deformation Δ_V that is predicted from an image to a learned instance-independent mean shape $\bar{V}$ [15]. We initialize the mean shape with the template 3D mesh. This parameterization allows the model to learn the space of possible deformations for each category.

Texture Representation. As the topology of our mesh is fixed, we can use a UV image I^{uv} representation to model the texture. The values in a UV image get mapped onto the surface via a fixed UV mapping. The UV mapping is either a spherical projection akin to unrolling a globe into a flat map [11], or when that is not good enough, is a distortion-minimizing unwrap of a template mesh along manually defined seams computed using blender [2].

Camera Projection. We assume a weak-perspective camera projection, parametrized by scale $\mathbf{s} \in \mathbb{R}$, translation $\mathbf{t} \in \mathbb{R}^2$ and rotation $\mathbf{R}$ (captured as Euler angles azimuth, elevation, cyclo-rotation $[az,el,cr] \in \mathbb{R}^3$). We use $\pi(P)$ to denote the projection of a set of 3D points P onto the image coordinates via the weak-perspective projection defined by $\pi \equiv (s, \mathbf{t}, \mathbf{R})$. We denote the image rendered by composing all three factors as $\tilde{I} = \mathcal{R}(V, I^{uv}, \pi)$ and silhouette rendered just from the shape and camera as $\tilde{S} = \mathcal{R}(V, \pi)$ where $\mathcal{R}(\cdot)$ is a differentiable renderer. We denote a set of camera hypotheses kept for each image, a camera-multiplex $\mathcal{C} = \{\pi_1, \cdots, \pi_K\}$. We describe its training details below.

3.2   Our Method

Figure 2 shows an overview of our approach. During training, we learn a function $f(I)$ to predict the 3D shape and texture of the object underlying image I. We optimize over the camera-multiplex for each instance in the training dataset instead of making a deterministic prediction. For every shape and texture prediction, we compute the loss from every camera in the camera-multiplex, which induces a distribution on the camera poses inside a multiplex. We then use the expected loss over the camera-multiplex to update $f(I)$. When the training of $f(\cdot)$ converges, we identify the optimal camera for each training example in the camera-multiplex. We then train a function $g(I)$ that predicts the optimal camera from a single image, such that at test time we can infer all shape, texture, and camera from a single image. We provide the details for the training process below.

For each training instance I, let $\mathcal{C} = \{\pi_1, \cdots, \pi_K\}$ denote its camera-multiplex with K cameras and S its silhouette. Note that while we omit the subscript on training instances for brevity, *every* instance maintains its own $\mathcal{C}$ independently. For every predicted shape $V = \bar{V} + \Delta V$ and texture I^{uv}, we compute the silhouette and image reconstruction loss against each camera π_k:

$$L_{\mathrm{mask},k} = ||S - \tilde{S}_k||_2^2 + \mathtt{dt}(S) * \tilde{S}_k, \tag{1}$$

$$L_{\mathrm{pixel},k} = \mathtt{dist}(\tilde{I}_k \odot S, I \odot S,), \tag{2}$$

$L_{\mathrm{mask},k}$ is the silhouette loss where $\tilde{S}_k = \mathcal{R}(V, \pi_k)$ is the silhouette rendered from camera π_k, and $\mathtt{dt}(S)$ is the uni-directional distance transform of the ground truth silhouette. $L_{\mathrm{pixel},k}$ is the image reconstruction loss computed over the foreground regions where $\mathcal{R}(V, I^{uv}, \pi_k)$ is the rendered image from camera π_k. For this, we use the perceptual distance metric of Zhang *et al.* [40]. To exploit the bilateral symmetry and to ensure symmetric texture prediction, we also render the mesh under a bilaterally symmetric second camera, and compute $L_{\mathrm{pixel},k}$ as the average pixel loss from the two cameras.

In addition to these losses, we employ a graph-laplacian smoothness prior on our shape $L_{\mathrm{lap}} = ||V_i - \frac{1}{|N(i)|} \sum_{j \in N(i)} V_j||^2$ that penalizes vertices i that are far away from the centroid of their adjacent vertices $N(i)$. For cars, motorcycles and shoes, we empirically observe better results using $L_{\mathrm{lap}} = ||LV||_2$ where L is the discrete Laplace-Beltrami operator that minimizes mean curvature [25]. For this, we construct L once using the template mesh at the start of training. Following [1,3,17], we also find it beneficial to regularize the deformations as it discourages arbitrarily large deformations and helps learn a meaningful mean shape. The corresponding energy term is expressed as $L_{\mathrm{def}} = ||\Delta_V||_2$.

Model Update. For iteratively refining the camera-multiplex, we use the summation of the silhouette and image reconstruction loss $L_{\pi_k} = L_{\mathrm{mask},k} + L_{\mathrm{pixel},k}$ as the loss for each camera π_k in the camera-multiplex. We optimize each camera to minimize L_k every time the training instance is encountered during the training. For updating the shape and the texture, we use the resulting losses over

the cameras as a distribution over the most likely camera pose in the camera-multiplex, and minimize the expected loss over all the cameras. Specifically, we compute the probability of π_k being the optimal camera through a softmin function $p_k = \frac{e^{-L_k/\sigma}}{\sum_j e^{-L_j/\sigma}}$ and train the shape and texture prediction modules with the final loss:

$$L_{\text{total}} = \sum_k p_k(L_{\text{mask},k} + L_{\text{pixel},k}) + L_{\text{def}} + L_{\text{lap}}. \qquad (3)$$

In practice, the temperature σ changes dynamically while computing p_k by linearly normalizing L_k to have a fixed range to standardize the peakiness of the probability distribution. We do not backpropagate through p_k. In summary, we iteratively refine the cameras in the multiplex against loss L_{π_k} and update the parameters of f through $L_{\texttt{total}}$ for every training sample.

We implement the camera multiplex for each image $\mathcal{C}_i$ as a variable stored in a dictionary. Every time an image is encountered during training, the corresponding camera multiplex is fetched from this dictionary of variables and used as if it were an input to the rest of the training pipeline. Most modern deep learning frameworks such as PyTorch [24] support having such a dictionary of variables.

Training a feed-forward camera predictor. When the training of f converges, for each training image we select the optimal camera to be the camera that minimizes the silhouette and image reconstruction losses. We then train a new camera prediction module $g(I)$ in a supervised manner such that at inference time our model can predict all 3D shape, texture, and camera at the same time.

Approach at test time. Given a novel image I at test time, we can use the learnt modules f and g to predict the 3D shape, texture and camera-viewpoint of the object underlying image I. $f(I)$ predicts shape $V = \bar{V} + \Delta_V$ and texture I^{uv} while $g(I)$ predicts the camera-viewpoint π. This is illustrated in Figure 2.

4   Experiments

In this section we provide quantitative and qualitative evaluation of our approach that learns to predict 3D shape, texture, and camera from a single image without using any keypoint annotations during training. We explore our approach on four object categories: birds, cars, motorcycles and shoes.

4.1   Experimental Detail

Datasets. We primarily use the CUB-200-2011 dataset [35], which has 6000 training and test images of 200 species of birds. In addition to this, we train and evaluate U-CMR on multiple categories: car, motorcycles from the Pascal3D+ dataset and shoes scraped from zappos.com. For CUB and Pascal3D, we use the

same train-test splits as CMR [15]. For the initial meshes for birds and cars, we use the 3D template meshes used by Kulkarni *et al.* [19]. For others, we download freely available online models, homogenize to a sphere and simplify to reduce the number of vertices. We symmetrize all meshes to exploit bilateral symmetry. We compute masks for the zappos shoes dataset, which contains white background images, via simple threshold-based background subtraction and hole-filling.

Architecture. For all but texture, we use the same architecture as that of CMR [15] and pass Resnet18 features into two modules - one each for predicting shape and texture. The shape prediction module is a set of 2 fully connected layers with $\mathbb{R}^{3|V|}$ outputs that are reshaped into Δ_V following [15]. For the texture prediction, prior work predicted flow, where the final output is an offset that indicates where to sample pixels from. In this work we directly predict the pixel values of the UV image through a decoder. The texture head is a set of upconvolutional layers that takes the output of Resnet18 preserving the spatial dimensions. We find that this results in a more stable camera, as the decoder network is able to learn a spatial prior over the UV image. We use SoftRas [20] as our renderer. Please see the supplementary material for details and ablation studies.

Camera-multiplex implementation. We use $K = 40$ for camera-multiplex. We initialize the camera multiplex $\mathcal{C}$ for every image in the training set, to a set of K points whose azimuth and elevation are uniformly spaced on the viewing sphere. For cars and motorcycles, we use $K = 8$ cameras - all initialized to zero elevation. We optimize each camera in the multiplex using the silhouette loss $L_{mask,k}$ before training shape and texture. To reduce compute time while training shape and texture, we reduce K from 40 to 4 after 20 epochs by pruning the camera-multiplex and keeping the top 4 cameras. Note that naive data augmentation that scales and/or translates the image without adjusting the camera-multiplex will result in the rendered shape being pixel-unaligned. We handle random crop and scale data augmentation during training by adjusting the scale and translation in the stored camera multiplex with a deterministic affine transformation before using it for rendering the shape.

Baselines. As no other approach predicts 3D shape, texture and pose without relying on keypoints or known camera or multi-view cues during training, as baseline we compare with ablations of our approach that do not use the camera-multiplex. This can be thought of CMR without keypoints, which simultaneously predicts shape, camera and texture and only supervises rendered silhouette and texture. We call this approach CMR-nokp and ensure that the experimental setup is comparable to U-CMR. Additionally, in the supplementary, we compare to two variants of CMR [15] that have more supervision than our setting. For camera prediction, we compare with CMR [15] (which uses additional keypoint supervision), CSM [19] and U-CMR without texture prediction (U-CMR-noTex).

4.2   Qualitative Evaluation

The problem when no keypoints and viewpoints are available is that there always exists a planar shape and texture that explains the image and silhouette for any arbitrary camera pose. We first demonstrate this point using CMR-nokp. We observe that, as expected, CMR-nokp results in a degenerate solution shown in Figure 3 where the recovered shape explains the image silhouette well, but when seen from a different viewpoint the shape is planar.

In Figure 6, we visualize U-CMR predictions on unseen images from the CUB test set. Our approach, despite not using any viewpoint or keypoint supervision is able to recover a full, plausible 3D shape of the birds and learns to predict their texture from a single image. Our approach captures various types of bird shapes, including shapes of water birds and songbirds. We are able to recover sharp long tails and some protrusion of legs and beaks. Please see supplementary for more results of random samples from test set and comparisons to CMR.

Fig. 3: **CMR without keypoints.** CMR-nokp, which directly predicts shape, texture, and camera without keypoint supervision or the proposed camera-multiplex, obtains degenerate solutions. The shape from predicted camera viewpoint (centre) explains the silhouette well but an alternate view (right) reveals that the model has learned to output a planar, flat bird shape.

We further analyze the shape space that we learn in Figure 4, where we run principal component analysis on all the shapes obtained on the train set. We find directions that capture changes in the body type, the head shapes, and the tail shapes. In Figure 4, we also show that the final mean shape deviates significantly from the template mesh it was initialized to, by becoming thinner and developing a more prominent tail. Please see the supplemental for more results.

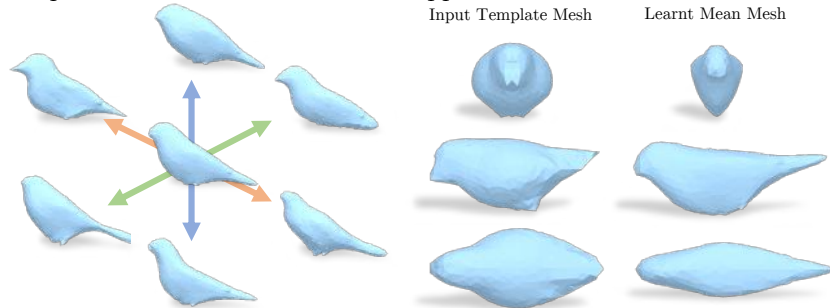Input Template Mesh   Learnt Mean Mesh

Fig. 4: **Learned Shape.** On the left, we visualize the space of learned shapes by running PCA. We see that the model has learned to output changes in body type, head shape, and tail types. On the right, we compare the template shape to the final learnt mean mesh. See text for discussion.

	Rotation Error ↓	Entropy (nats) ↑	Wasserstein Dist ↓ Azimuth	Elevation
GT	-	7.44	-	-
CMR [15]	22.94 °	7.25	6.03 °	4.33 °
CMR-nokp	87.52 °	5.73	64.66 °	12.39 °
CSM [19]	61.93 °	5.83	27.34 °	11.28 °
U-CMR (noTex)	61.82 °	**7.36**	16.08 °	7.90 °
U-CMR	**45.52** °	7.26	**8.66** °	**6.50** °

Table 2: **Quantitative evaluation of camera pose predictions on the test dataset.** We plot rotation error as the geodesic distance from the ground-truth, the entropy of the azimuth-elevation distribution and the wasserstein distance of marginal Az/El w.r.t. ground-truth. U-CMR outperforms all methods in the absence of keypoints.
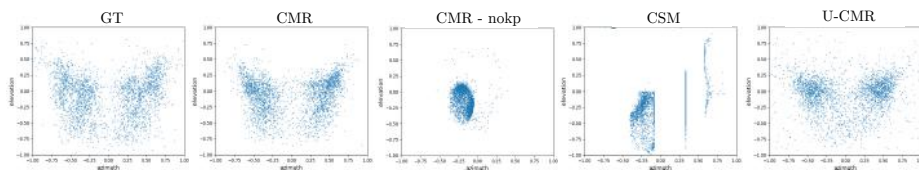
Fig. 5: **Camera Pose Distributions on CUB Test.** We show azimuth-elevation scatter plots over the entire CUB test set for different approaches. From left to right, we show (i) pseudo ground-truth cameras computed via running SfM on keypoints,(ii) the cameras predicted by CMR which uses the SfM cameras as supervision, (iii) CMR without viewpoint and keypoint supervision (CMR-nokp), (iv) CSM [19] and (v) our approach U-CMR. The last three approaches are weakly-supervised and do not use any keypoint annotation. Notice how the camera pose collapses in CMR-nokp and CSM, while U-CMR with camera-multiplex is able to obtain distribution similar to the ground truth cameras.

4.3   Quantitative Evaluation

We conduct quantitative evaluation on the camera poses obtained from our approach, since there are no 3D ground truth shapes on this dataset. For camera evaluation, we compare our approach to CSM [19], which learns to output dense correspondences of the image against a template 3D shape as well as the camera poses from image collections without keypoints. Note that they do not learn to predict 3D shapes. We used the same 3D template mesh as CSM and therefore are comparable to CSM. We evaluate cameras from different approaches on metrics measuring their accuracy and collapse. We compare our predicted cameras to the pseudo ground-truth cameras computed in CMR [15] using SFM on keypoints. For evaluating accuracy, we compute the rotation error $\text{err}_R = \arccos\left(\frac{\text{Tr}(\tilde{R}^T R^*) - 1}{2}\right)$ as the geodesic distance between the predicted camera rotation $\tilde{R}$ and the pseudo ground-truth camera rotation R^*. We report the average rotation error (in degrees) over the entire test dataset. To measure collapse, we analyze the azimuth-rotation distribution and report (i) its entropy (in nats) and (ii) it's

Wasserstein distance to the pseudo ground truth azimuth-elevation distribution. Because of computational ease, we only report the Wasserstein distance on the azimuth and elevation marginals. We primarily focus on azimuth and rotation because changes in scale, translation and cyclo-rotation of a camera only warp the image in 2D and don't constitute a "novel viewpoint".

Table 2 reports the numbers on all metrics for supervised (CMR) and weakly-supervised (CMR-nokp, CSM, U-CMR) methods. Observe that all the weakly-supervised baselines incur significant camera pose collapse - as can be seen by the entropy of their distributions. In contrast, U-CMR, despite being weakly-supervised, achieves an entropy that is slightly better than CMR - the supervised baseline. We automatically learn a camera distribution that is almost as close to the ground-truth distribution (in Wasserstein distance) as the supervised baseline (CMR). U-CMR is more accurate than CMR-nokp and CSM and achieves an average rotation error at least 15 degrees better than them. This table also suggests that the texture loss helps with refining camera poses to make them more accurate as U-CMR (noTex) is well-distributed with a very high entropy but is not as accurate as U-CMR.

Figure 5 visualizes the azimuth-elevation distributions of different approaches. This figure illustrates that while CSM prevents an extreme mode collapse of the camera, their camera pose distribution still collapses into certain modes. For making this figure, we employ CSM's public model, trained on the same CUB dataset with a fixed template shape. This empirical evidence exemplifies the fact that U-CMR's weakly-supervised camera-multiplex optimization approach learns a camera pose distribution that's much better than other weakly-supervised baselines and almost as good as supervised methods.

4.4   Evaluations on other categories

While our primary evaluation is on the CUB Birds dataset, we also run our approach on cars and motorcycles from the Pascal3D+ dataset, and the shoe images we scraped from zappos.com. We show qualitative visualizations of predicted shape, texture and camera for all categories. For Pascal3D cars, we also compare IoU of predicted shapes to CMR, previous deformable model fitting-based [17] and volumetric prediction [32] methods. All three of these approaches leverage segmentation masks and cameras and keypoints to learn 3D shape inference.

Figure 7 shows qualitative results on selected images from their respective test set. U-CMR learns accurate camera poses and diverse yet plausible shapes for cars and motorcycles. For shoes, U-CMR shapes are reasonable but not as diverse because of biases in the underlying dataset. We observe some artifacts where the sides of the cars have concave indentations and some parts of the shoes are tapered and pointy. These issues stem from using weak-perspective projections and limitations of the regularization, which is not spatially adaptive. Please see the supplemental for each category's PCA visualizations and the initial template.

We report the mean IoU on the test set in Table 3 and observe that U-CMR performs comparably to alternate methods that require more supervision.

Fig. 6: **Qualitative results.** For each input test image on the left, we show the predicted mesh, the textured mesh, and the textured mesh from multiple views.

4.5   Limitations

While U-CMR shows promising results in the direction of weakly supervised 3D shape understanding, it has some limitations. Foremost limitation is that we do
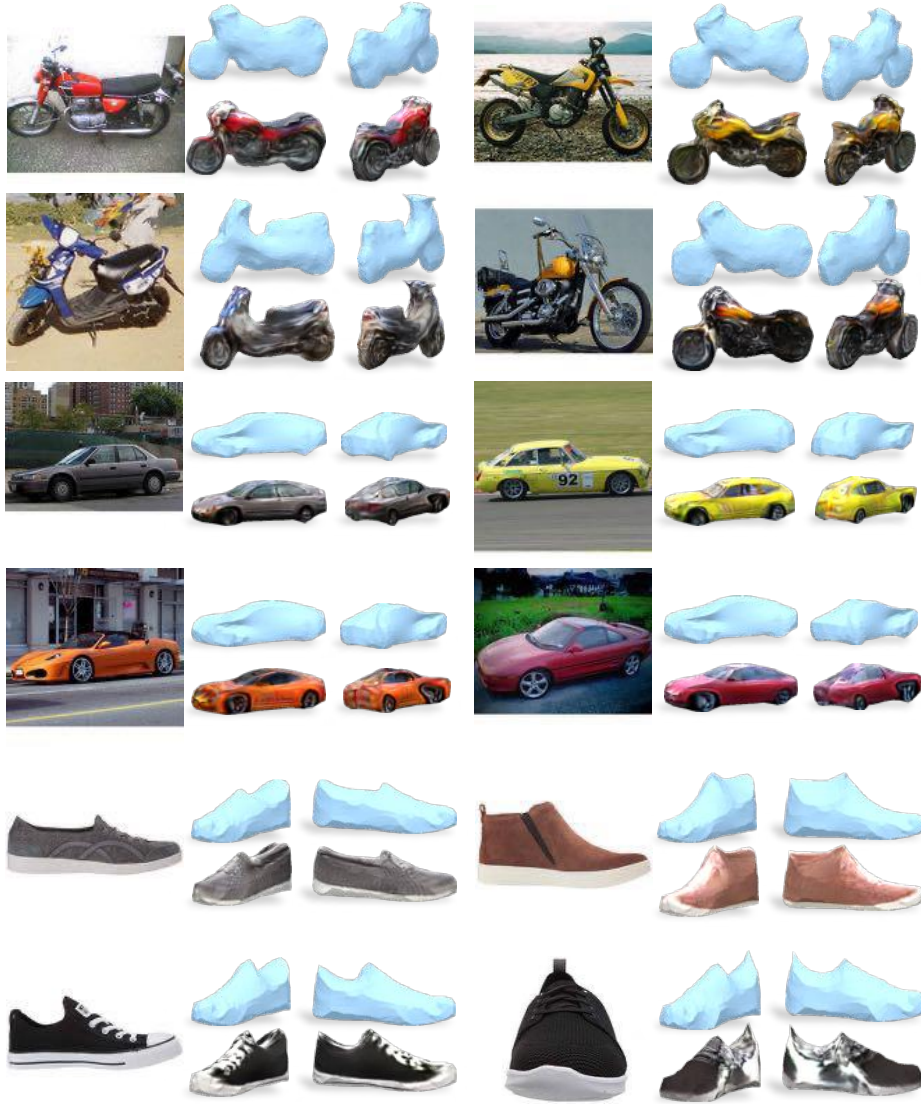
Fig. 7: **Qualitative results on cars, motorcycles and shoes.** For each image, we show the predicted 3D shape and texture from two viewpoints.

not model articulation and expect to fail in cases with significant articulation. In Figure 8, we demonstrate various modes of failure for shape, texture and camera-viewpoint prediction. Our approach struggles when the bird shape is significantly different from the template mesh and undergoes large articulation, such as the case with flying birds. It is challenging to identify correct camera poses when there's a large deformation like this without keypoints. The data

Category	CSDM [17]	DRC [32]	CMR [15]	U-CMR
Car	0.60	0.67	0.640	0.646

Table 3: **Reconstruction evaluation using PASCAL 3D+.** We report the mean intersection over union (IoU) on PASCAL 3D+ to benchmark the obtained 3D reconstructions (higher is better). We compare to CMR [15], a deformable model fitting approach (CSDM [17]) and a volumetric prediction approach (DRC [32]). CSDM and DRC use image collection supervision in addition to keypoints/cameras.

imbalance between flying and not flying birds also exacerbates this problem. The two examples in the top row of the figure show how our shape prediction fails when the bird in the image is flying, or has it's wings open. The example in the bottom right shows an articulated bird with it's head twisted back. Due to the lack of an articulation model, these failure cases are expected. We also fail at predicting good texture sometimes - especially for parts of the object that are not visible. The example on the bottom left of Figure 8 and bottom right of Figure 7 shows how background colours may leak into the predicted texture.

Fig. 8: **Failure Modes.** The columns, from left to right, show the input image, the predicted shape and texture from the predicted camera, and finally a different view of the textured mesh. See the text for discussion.

5   Conclusion

In this work, we present a learning framework that can decompose an image of a deformable object into its 3D shape, texture, and camera viewpoint. In order to solve this highly under-constrained problem, we propose a representation for maintaining a distribution over possible camera viewpoints called camera-multiplex. This allows the model to maintain a possible set of camera hypothesis, avoiding the learning process from getting stuck in a bad local minima. We show our approach on four categories, where we show that U-CMR can recover reasonable 3D shape and texture without viewpoints and keypoints.

References

1. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIG-GRAPH (1999)
2. Blender Online Community: Blender - a 3D modelling and rendering package. Blender Institute, Amsterdam (2019), http://www.blender.org
3. Cashman, T.J., Fitzgibbon, A.W.: What shape are dolphins? building 3D morphable models from 2D images. TPAMI (2013)
4. Chen, W., Wang, H., Li, Y., Su, H., Wang, Z., Tu, C., Lischinski, D., Cohen-Or, D., Chen, B.: Synthesizing training images for boosting human 3d pose estimation. In: 3DV (2016)
5. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016)
6. Dai, Y., Li, H., He, M.: A simple prior-free method for non-rigid structure-from-motion factorization. IJCV (2014)
7. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)
8. Gadelha, M., Maji, S., Wang, R.: 3d shape induction from 2d views of multiple objects. In: 3DV (2017)
9. Girdhar, R., Fouhey, D., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: ECCV (2016)
10. Gkioxari, G., Malik, J., Johnson, J.: Mesh r-cnn. In: ICCV (2019)
11. Hughes, J.F., Foley, J.D.: Computer graphics: principles and practice. Pearson Education (2014)
12. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. In: NeurIPS (2018)
13. Jakab, T., Gupta, A., Bilen, H., Vedaldi, A.: Unsupervised learning of object landmarks through conditional image generation. In: NeurIPS (2018)
14. Kanazawa, A., Kovalsky, S., Basri, R., Jacobs, D.: Learning 3d deformation of animals from 2d images. In: Computer Graphics Forum. Wiley Online Library (2016)
15. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: ECCV (2018)
16. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. In: NeurIPS (2017)
17. Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-specific object reconstruction from a single image. In: CVPR (2015)
18. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018)
19. Kulkarni, N., Gupta, A., Tulsiani, S.: Canonical surface mapping via geometric cycle consistency. In: ICCV (2019)
20. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: ICCV (2019)
21. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer. In: ECCV (2014)
22. Novotny, D., Ravi, N., Graham, B., Neverova, N., Vedaldi, A.: C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In: ICCV (2019)
23. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)

24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
25. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental mathematics (1993)
26. Shu, Z., Sahasrabudhe, M., Alp Guler, R., Samaras, D., Paragios, N., Kokkinos, I.: Deforming autoencoders: Unsupervised disentangling of shape and appearance. In: ECCV (2018)
27. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: CVPR (2017)
28. Thewlis, J., Bilen, H., Vedaldi, A.: Unsupervised learning of object frames by dense equivariant image labelling. In: NeurIPS (2017)
29. Torresani, L., Hertzmann, A., Bregler, C.: Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. TPAMI (2008)
30. Tulsiani, S., Efros, A.A., Malik, J.: Multi-view consistency as supervisory signal for learning shape and pose prediction. In: CVPR (2018)
31. Tulsiani, S., Gupta, S., Fouhey, D., Efros, A.A., Malik, J.: Factoring shape, pose, and layout from the 2d image of a 3d scene. In: CVPR (2018)
32. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: CVPR (2017)
33. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: CVPR (2017)
34. Vicente, S., Carreira, J., Agapito, L., Batista, J.: Reconstructing pascal voc. In: CVPR (2014)
35. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
36. Wu, S., Rupprecht, C., Vedaldi, A.: Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. arXiv preprint arXiv:1911.11130 (2019)
37. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR (2015)
38. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: A benchmark for 3d object detection in the wild. In: WACV (2014)
39. Yan, X., Yang, J., Yumer, E., Guo, Y., Lee, H.: Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In: NeurIPS (2016)
40. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep networks as a perceptual metric. In: CVPR (2018)
41. Zuffi, S., Kanazawa, A., Berger-Wolf, T., Black, M.J.: Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In: ICCV (2019)

Appendix

Overview

In this supplementary material, we present additional results including visualization of the shape, texture and camera-multiplex, comparisons to CMR [15], qualitative results on random test samples, an ablation study on texture prediction model, and additional details on the network architecture.

6   More Results

PCA in Texture space In Figure 9, we visualize the learnt texture space by running PCA on predicted uv-textures across the entire test dataset. On the left, the mean texture is a rather dull gray colour, as expected. On the right, we visualize axes of variation. In the first column, we see low-frequency variations in overall colour, head and belly of the bird. In the second column, we see slightly higher-frequency variations that assign different colors to different parts (head, back, belly and wings) of the bird. We can even recognize eyes and beak in the last two rows on the right.
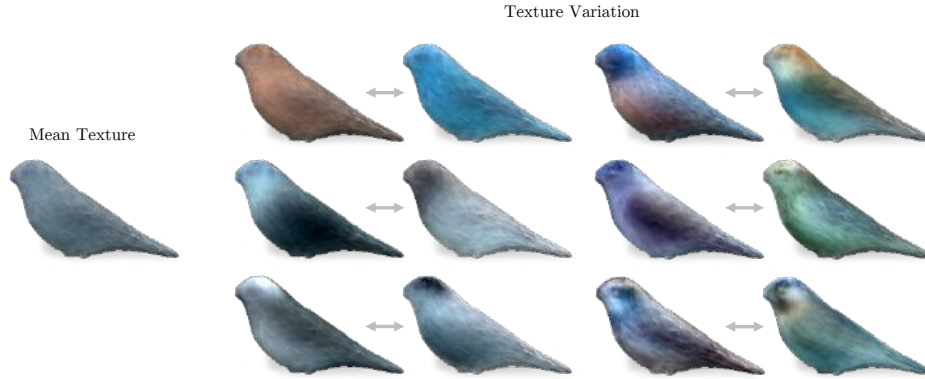
Fig. 9: **PCA in UV-Texture space.** We visualize the learnt texture space by running PCA over all predicted uv-texture maps on the test dataset, and rendering axes of variation on the learnt mean bird shape. See text for discussion.

Learnt shape space for other categories In Figure 10, we compare the input template mesh to the learnt shape space for car, motorbike and shoe categories. Observe, on the left, that the input template and learnt mean shape differ substantially for each of these categories - some more than others. For example, the front tire of the motorbike becomes more significantly more prominent, the back of the car becomes more rounded and the shoe becomes slimmer and more elongated. On the right, we visualize the space of learned shapes by running PCA on all shapes obtained on training and test dataset. The three PCA axis visualized show interesting deformations. For example, in the motorcycle, the three visualized axes vary in prominence of front tire, size of fuel-tank and concavity of seat respectively.
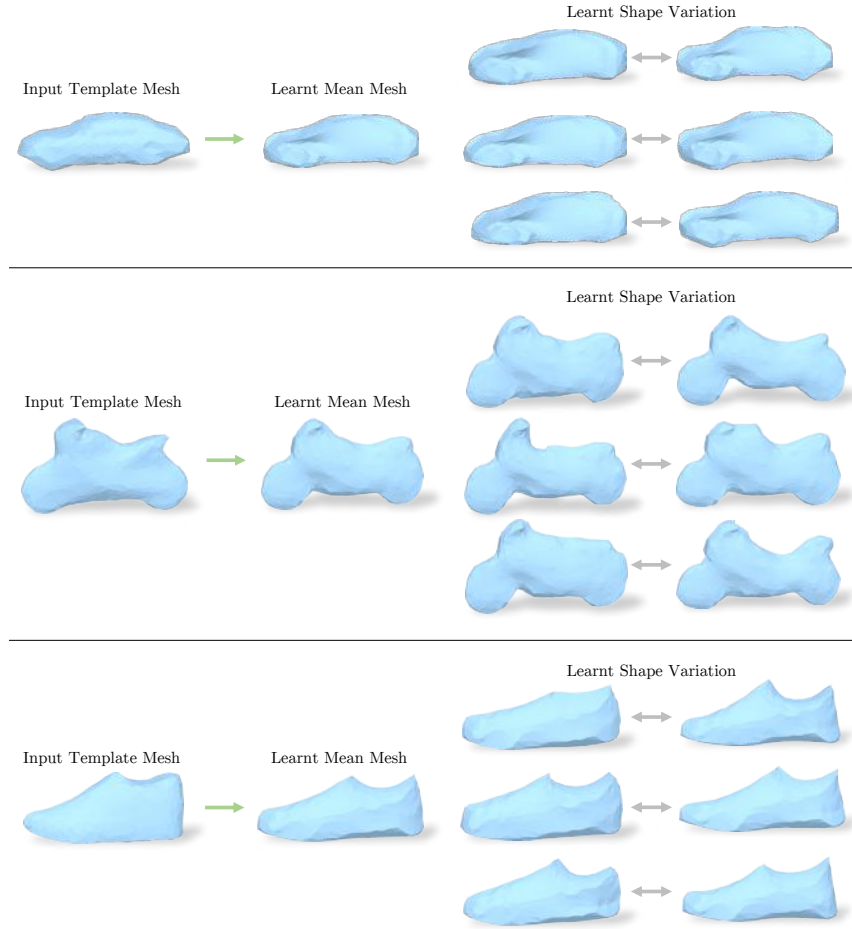
Fig. 10: **Learned Shape on other categories.** On the left, we compare the template shape to the final learnt mean mesh. On the right, we visualize the space of learned shapes by running PCA on all shapes and show three axis of deformations.

Camera-multiplex visualization over time. Figure 11 shows how the azimuth-elevation distribution of camera poses in the camera-multiplex (over the entire training dataset) changes as training progresses. Observe that the distribution changes rapidly initially and results in a final distribution that is very different from the initial distribution.

Qualitative comparison to CMR. We qualitatively compare results from U-CMR to 2 variants of CMR [15]. The first is the official CMR model (CMR-official) that was trained using additional keypoint losses and used NMR [18] as it's differentiable renderer. The second is our implementation of CMR (CMR-ours) which is similar to U-CMR in it's architecture for shape/texture, using Softras
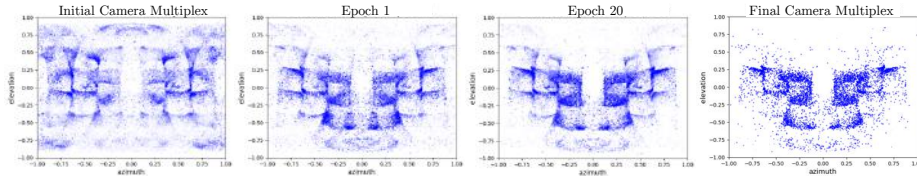
Fig. 11: **Change in Camera Pose Distributions during training.** We show azimuth-elevation scatter plots of $K = 40$ camera poses in all the camera-multiplex of the CUB train set as training progresses. Points corresponding to less probable cameras have a lower alpha value and are more transparent. Starting from top left, we have the camera poses after the camera-multiplex initialization, after 1 training epoch, after 20 training epochs and the final optimized camera poses. The number of camera poses in each multiplex (K) is pruned down from 40 to 4 after epoch 20.

[20] for rendering, regularizing shape using the graph-laplacian and having the same template mesh as it's initial mean shape, but different from U-CMR in that it uses the ground-truth camera pose from SFM during training. Unlike CMR-official, CMR-ours does not include vertex-keypoint reprojection loss.

In Figures 13-14, we compare CMR-official, CMR-ours and U-CMR. For each input image, the first row is from CMR-official, second is from CMR-ours and the last row is U-CMR. Observe that CMR-official is not as accurate as CMR-ours in capturing the shape and texture of the underlying bird but has pointier beaks and feet because of the keypoint reprojection loss it uses. The figures show that U-CMR shapes are qualitatively very similar to CMR-ours, hence exemplifying our assertion that U-CMR's camera-multiplex optimization alleviates the need for ground-truth camera pose supervision for most cases.

We compare U-CMR and CMR-ours on a random subset of 15 images from the test dataset in Figures 15-17. Observe that U-CMR (second row) accurately predicts shapes that are very similar to those from CMR-ours when the bird is not articulating too much.

Ablation on texture prediction model. We experiment with two different architectures for predicting the texture. First, we explore predicting texture as texture-flow, which is used in CMR. Texture-flow is a 2D positional offset for every pixel in the UV image that specifies where to sample the RGB values from the input image. Second, we predict the UV image values directly (Texture-gen) using a decoder attached to a bottleneck with spatial dimensions preserved. This is the final approach used in U-CMR. We observed that predicting a flow-field can lead to flat degenerate shapes and a collapse of optimized camera poses. Figure 12 shows the collapse in the final optimized camera poses when predicting texture as a flow-field. This is because even when the camera pose is wrong, texture-flow is able to learn to adjust to the bad camera, as the output of the texture-flow across different instances is not necessarily correlated. However, when predicting the texture directly through a decoder, the network learns an implicit spatial prior of the texture across the dataset. For example, the network

needs to learn to generate the texture of the eye at the same location in every texture map. Similarly for wings, breast, head etc; as the texture map is predicted in a canonical semantic space. This spatial prior that is learned through a spatial decoder allows the texture prediction to disambiguate incorrect and correct poses in the camera-multiplex.
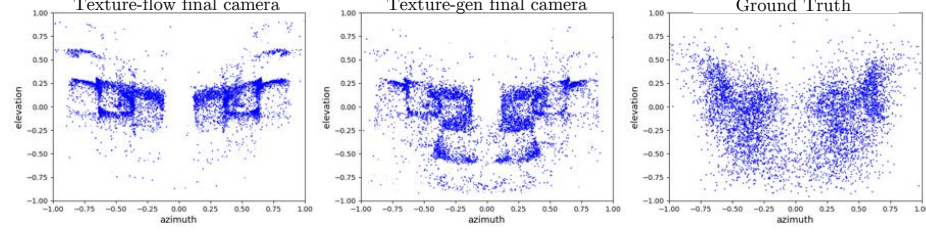

Fig. 12: **Texture-flow camera-multiplex distribution on training set.** On the left, we show the azimuth-elevation distribution of the final camera-multiplex when we predict texture as a flow-field for sampling from the input image. Note how the camera poses have collapsed to 2 broad areas. Center: U-CMR camera-multiplex, Right: GT camera distribution

7   Training details

7.1   Architecture details.

Our code is available on our project page: https://shubham-goel.github.io/ucmr. The shape and texture predictor f has an encoder-decoder architecture. Image $I \in \mathbb{R}^{256 \times 256 \times 3}$ is first encoded to latent feature map $z \in \mathbb{R}^{4 \times 4 \times 256}$ using Resnet-18. The shape head takes flattened $z \in \mathbb{R}^{16 \times 256}$ as input and passes it through 2 fully connected layers, each with 200 output channels and then a final linear layer for predicting $\Delta_V \in \mathbb{R}^{|V| \times 3}$. The texture head takes the latent feature map and bilinearly samples it to $z \in \mathbb{R}^{4 \times 8 \times 256}$, this is followed by 7 Resnet blocks with 256, 256, 256, 128, 64, 32, 16 output channels respectively, with intermediate bilinear upsampling by a factor of 2 after blocks 1, 3, 4, 5, 6. This is then sent to a final convolution layer that outputs the texture map $I^{uv} \in \mathbb{R}^{128 \times 256 \times 3}$. The Resnet encoder uses ReLU activations while the shape and texture heads use Leaky-ReLU activations. All networks use batchnorm for normalization. We will release our code upon publication.

After training the shape and texture prediction with camera-multiplex, we learn the feed-forward camera pose predictor g. We attach this as another head to the latent variable z from the shared Resnet-18 trained for shape and texture prediction with camera-multiplex. We freeze the encoder, and then train a fully connected head for predicting camera scale $s \in \mathbb{R}$, translation $t \in \mathbb{R}^2$ and rotation (as quaternion $q \in \mathbb{R}^4$) through 2 fully connected layer each with 200 channels.
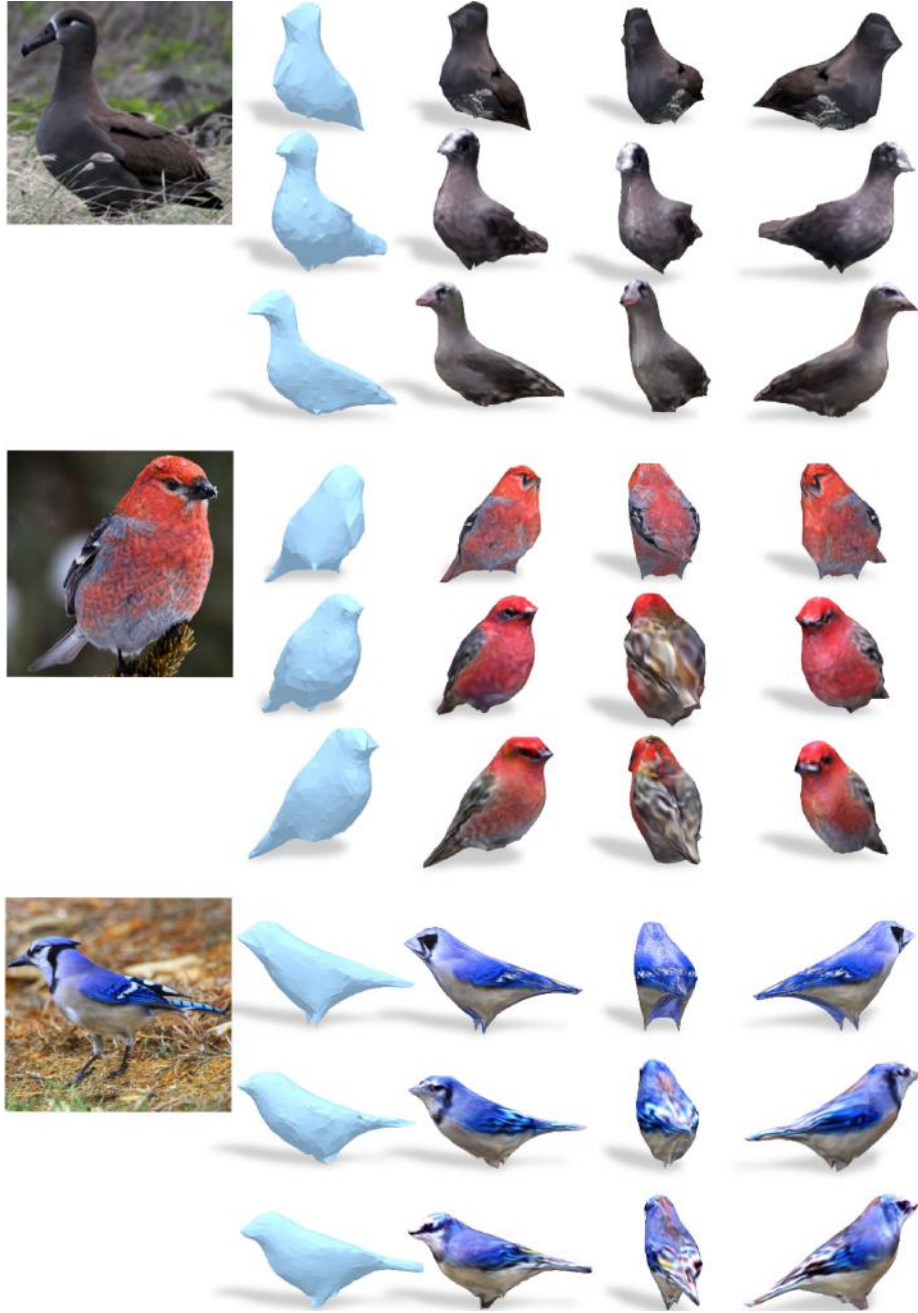
Fig. 13: **CMR-official vs CMR-ours vs U-CMR.** We compare U-CMR (third row) to CMR-ours (second row) and CMR-official (first row) on selected images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.
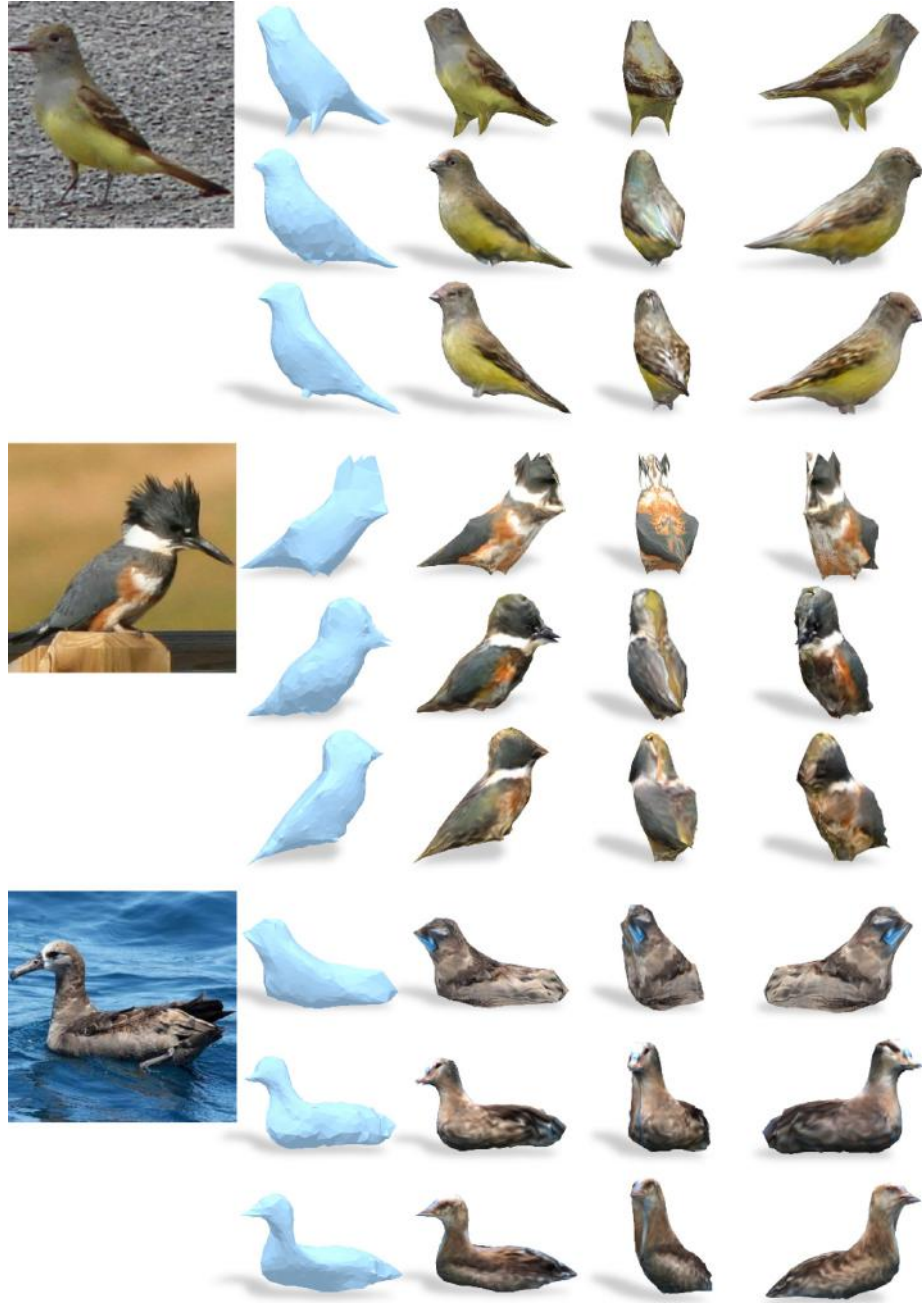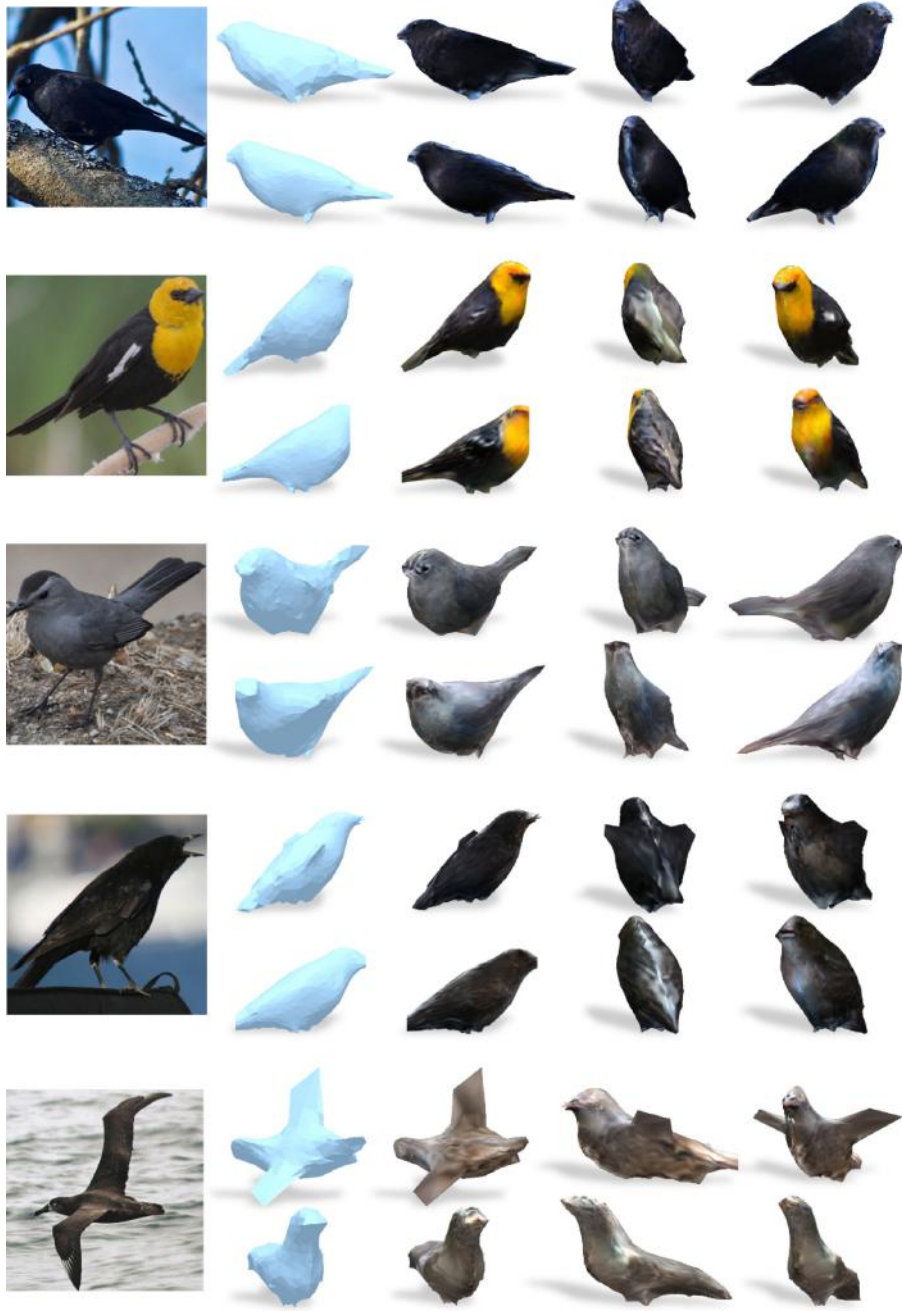
Fig. 14: **CMR-official vs CMR-ours vs U-CMR.** We compare U-CMR (third row) to CMR-ours (second row) and CMR-official (first row) on selected images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

Fig. 15: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the testset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.
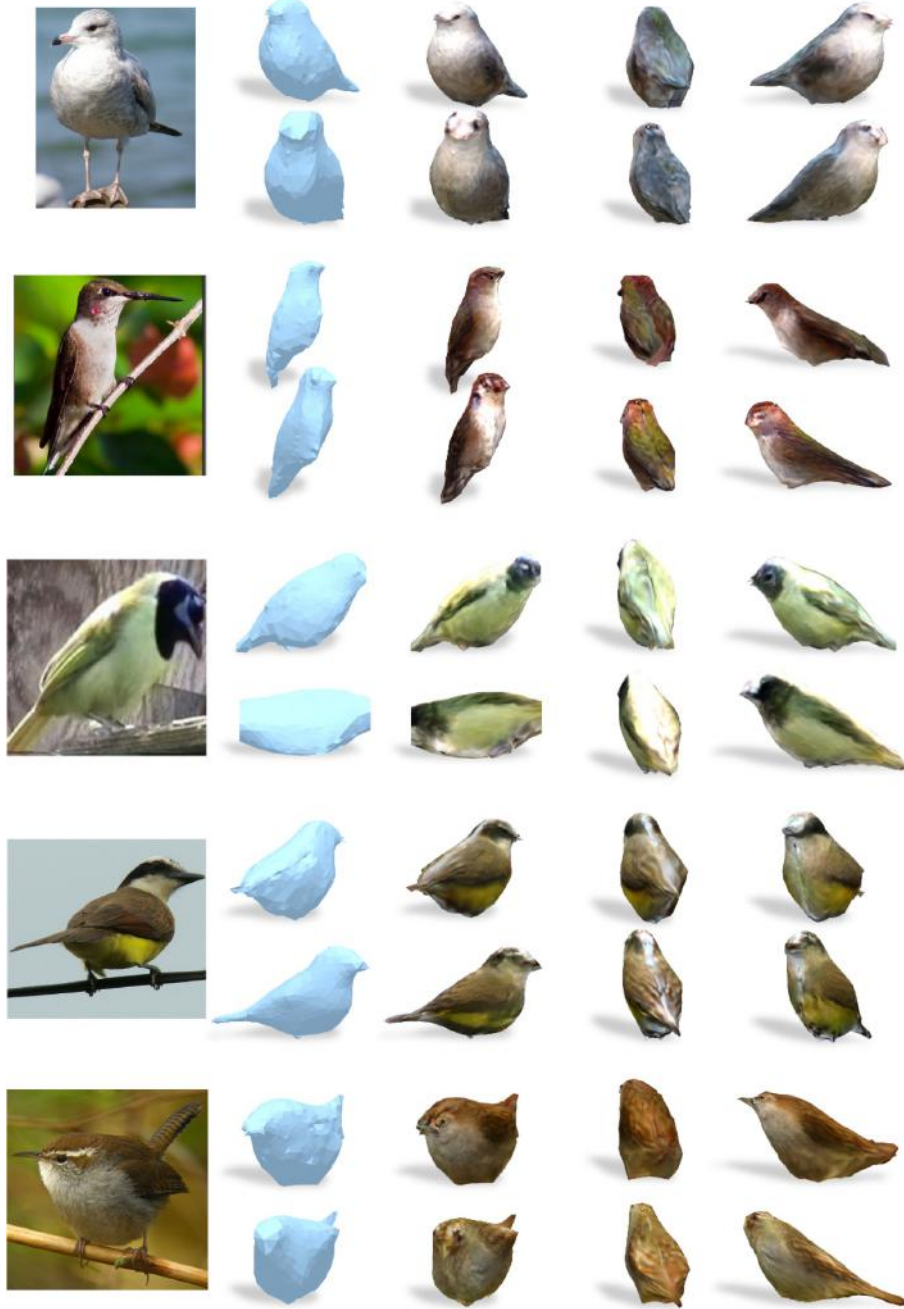
Fig. 16: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.
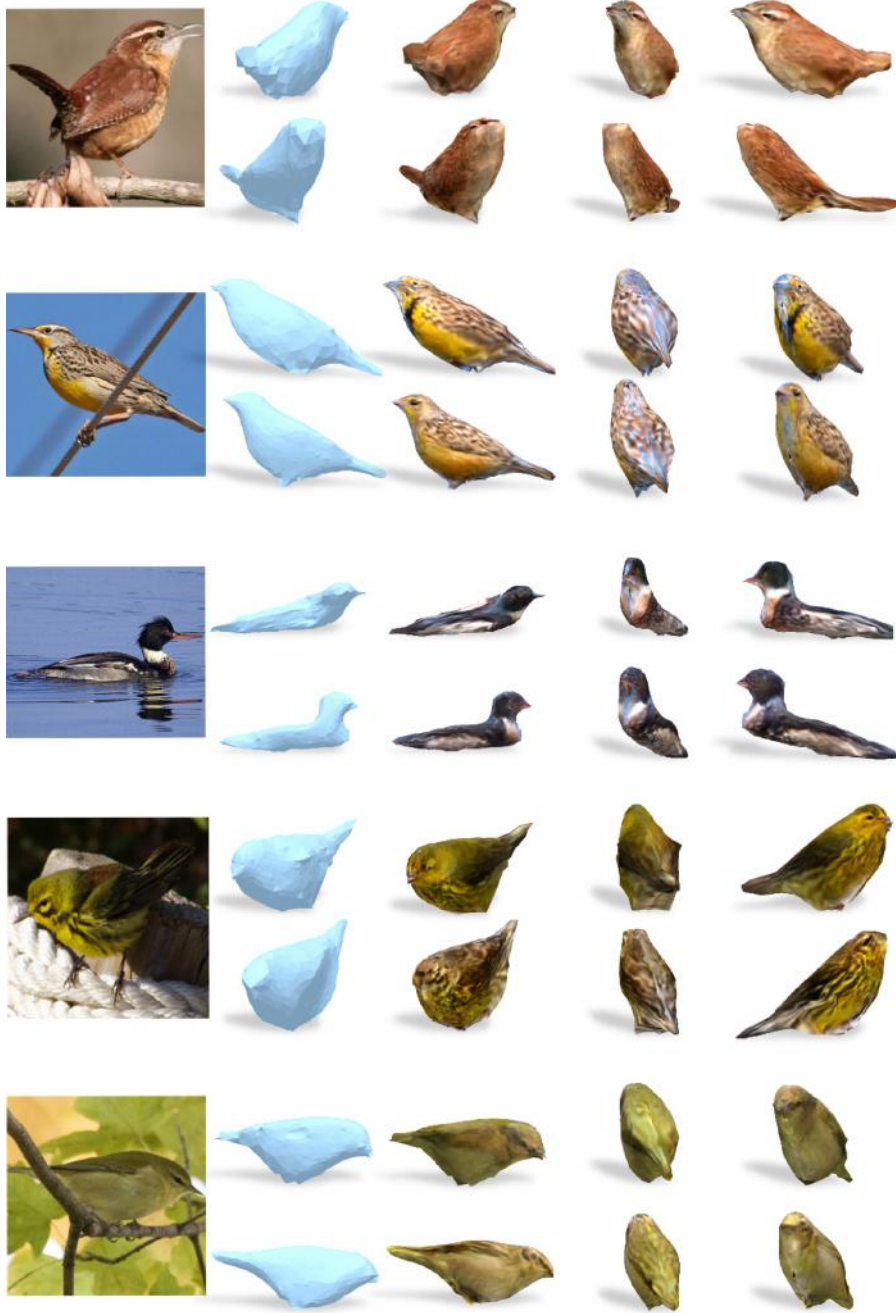
Fig. 17: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.