# Canonical 3D Deformer Maps: Unifying parametric and non-parametric methods for dense weakly-supervised category reconstruction

**David Novotny*** **Roman Shapovalov*** **Andrea Vedaldi**
Facebook AI Research
{dnovotny, romansh, vedaldi}@fb.com

## Abstract

We propose the *Canonical 3D Deformer Map*, a new representation of the 3D shape of common object categories that can be learned from a collection of 2D images of independent objects. Our method builds in a novel way on concepts from parametric deformation models, non-parametric 3D reconstruction, and canonical embeddings, combining their individual advantages. In particular, it learns to associate each image pixel with a deformation model of the corresponding 3D object point which is canonical, i.e. intrinsic to the identity of the point and shared across objects of the category. The result is a method that, given only sparse 2D supervision at training time, can, at test time, reconstruct the 3D shape and texture of objects from single views, while establishing meaningful dense correspondences between object instances. It also achieves state-of-the-art results in dense 3D reconstruction on public in-the-wild datasets of faces, cars, and birds.

## 1 Introduction

We address the problem of learning to reconstruct 3D objects from individual 2D images. While 3D reconstruction has been studied extensively since the beginning of computer vision research [49], and despite exciting progress in monocular reconstruction for objects such as humans, a solution to the general problem is still elusive. A key challenge is to develop a *representation* that can learn the 3D shapes of common objects such as cars, birds and humans from 2D images, without access to 3D ground truth, which is difficult to obtain in general. In order to do so, it is not enough to model individual 3D shapes; instead, the representation must also *relate the different shapes* obtained when the object deforms (e.g. due to articulation) or when different objects of the same type are considered (e.g. different birds). This requires establishing *dense correspondences* between different shapes, thus identifying equivalent points (e.g. the left eye in two birds). Only by doing so, in fact, the problem of reconstructing independent 3D shapes from 2D images, which is ill-posed, reduces to learning a single deformable shape, which is difficult but approachable.

In this paper, we introduce the *Canonical 3D Deformer Map* (C3DM), a representation that meets these requirements (Figure 1). C3DM combines the benefits of parametric and non-parametric representations of 3D objects. Conceptually, C3DM starts from a *parametric 3D shape model* of the object, as often used in Non-Rigid Structure From Motion (NR-SFM [12]). It usually takes the form of a *mesh* with 3D vertices $\mathbf{X}_1, \ldots, \mathbf{X}_K \in \mathbb{R}^3$ expressed as a linear function of global deformation parameters $\boldsymbol{\alpha}$, such that $\mathbf{X}_k = B_k\boldsymbol{\alpha}$ for a fixed operator $B_k$. Correspondences between shapes are captured by the identities $k$ of the vertices, which are invariant to deformations. Recent works such as Category-specific Mesh Reconstruction (CMR) [30] put this approach on deep-learning rails, learning to map an image $I$ to the deformation parameters $\boldsymbol{\alpha}(I)$. However, working with meshes

*Authors contributed equally.

Preprint. Under review.

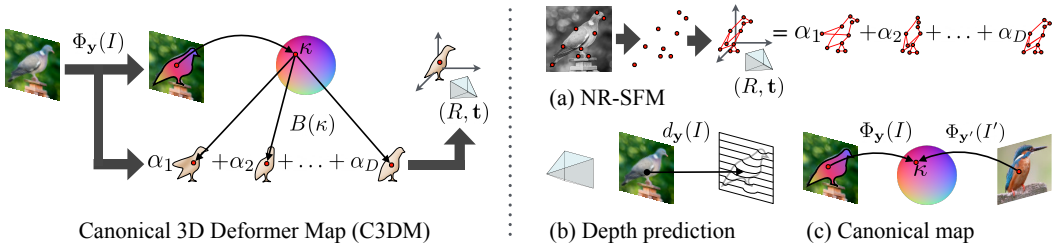


Figure 1: The C3DM representation (left) associates each pixel $\mathbf{y}$ of the image $I$ with a *deformation operator* $B(\kappa)$, a function of the object canonical coordinates $\kappa = \Phi_{\mathbf{y}}(I)$. C3DM then reconstructs the corresponding 3D point $\mathbf{X}$ as a function of the global object deformation $\boldsymbol{\alpha}$ and viewpoint $(R, \mathbf{t})$. It extends three ideas (right): (a) non-rigid structure from motion computes a *sparse* parametric reconstruction starting from 2D keypoints rather than an image; (b) a monocular depth predictor $d_{\mathbf{y}}(I)$ non-parametrically maps each pixel to its 3D reconstruction but lacks any notion of correspondence; (c) a canonical mapping $\Phi_{\mathbf{y}}(I)$ establishes dense correspondences but does not capture geometry.

causes a few significant challenges, including guaranteeing that the mesh does not fold, rendering the mesh onto the image for learning, and dealing with the finite mesh resolution. It is interesting to compare parametric approaches such as CMR to *non-parametric depth estimation models*, which directly map each pixel $\mathbf{y}$ to a depth value $d_{\mathbf{y}}(I)$ [70, 33, 19], describing the geometry of the scene in a dense manner. The depth estimator $d_{\mathbf{y}}(I)$ is easily implemented by means of a convolutional neural network and is not bound to a fixed mesh resolution. However, a depth estimator has no notion of correspondences and thus of object deformations.

Our intuition is that these two ways of representing geometry, parametric and non-parametric, can be combined by making use of the third notion, a *canonical map* [58, 51, 35]. A canonical map is a non-parametric model $\Phi_{\mathbf{y}}(I) = \kappa$ that associates each pixel $\mathbf{y}$ to the intrinsic coordinates $\kappa$ of the corresponding object point. The latter can be thought of as a continuous generalization of the index $k$ that in parametric models identifies a vertex of a mesh. Our insight is that *any* intrinsic quantity — i.e. one that depends only on the identity of the object point — can then be written as a function of $\kappa$. This includes the *3D deformation operator* $B_\kappa$, so that we can reconstruct the 3D point found at pixel $\mathbf{y}$ as $\mathbf{X}_{\mathbf{y}} = B_\kappa \boldsymbol{\alpha}$. Note that this also requires to learn the mapping $\kappa \mapsto B_\kappa$, which we can do by means of a small neural network.

We show that the resulting representation, C3DM, can reconstruct the shape of 3D objects densely and from single images, using only easily-obtainable 2D supervision at training time — the latter being particularly useful for 3D reconstruction from traditional non-video datasets. We extensively evaluate C3DM and compare it to CMR [30], state-of-the-art method for monocular category reconstruction. C3DM achieves both higher 3D reconstruction accuracy and more realistic visual reconstruction on real-world datasets of birds, human faces, and four other deformable categories of rigid objects.

## 2 Related work

The literature contains many impressive results on image-based 3D reconstruction. To appreciate our contribution, it is essential to characterize the assumptions behind each method, the input they require for training, and the output they produce. Multiple works [40, 6, 21, 53, 11, 38, 26, 71, 28, 47, 66, 56, 61, 45, 46, 29, 34, 53, 38, 50, 46, 62, 47] take as input an existing parametric 3D model of the deformable object such as SMPL [40] or SCAPE [6] for humans bodies, or Basel [48] for faces and *fit it to images*. In our case, no prior parametric 3D model is available; instead, our algorithm *simultaneously learns and fits a 3D model using only 2D data as input*.

**Sparse NR-SFM** methods receive sparse 2D keypoints as input and lift them in 3D, whereas C3DM receives as input an image and produces a *dense* reconstruction. In other words, we wish to obtain dense reconstruction of the objects although only sparse 2D annotations are still provided during training. For learning, NR-SFM methods need to separate the effect of viewpoint changes and deformations [69]. They acheive it by constraining the space of deformations in one of the following ways: assume that shapes span a low-rank subspace [3, 18, 17, 76] or that 3D trajectories are smooth in time [4, 5], or combine both types of constraints [1, 20, 37, 36], or use multiple subspaces [76, 2], sparsity [73, 74] or Gaussian priors [60]. In Section 3.2, we use NR-SFM to define one of the loss

functions. We chose to use the recent C3DPO method [44], which achieves that separation by training a canonicalization network, due to its state-of-the-art performance.

**Dense 3D reconstruction.** Differently from our work, most of the existing approaches to dense 3D reconstruction assume either 3D supervision or rigid objects and multiple views. Traditional *multi-view* approaches [7] perform 3D reconstruction by analyzing disparities between two or more calibrated views of a rigid object (or a non-rigid object simultaneously captured by multiple cameras), but may fail to reconstruct texture-less image regions. Learning multi-view depth estimators with [70] or without [33] depth supervision can compensate for lacking visual evidence. The method of Innmann et al. [27] can reconstruct mildly non-rigid objects, but still requires multiple views.

*Single-view* dense reconstruction of object categories was also addressed in prior works, but most of them require depth supervision [41, 57]. Among those that do not, [14] proposes a morphable model of dolphins supervised with 2D keypoints and segmentation masks, while Vicente et al. [63] and Carreira et al. [13] reconstruct the categories of PASCAL VOC. Most of these methods start by running a traditional SFM pipeline to obtain the mean 3D reconstruction and camera matrices. Kar et al. [31] replace it with NR-SFM for reconstructing categories from PASCAL3D+. VpDR [42] uses an image-driven approach for dense reconstruction of *rigid* categories from monocular views.

A number of recent mesh reconstruction methods based on *differentiable rendering* can also be trained with 2D supervision only. Kanazawa et al. [30] introduced CMR, a deep network that can reconstruct the shape and texture of deformable objects; it is the closest to our work in terms of assumptions, type of supervision, and output, and is currently state of the art for reconstruction of classes other than humans. DIB-R [16] uses a more advanced rendering technique that softly assigns all image pixels, including background, to the mesh faces. In contrast to these methods, we avoid computationally expensive rendering by leveraging NR-SFM pre-processing and cross-image consistency constraints.

**Canonical maps.** A *canonical map* is a function that maps image pixels to identifiers of the corresponding object points. Examples include the UV surface coordinates used by Dense Pose [22] and spherical coordinates [58]. Thewlis et al. [58, 59], Schmidt et al. [51] learn canonical maps in an unsupervised manner via a bottleneck, whereas Kulkarni et al. [35] do so by using consistency with an initial 3D model. Normalized Object Coordinate Space (NOCS) [65] also ties canonical coordinates and object pose, however it does not allow for shape deformation; different shapes within category have to be modelled by matching to one of the hand-crafted exemplars. Instead, we learn the dense parametric deformation model for each object category from 2D data.

## 3 Method

In this section and Figure 2, we describe the proposed representation and how to learn it.

### 3.1 The C3DM representation

**Canonical map.** Let $I \in \mathbb{R}^{3\times H\times W}$ be an image and $\Omega \subset \{1,\dots,H\}\times\{1,\dots,W\}$ be the image region that contains the object of interest. We consider a *canonical map* $\kappa = \Phi(\mathbf{y}; I)$ sending pixels $\mathbf{y} \in \Omega$ to points on the unit sphere $\kappa \in \mathbb{S}^2$, which is topologically equivalent to any 3D surface $\mathcal{S} \subset \mathbb{R}^3$ without holes. It can be interpreted as a space of indices or coordinates $\kappa$ that identify a dense system of 'landmarks' for the deformable object category. A landmark, such as the corner of the left eye in a human, is a point that can be identified repeatably despite object deformations. Note that the index space can take other forms than $\mathbb{S}^2$, however the latter is homeomorphic to most surfaces of 3D objects and has the minimum dimensionality, which makes it a handy choice in practice.

**Deformation model.** We express the 3D location of a landmark $\kappa$ as $\mathbf{X}(\kappa; I) = B(\kappa)\boldsymbol{\alpha}(I)$, where $\alpha(I) \in \mathbb{R}^D$ are image-dependent deformation parameters and $B(\kappa) \in \mathbb{R}^{3\times D}$ is a linear operator indexed by $\kappa$. This makes $B(\kappa)$ an *intrinsic* property, invariant to the object deformation or viewpoint change. The full 3D reconstruction $\mathcal{S}$ is given by the image of this map: $\mathcal{S}(I) = \{B(\kappa)\boldsymbol{\alpha}(I) : \kappa \in \mathbb{S}^2\}$. The reconstruction $\mathbf{X}(\mathbf{y}; I)$ specific to the pixel $\mathbf{y}$ is instead given by composition with the canonical map:

$$\mathbf{X}(\mathbf{y}; I) = B(\kappa)\boldsymbol{\alpha}(I), \quad \text{where} \quad \kappa = \Phi(\mathbf{y}; I). \tag{1}$$

**Viewpoint.** As done in NR-SFM, we assume that the 3D reconstruction is 'viewpoint-free', meaning that the viewpoint is modelled not as part of the deformation parameters $\boldsymbol{\alpha}(I)$, but explicitly, as a

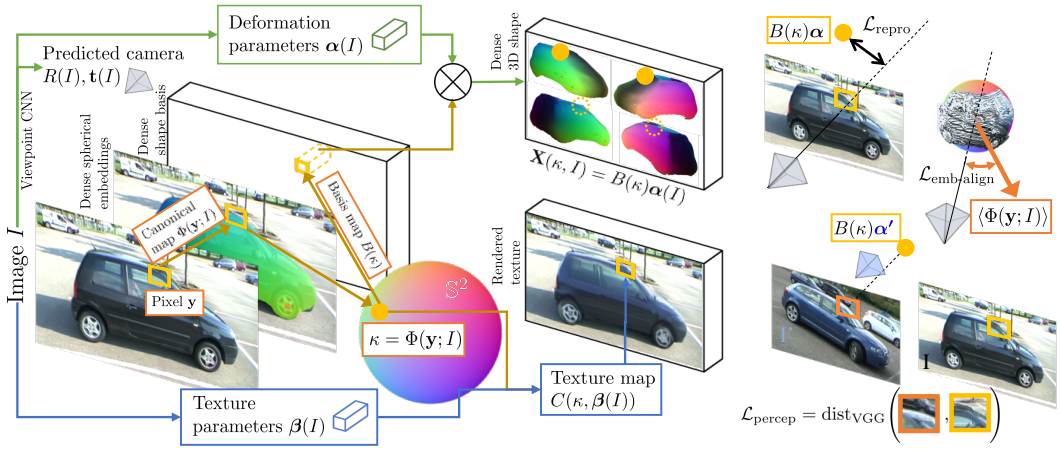


Figure 2: **Detailed system overview.** At test time, the image is passed through the network $\Phi$ to obtain the map of dense embeddings $\kappa \in \mathbb{S}^2$. The network $B$ converts them individually to deformation operators. In the meantime, the image is passed to the viewpoint network to predict the camera orientation $R$ and shape parameters $\boldsymbol{\alpha}$. Eq. (1) combines these quantities to obtain 3D reconstruction for each pixel within the object mask. At training time, sparse 2D keypoints are preprocessed with C3DPO [44] to obtain "ground truth" camera orientation $R^*$ and shape parameters $\boldsymbol{\alpha}^*$. These, together with the C3DPO basis $B^*$, are used in (4) to supervise the corresponding predicted variables. On the right, three more loss functions are illustrated: reprojection loss (5), cross-projection perceptual loss (6), and (8) aligning the camera orientation with average embedding direction.

separate rigid motion $(R(I), \mathbf{t}(I)) \in \mathbb{SE}(3)$. The rotation $R$ is regressed from the input image in the form proposed by Zhou et al. [75], and translation $\mathbf{t}(I)$ is found by minimizing the reprojection, see Section 3.2 for details. We assume to know the perspective/ortographic *camera model* $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ mapping 3D points in the coordinate frame of the camera to 2D image points (see sup. mat. for details). With this, we can recover the coordinates $\mathbf{y}$ of a pixel from its 3D reconstruction $\mathbf{X}(\mathbf{y}; I)$ as:

$$\mathbf{y} = \pi \left( R(I)\mathbf{X}(\mathbf{y}; I) + \mathbf{t}(I) \right). \tag{2}$$

Note that $\mathbf{y}$ appears on both sides of eq. (2); this lets us define the self-consistency constraint (5).

**Texture.** In addition to the deformation operator $B(\kappa)$, any intrinsic property can be descried in a similar manner. An important example is reconstructing the *albedo* $I(\mathbf{y})$ of the object, which we model as:

$$I(\mathbf{y}) = C(\kappa; \boldsymbol{\beta}(I)), \quad \kappa = \Phi(\mathbf{y}; I), \tag{3}$$

where $C(\kappa; \boldsymbol{\beta})$ maps a small number of image-specific texture parameters $\boldsymbol{\beta}(I) \in \mathbb{R}^{D'}$ to the color of landmark $\kappa$. In Section 4.1, we use this model to transfer texture between images of different objects.

**Implementation via neural networks.** The model above includes several learnable functions that are implemented as deep neural networks. In particular, the canonical map $\Phi(I)$ is implemented as an image-to-image convolutional network (CNN) with an $\mathbb{R}^{3\times H\times W}$ input (a color image) and an $\mathbb{R}^{3\times H\times W}$ output (the spherical embedding). The last layer of this network normalizes each location in $\ell^2$ norm to project 3D vectors to $\mathbb{S}^2$. Functions $\boldsymbol{\alpha}(I)$, $\boldsymbol{\beta}(I)$ and $R(I)$ predicting deformation, texture and viewpoint rotation are also implemented as CNNs. Translation $\mathbf{t}$ is found by minimising the reprojection, as explained below. Finally, functions $B(\kappa)$ and $C(\kappa)$ mapping embeddings to their 3D deformation and texture models are given by multi-layer perceptrons (MLP). The latter effectively allows $\kappa$, and the resulting 3D and texture reconstruction, to have arbitrary resolution.

### 3.2 Learning formulation

In order to train C3DM, we assume available a collection of independently-sampled views of an object category $\{I_n\}_{n=1}^N$.[2] Furthermore, for each view, we require annotations for the silhouette $\Omega_n$ of the object as well as the 2D locations of $K$ landmarks $Y_n = (\mathbf{y}_{n1}, \ldots, \mathbf{y}_{nK})$. In practice, this

[2] "Independent" means that views contain different object deformations or even different object instances.

information can often be extracted automatically via a method such as Mask R-CNN [25] and HRNet [55]. Note that we require to annotate only a small set of $K$ landmarks, whereas C3DM learns a continuos (infinite) set of those. We use the deformation basis from an NR-SFM method as a prior and add a number of consistency constraints for self-supervision, as discussed next.

**NR-SFM Prior.** Since our model generalizes standard parametric approaches, we can use any such method to bootstrap and accelerate learning. We use the output of the recent C3DPO [44] algorithm $\mathcal{A}_n^* = (B_n^*, \mathcal{V}_n^*, \boldsymbol{\alpha}_n^*, R_n^*)$ in order to anchor the deformation model $B(\kappa)$ in a visible subset $\mathcal{V}_n^*$ of $K$ discrete landmarks, as well as the deformation and viewpoint parameters, for each training image $I_n$.

Note that, contrary to C3DM, C3DPO takes as input the 2D location of the sparse keypoints both at training *and test time*. Furthermore, it can only learn to lift the keypoints for which ground-truth is available at training time. In order to learn C3DM, we thus need to learn from scratch the deformation and viewpoint networks $\boldsymbol{\alpha}(I)$ and $R(I)$, as well as the continuous deformation network $B(\kappa)$. This is necessary so that at test time C3DM can reconstruct the object in a dense manner given only the image $I$, not the keypoints, as input. At training time, we supervise the deformation and viewpoint networks from the C3DPO output via the loss:

$$\mathcal{L}_{\text{pr}}(\Phi, B, \boldsymbol{\alpha}, R; I, Y, \mathcal{A}^*) = \frac{1}{|\mathcal{V}^*|} \sum_{k \in \mathcal{V}^*} \|B(\Phi(\mathbf{y}_k; I)) - B_k^*\|_\epsilon + w_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}(I) - \boldsymbol{\alpha}^*\|_\epsilon + w_R d_\epsilon(R(I); R^*), \tag{4}$$

where $\|z\|_\epsilon$ is the pseudo-Huber loss [15] with soft threshold $\epsilon$ and $d_\epsilon$ is a distance between rotations.[3]

**Projection self-consistency loss.** As noted in Section 2, the composition of eqs. (1) and (2) must yield the identity function. This is captured by the *reprojection consistency loss*

$$\mathcal{L}_{\text{repro}}(\Phi, B, \boldsymbol{\alpha}, R; \Omega, I) = \min_{\mathbf{t} \in \mathbb{R}^3} \sum_{\mathbf{y} \in \Omega} \|\hat{\mathbf{y}}(\mathbf{t}) - \mathbf{y}\|_\epsilon, \quad \hat{\mathbf{y}}(\mathbf{t}) = \pi\Big(R(I)\, B(\Phi(\mathbf{y}; I))\, \boldsymbol{\alpha}(I) + \mathbf{t}\Big). \tag{5}$$

It causes the 3D reconstruction of an image pixel $\mathbf{y}$, which is obtained in a viewpoint-free space, to line up with $\mathbf{y}$ once the viewpoint is accounted for. We found optimizing over translation $\mathbf{t}$ in eq. (5) to obtain $\mathbf{t}(I, \Omega, \Phi, B, \boldsymbol{\alpha}, R)$ based on the predicted shape to be more accurate than regressing it directly. Refer to Appendix C in sup. mat. for optimization algorithm. We use the obtained value as the translation prediction $\mathbf{t}(I)$, in particular, in eq. (6), only implying the dependency on the predictors to simplify the notation. We backpropagate gradients from all losses through this minimization though.

**Apperance loss.** Given two views $I$ and $I'$ of an object, we can use the predicted geometry and viewpoint to establish dense correspondences between them. Namely, given a pixel $\mathbf{y} \in \Omega$ in the first image, we can find the corresponding pixel $\hat{\mathbf{y}}'$ in the second image as:

$$\hat{\mathbf{y}}' = \pi\Big(R(I')\, B(\Phi(\mathbf{y}; I))\, \boldsymbol{\alpha}(I') + \mathbf{t}(I')\Big). \tag{6}$$

This equation is similar to eq. (5), in particular, the canonical map is still computed in the image $I$ to identify the landmark, however the shape $\boldsymbol{\alpha}$ and viewpoint $(R, \mathbf{t})$ are computed from another image $I'$. Assuming that color constancy holds, we could then simply enforce $I(\mathbf{y}) \approx I'(\hat{\mathbf{y}}')$, but this constraint is violated for non-Lambertian objects or images of different object instances. We thus relax this constraint by using a *perceptual loss* $\mathcal{L}_{\text{percep}}$, which is based on comparing the activations of a pre-trained neural network instead [72]. Please refer to Appendix B in the sup. mat. for details.

Due to the robustness of the perceptual loss, most images $I$ can be successfully matched to a fairly large set $\mathcal{P}_I = \{I'\}$ of other images, even if they contain a different instance of the object. To further increase robustness to occlusions, large viewpoint differences, and other nuisance factors, inspired by Khot et al. [33], given a batch of training images, we compare each pixel in $I$ only to the $k \leq |\mathcal{P}_I|$ images $I'$ in the batch that match the pixel best. This bring us to the following formulation:

$$\mathcal{L}_{\text{percep}}^{\text{min-k}}(\Phi, B, \boldsymbol{\alpha}, R, \mathbf{t}; \Omega, I, \mathcal{P}_I) = \frac{1}{k} \sum_{\mathbf{y} \in \Omega} \min_{Q \subset \mathcal{P}_I : |Q| = k} \sum_{I' \in Q} \mathcal{L}_{\text{percep}}(\Phi, B, \boldsymbol{\alpha}, R, \mathbf{t}; \mathbf{y}, I, I'). \tag{7}$$

**Learning the texture model.** The texture model $(C, \beta)$ can be learned in a similar manner, by minimizing the combination of the photometric and perceptual (7) losses between the generated and original image. Please refer to the supplementary material for specific loss formulations. We do not back-propagate their gradients beyond the appearance model as it deteriorates the geometry.

[3] $\|z\|_\epsilon = \epsilon(\sqrt{1 + (\|z\|/\epsilon)^2} - 1)$; it behaves as a quadratic function of $\|z\|$ in the vicinity of 0 and a linear one when $\|z\| \to \infty$, which makes it both smooth and robust to outliers. See sup. mat. for definition of $d_\epsilon$.


Figure 3: **Canonical mapping and texture transfer for CUB and Freiburg Cars**. Given a target image $I_B$ (1st row), C3DM extracts the canonical embeddings $\kappa = \Phi(\mathbf{y}; I_B)$ (2nd row). Then, given the appearance descriptor $\boldsymbol{\beta}(I_A)$ of a texture image $I_A$ (4th row), the texture network $C$ transfers its style to get a styled image $I_C(\mathbf{y}) = C(\Phi_{\mathbf{y}}(I_B);\ \boldsymbol{\beta}(I_A))$ (3rd row), which preserves the geometry of the target image $I_B$. Note that we model the texture directly rather than warp the source image, so even the parts occluded in the source image $I_A$ can be styled (5th and 6th columns).

**Soft occlusion regularization.** We introduce a soft occlusion model that ties the spherical embedding space and camera orientation. It forces the model to use the whole embedding space and avoid re-using its parts for the regions of similar appearance, such as left and right sides of a car. We achieve it by aligning the direction of the mean embedding vector $\kappa$ with the camera direction, minimizing

$$\mathcal{L}_{\text{emb-align}}(\Phi, R; \Omega, I) = [0 \quad 0 \quad 1]\, R(I) \frac{\bar{\kappa}}{\|\bar{\kappa}\|}, \quad \text{where } \bar{\kappa} = \frac{1}{|\Omega|} \sum_{\mathbf{y} \in \Omega} \Phi(\mathbf{y}; I). \tag{8}$$

**Mask reprojection loss.** We observed that on some datasets like CUB Birds, the reconstructed surface tends to be noisy due to some parts of the embedding space overfitting to specific images. To prevent it interfering with other images, we additionally minimize the following simple loss function:

$$\mathcal{L}_{\text{mask}}(B, \boldsymbol{\alpha}, R, \mathbf{t}; \Omega) = \int_{\mathbb{S}^2} \left[\!\!\left[ \pi\Big( R\, B(\kappa)\, \boldsymbol{\alpha} + \mathbf{t} \Big) \notin \Omega \right]\!\!\right] d\kappa, \tag{9}$$

where we approximate the integration by taking a uniform sample of 1000 points $\kappa$ on a sphere.

## 4 Experiments

**Implementation details.** We build on the open-source implementation of C3DPO for preprocessing[4] and set $\boldsymbol{\alpha} \in \mathbb{R}^{10}$, $\boldsymbol{\beta} \in \mathbb{R}^{128}$. The canonical map network $\Phi$ uses the Hypercolumns architecture [23] on top of ResNet-50 [24], while basis and texture networks $B$ and $C$ are MLPs. See Appendices A and F in sup. mat. for description of the architecture, hyperparameters and optimization.

**Benchmarks.** We evaluate the method on a range of challenging datasets. We use C3DM to generate from each test image: (1) a full 360° shape reconstruction as a point cloud $\{B(\kappa)\boldsymbol{\alpha}(I) : \kappa \in \mathcal{K}\}$, where $\mathcal{K}$ consists of 30k sampled embeddings from random training set images, and (2) a depth map from the estimated image viewpoint obtained for each pixel $\mathbf{y} \in \Omega$ as the coordinate $z$ of $R\mathbf{X}(\mathbf{y}; I)$. We compare the full reconstructions against ground-truth point clouds using symmetric Chamfer distance $d_{\text{pcl}}$ (after ICP alignment [10]) and, whenever the dataset has depth maps or calibrations to project the ground-truth meshes, predicted depth maps against ground-truth depth maps as the average per-pixel depth error $d_{\text{depth}}$. In particular, to compute the symmetric Chamfer distance between the predicted and ground-truth point clouds $d_{\text{pcl}}(\hat{C}, C)$, we first correct the scale

[4] `https://github.com/facebookresearch/c3dpo_nrsfm`

ambuiguity by normalising the variance of the predicted point cloud to match ground truth. Then, we align them with ICP to obtain the $\tilde{C} = sR\hat{C} + \mathbf{t}$ rigidly aligned with $C$. We define Chamfer distance as the mean $\ell^2$ distance from each point in $C$ to its nearest neighbour in $\tilde{C}$ and make it symmetric:

$$d_{\text{pcl}}(\hat{C}, C) = \frac{1}{2}\Big(d_{Ch}(\tilde{C}, C) + d_{Ch}(C, \tilde{C})\Big), \quad \text{where } d_{Ch}(\tilde{C}, C) = \frac{1}{|C|}\sum_{\mathbf{X}\in C}\min_{\tilde{\mathbf{X}}\in\tilde{C}}\|\tilde{\mathbf{X}} - \mathbf{X}\|. \quad (10)$$

To compute the average per-pixel error between the predicted and ground-truth depth maps $d_{\text{depth}}(\hat{D}, D)$, we first normalize the predicted depth to have the same mean and variance as ground truth within the object mask $\Omega$ in order to deal with the scale ambuiguity of 3D reconstruction under perspective projection. Then, we compute the mean absolute difference between the the resulting depth maps within $\Omega$ as $d_{\text{depth}}(\hat{D}, D) = \frac{1}{|\Omega|}\sum_{\mathbf{y}\in\Omega}|\hat{D}_{\mathbf{y}} - D_{\mathbf{y}}|$.

We evaluate on **Freiburg Cars** [52] dataset, containing videos of cars with ground truth SfM/MVS point clouds and depth maps reporting $d_{\text{pcl}}$ and $d_{\text{depth}}$. In order to prove that C3DM can learn from independent views of an object category, we construct training batches so that the appearance loss (6) compares only images of *different* car instances. We further compare our model to the previously published results on a non-rigid category of human faces, training it on **CelebA** [39] and testing it on **Florence 2D/3D Face** [9]. The latter comes with ground-truth point clouds but no depth maps, so we report $d_{\text{pcl}}$ for the central portion of the face. As viewpoints don't vary much in the face data, we also consider **CUB-200-2011 Birds** [64], annotated with 15 semantic 2D keypoints. It lacks 3D annotations, so we adopt the evaluation protocol of CMR [30] and compare against them qualitatively. We compare to CMR using $d_{\text{pcl}}$ on 4 categories from **Pascal3D+** [67], which come with approximate ground-truth shapes obtained by manual CAD model alignment. See Appendix E for details.

**Baseline.** Our best direct competitor is CMR [30]. For CUB, we use the pre-trained CMR models made available by the authors, and for the other datasets we use their source code to train new models, making sure to use the same train/test splits. For depth evaluation, we convert the mesh output of CMR into a depth map using the camera parameters estimated by CMR, and for shape evaluation, we convert the mesh into a point cloud by uniformly sampling 30k points on the mesh.

### 4.1 Evaluating the canonical map

First, we evaluate the learned canonical map $\Phi_{\mathbf{y}}(I)$ qualitatively by demonstrating that it captures stable object correspondences. In row 2 of Figure 3, we overlay image pixels with color-coded 3D canonical embedding vectors $\kappa = \Phi_{\mathbf{y}}(I)$. The figure shows that the embeddings are invariant to viewpoint, appearance and deformation. Next, we make use of the texture model (3) to perform texture transfer. Specifically, given a pair of images $(I_A, I_B)$, we generate an image $I_C(\mathbf{y}) = C(\Phi_{\mathbf{y}}(I_B);\ \boldsymbol{\beta}(I_A))$ that combines the geometry of image $I_B$ and texture of image $I_A$. Row 3 of Figure 3 shows texture transfer results for several pairs of images from our benchmark data.

### 4.2 Evaluating 3D reconstructions

**Ablation study.** In Table 1, we evaluate the quality of 3D reconstruction by C3DM trained with different combinations of loss functions. It shows that each model components improves performance

| Active Losses $\mathcal{L}$ | | | | Fl. Face | Frei. Cars | |
|---|---|---|---|---|---|---|
| repro | basis | min-k percep | emb-align | $d_{\text{pcl}}$ | $d_{\text{depth}}$ | $d_{\text{pcl}}$ |
| | ✓ | ✓ | ✓ | 6.582 | 0.548 | 0.247 |
| ✓ | | ✓ | ✓ | 7.406 | 0.550 | 0.462 |
| ✓ | ✓ | | ✓ | 5.647 | 0.361 | 0.141 |
| ✓ | ✓ | ✓ | | 5.592 | 0.498 | 0.186 |
| ✓ | ✓ | ✓ | ✓ | **5.574** | **0.311** | **0.123** |

Table 1: **3D reconstruction accuracy for different variants of C3DM on Freiburg Cars and Florence Face.** We evaluate the effect of disabling losses (5), (7), (8), and the first term in (4), one-by-one.

| Dataset | CMR [30] | C3DM |
|---|---|---|
| Flo. Face | 13.09 | **5.57** |
| Frei. Cars | 0.20/0.50 | **0.12/0.31** |
| P3D Plane | 0.022 | **0.019** |
| P3D Chair | 0.049 | **0.043** |
| P3D Car | 0.028 | 0.028 |
| P3D Bus | 0.037 | **0.036** |

Table 2: $d_{\text{pcl}}$ **on Freiburg Cars, Florence Face, and Pascal 3D+** comparing our method to CMR [30]. For Frei. Cars, $d_{\text{depth}}$ is also reported after slash.

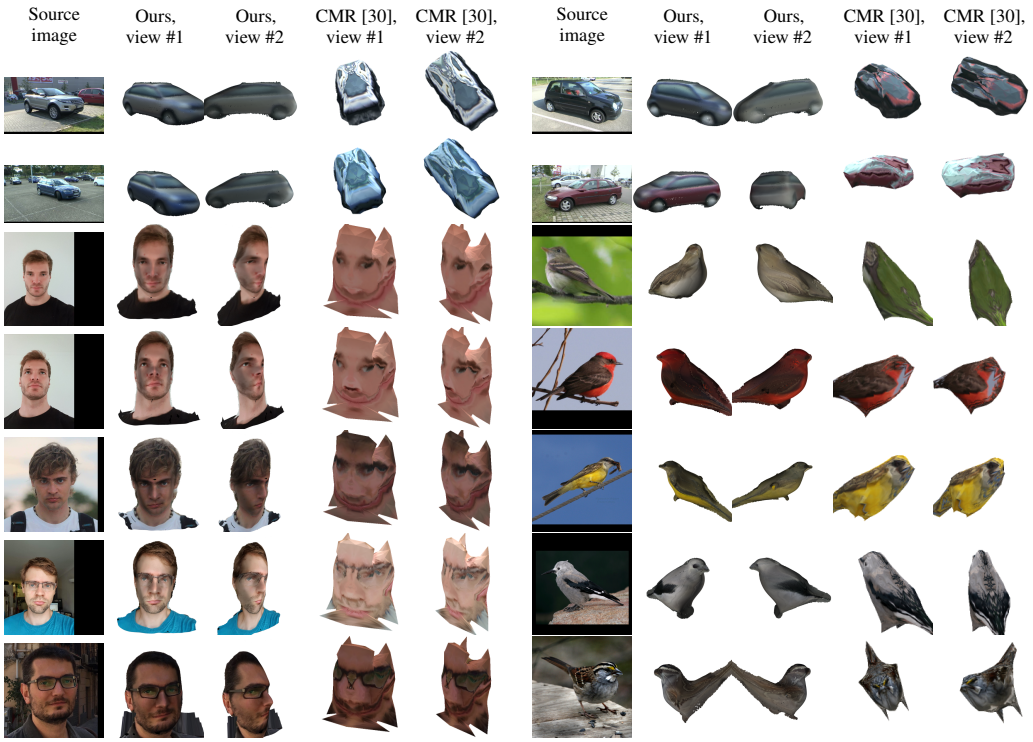


Figure 4: **Visual comparison of the results on Freiburg Cars (top two rows), face images (left column) and CUB Birds datasets (right column).** For each dataset, we show the source image (1st column), C3DM and CMR reconstructions from the original viewpoint (*view #1*, 2nd and 4th columns, respectively) and from an alternative viewpoint (*view #2*, 3rd and 5th columns).

across all metrics and datasets. The contribution of the appearance loss (7) is higher for cars, where the keypoints are sparse; for faces, on the other hand, the network can get far by interpolating between the embeddings of the 98 landmarks even without appearance cues. The camera-embedding alignment loss (8) is also more important for cars because of the higher viewpoint diversity.

**Comparison with the state-of-the-art.** Table 2 compares the Chamfer distance $d_{\text{pcl}}$ and depth error $d_{\text{depth}}$ (where applicable) of C3DM against CMR [30]. On Freiburg Cars and Florence Face, our method attains significantly better results than CMR. C3DM produces reasonble reconstructions and generally outperforms CMR on four categories from Pascal3D+ with big lead on chairs. The visualisations in Figure 4 confirm the trend: C3DM is better at modelling fine details.

On Freiburg Cars, our method can handle perspective distortions better and is less dependent on instance segmentation failures since it does not have to satisfy the silhouette reprojection loss. On CelebA, CMR, which relies on this silhouette reprojection loss, produces overly smooth meshes that lack important details like protruding noses. Conversely, C3DM leverages the keypoints lifted by C3DPO to accurately reconstruct noses and chins. On CUB Birds, it is again apparent that C3DM can reconstruct fine details like beaks. See Appendix G and videos for more visual results.

## 5 Conclusions

We have presented C3DM, a method that learns under weak 2D supervision to densely reconstruct categories of non-rigid objects from single views, establishing dense correspondences between them in the process. We showed that the model can be trained to reconstruct diverse categories such as cars, birds and human faces, obtaining better results than existing reconstruction methods that work under the same assumptions. We also demonstrated the quality of dense correspondences by applying them to transfer textures. The method is still limited by the availability of some 2D supervision (silhouettes and sparse keypoints) at training time. We aim to remove this dependency in future work.

## Potential broader impact

Our work achieves better image-based 3D reconstruction than the existing technology, which is already available to the wider public. While we outperform existing methods on benchmarks, however, the capabilities of our algorithm are not sufficiently different to be likely to open new possibilities for misuse.

Our method interprets images and reconstructs objects in 3D. This is conceivably useful in many applications, from autonomy to virtual and augmented reality. Likewise, it is possible that this technology, as any other, could be misused. However, we do not believe that our method is more prone to misuse than most contributions to machine learning.

In particular, there could be some concerns that 3D reconstructions could be used for extracting biometrics or for re-enacting. However, our reconstruction technique is generic and, as such, there exist specialized methods that achieve better results on these specific tasks, e.g. for human faces. For example, it would be next to impossible to recognize an identity from our geometric reconstructions alone due to imprecisions in the fine details of the geometry.

As for any research output, there is an area of uncertainty on how our contributions could be incorporated in future research work and the consequent impact of that. We believe that our advances are methodologically significant, and thus we hope to have a positive impact in the community, leading to further developments down the line. However, it is very difficult to predict the nature of all such possible developments.

# A Architecture details

Figure V shows the backbone of our architecture, together with the basis and texture predictors $B$ and $C$. The trunk of C3DM consists of a Feature Pyramid Network pre-trained on ImageNet. In more detail, Conv-Upsample blocks are attached to the outputs of each of the Res1, Res2, Res3 and Res4 layers of a ResNet50. Each Conv-Upsample outputs a tensor with the spatial resolution of the first auxiliary branch that takes Res1 as an input. The four tensors are then summed and $\ell^2$-normalized in order to produce the canonical embedding tensor $\kappa$.

The insets of Figure V show the architecture of the basis and texture networks $B(\kappa)$ and $C(\kappa, \boldsymbol{\beta}(I))$. The networks follow the C3DPO [44] architecture. Each of them consists of a fully connected (FC) layer, followed by three fully connected residual blocks (shown in detail in the lower-right inset) and another fully connected layer adapting the output dimensionality. The LayerNorm layers [8] used in these networks only perform $\ell^2$ normalization across channels, without using trainable parameters. The basis network takes as input the map of 2D canonical embeddings $\kappa$, while the texture network concatenates them with the same texture descriptor $\boldsymbol{\beta}$ to get the 130-dimensional vector for each pixel. The basis network outputs the 30-dimensional vector for each pixel (10 3-dimensional basis vectors), while the texture network outputs 3D per-pixel colors.

Figure VI extends the diagram with the computations specific to the training time. For supervision, the training also runs C3DPO on 2D keypoints and uses the predictions and bases to define the NR-SFM prior loss (4). The diagram also shows the reprojection consistency loss (5), cross-image perceptual loss (6), which requires the viewpoint and shape predictions for other images in the batch, camera-embedding alignment loss (8), and the texture model loss (13).

**Batch sampling.** In each training epoch, we sample 3000 batches of 10 random images (adding a constraint on Freiburg Cars that they don't come from the same sequence). We optimize the network using SGD with momentum, starting with learning rate 0.001 and decreasing $10\times$ whenever the objective plateaus. We stop training after 50 epochs.

Since most datasets are biased in terms of the viewpoints, e.g. birds are less likely to be photographed from the front or back than from the side, we apply inverse propensity correction on the distribution of 1D rotations to ensure uniform coverage. We correct the distribution of rotations in the horizontal plane only, assuming that the pitch varies less than the azimuth, which is true for most object-centric datasets. In particular, we first find the upward direction as an eigenvector of the rotation axes extracted from the camera orientations extracted by NR-SFM from the training set: $\{R_n^*\}$. Then we compute the azimuth $a(R_n^*)$ as the rotation component around the estimated upward axis. The sampling weight for an image $I_n$ is thus found as $\big(p(a(R_n^*))\big)^{-1}$, where the distribution $p$ is approximated by a histogram of 16 bins. Note that we only need to do this at training time when NR-SFM viewpoint predictions are available; at test time, the networks can take a single image.

To compute the min-k cross-image perceptual loss (6), we treat the first image $I$ in the batch as a target and warp the rest of the images using their *estimated* camera and shape parameters $R(I'), \mathbf{t}(I'), \boldsymbol{\alpha}(I')$. For each pixel, we average the distances to $k = 6$ closest feature maps as per eq. (6).

**Implementation.** We implemented C3DM using Pytorch framework. We run training on a single NVidia Tesla V100 GPU with 16 Gb of memory. Training for full 50 epochs takes around 48 hours.

**Runtime analysis** On a single gpu, the feedforward pass of our network takes one average 0.111 sec per image.

# B Details of the photometric and perceptual losses

To enforce photometric consistency, we can use the following loss:

$$\mathcal{L}_{\text{photo}}(I'; \Omega, I) = \sum_{\mathbf{y} \in \Omega} \|I'(\mathbf{y}) - I(\mathbf{y})\|_\epsilon. \tag{11}$$

Here $I$ and $I'$ are two images, $\Omega$ is the region of image $I$ that contains the object (*i.e.*the object mask).

To capture higher-level consistency between images, in particular in the cross-image consistency loss (6) between the target image and warped reference image, we use *perceptual loss* $\mathcal{L}_{\text{percep}}$ that

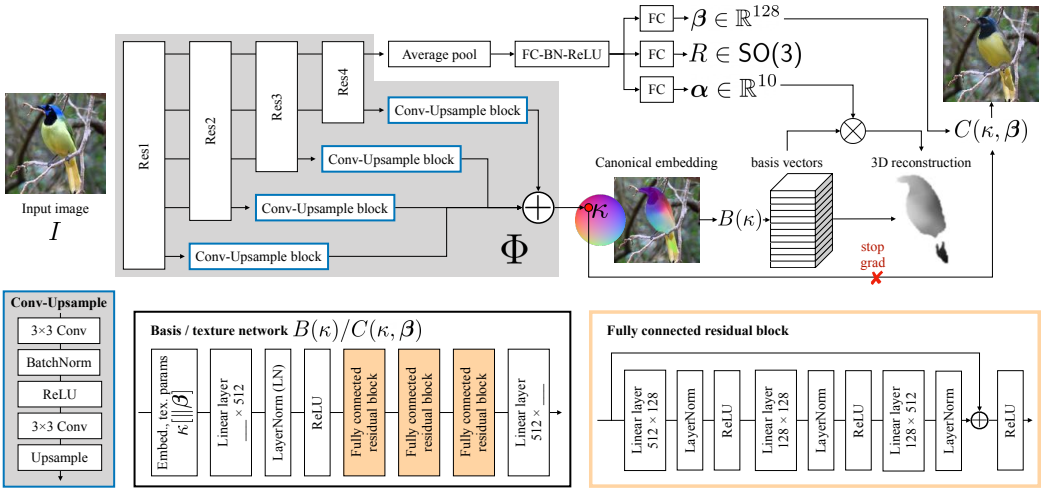


Figure V: **The detailed architecture of prediction-time C3DM flow.** All networks share the common ResNet50 backbone. Camera orientation, shape and texture parameters are regressed from the final residual layer. The embedding prediction network $\Phi$ processes outputs of the four residual blocks with the Conv-Upsample subnetwork shown in the left inset, then sums and normalises their outputs to obtain the map of spherical embeddings $\kappa$. They are passed through basis and texture networks that share the architecture, which is shown in the middle and right insets. Finally, the predicted basis vectors are multiplied by shape parameters $\boldsymbol{\alpha}$ to obtain 3D reconstruction of the visible points.

compares the activations of a pre-trained neural network [72]. Specifically, we compute pseudo-Huber loss between the activations of a VGG network, averaged over several layers. The perceptual loss uses the pretrained VGG-19 network [54]. Let $\Psi_l(I)$ be the layer $l$ activations of VGG-19 fed by the image $I$. We then define the perceptual loss as

$$\mathcal{L}_{\text{percep}}(I'; \Omega, I) = \sum_{\mathbf{y} \in \Omega} \sum_{l \in \{0,5,10,15\}} \left\| \text{upsample}\big(\Psi_l(I') - \Psi_l(I)\big)[\mathbf{y}] \right\|_\epsilon, \tag{12}$$

where upsample() interpolates the feature map to the match the resolution of the network input.

We can now formally define the optimisation problem for the texture model dwscribed in Section 3.1. Given the input image $I$ and 2D embeddings for all its pixels $\boldsymbol{\kappa}$, it re-produces the image $I'$ using $I'(\mathbf{y}) = C(\kappa(\mathbf{y}); \boldsymbol{\beta}(I))$. The weights of neural networks implementing $C$ and $\boldsymbol{\beta}$ are found by minimising

$$\mathcal{L}_{\text{tex}}(I'; \Omega, I) = w_{\text{photo}} \mathcal{L}^{\text{tex}}_{\text{photo}}(I'; \Omega, I) + w_{\text{percep}} \mathcal{L}^{\text{tex}}_{\text{percep}}(I'; \Omega, I). \tag{13}$$

Please note again that the gradients of $\mathcal{L}_{\text{tex}}$ are not propagated beyond $\boldsymbol{\kappa}$ to preserve its sole dependence on geometry.

## C Camera models and ray-projection loss

**Camera models.** We have to define a camera model $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ mapping 3D points in the coordinate frame of the camera to 2D image points in order to compute reprojection and photometric losses. If the camera calibration is unknown (as in CelebA, Florence Face, CUB, Pascal 3D+ datasets), we use an *orthographic camera* $\pi(\mathbf{X}) = [x_1, x_2]^\top$ where $\mathbf{X} = [x_1, x_2, x_3]^\top$. In this case, we also set $\mathbf{t} = 0$ as translation can be removed by centering the 2D data [44] in pre-processing.

If the camera calibration is known (in Freiburg Cars), we can also use a more accurate *perspective camera model* instead:

$$\pi(\mathbf{X}) = \frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{14}$$

where $f$ is the focal length.

Further to Section 3.1, here, we describe additional implementation details that were important for the success of the perspective projection model on the Freiburg Cars dataset.

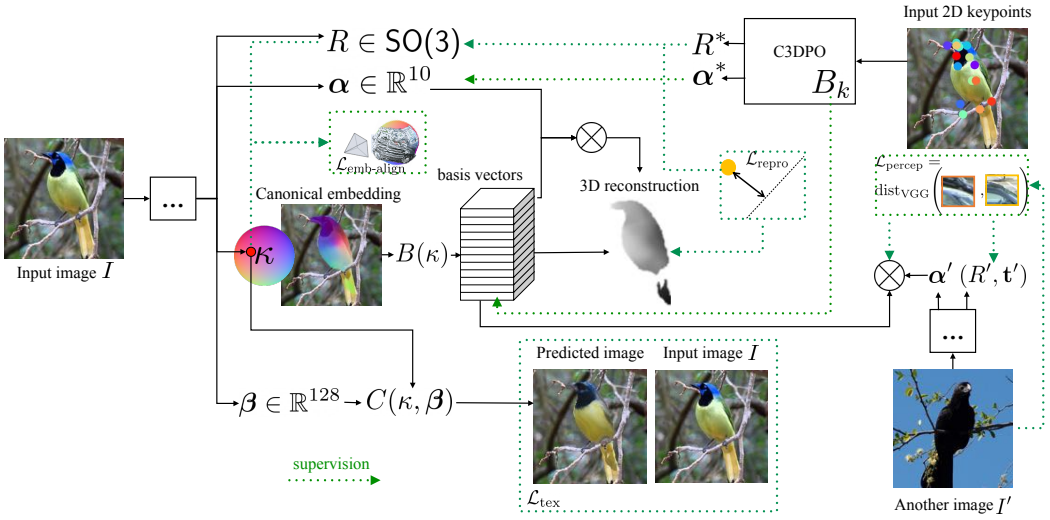


Figure VI: **The training time C3DM flow,** where the backbones showed in Figure V are collapsed to the boxes with ellipses. We supervise the predicted basis map with C3DPO bases at keypoint locations. At training time, we also run C3DPO on 2D keypoints to supervise shape parameters and camera orientation. Embedding alignment loss acts on the estimated camera orientation and average spherical embeddings. We project the 3D reconstruction using the estimated camera parameters to define the reprojection consistency loss. To define the cross-image perceptual consistency loss, we run our network on another image (in practice, the other images in the batch are used) and use its shape and camera parameters to project the estimated basis vectors and compare with that image. Finally, we supervise the output of the texture model with the original image.

**Ray-projection loss** For perspective model, we have also found an improvement that significantly stabilizes the C3DPO algorithm that we use to constrain C3DM. The idea is to modify reprojection loss to measure, instead of the distance between 2D projections $\mathbf{y}$ and $\hat{\mathbf{y}}$, the distance of the 3D point $\mathbf{X}(\mathbf{y})$ to the line passing through $\mathbf{y}$ and the camera center. The advantage is removing the division embedded in the perspective projection equation (14).

In order to minimize the reprojection error (5) under the perspective projection model, a naïve implementation would minimize the following perspective re-projection loss:

$$\mathcal{L}^{\text{persp}}_{\text{repro}}(\Phi; \Omega, I) = \sum_{\mathbf{y} \in \Omega} \left\| \pi_{I,\mathbf{0}}(\mathbf{X}_{R,\mathbf{t}}(\mathbf{y})) - \mathbf{y} \right\|_{\epsilon}, \tag{15}$$

where $\mathbf{X}_{R,\mathbf{t}}(\mathbf{y}) = R\mathbf{X}(\mathbf{y}) + \mathbf{t}$ is the 3D point extracted from pixel $\mathbf{y}$ and expressed in the coordinate frame of the camera of the image $I_n$. Unfortunately, we found that the division in the perspective projection formula $\pi_{I,\mathbf{0}} = \frac{f}{x_3}[x_1\ x_2]^\top$ leads to unstable training. This is due to exploding gradient magnitudes caused by 3D points $\mathbf{X}$ predicted to lie too close to the camera projection plane. While this could be extenuated by clamping the points to lie in a safe distance from the camera plane, due to the non-linearity of the projection gradient, the re-projection loss (15) still would not converge stably.

In order to remove the gradient non-linearity, we alter the re-projection loss to the *ray-projection* loss:

$$\mathcal{L}^{\text{ray}}_{\text{repro}}(\Phi; \Omega, I) = \sum_{\mathbf{y} \in \Omega} \left\| \mathbf{X}_{R,\mathbf{t}}(\mathbf{y}) - \left[\mathbf{r}(\mathbf{y})^\top \mathbf{X}_{R,\mathbf{t}}(\mathbf{y})\right] \mathbf{r}(\mathbf{y}) \right\|_{\epsilon}, \tag{16}$$

where $\mathbf{r}(\mathbf{y})$ stands for the direction vector of the projection ray passing through the pixel $\mathbf{y}$ in the image $I$:

$$\mathbf{r}(\mathbf{y}) = \frac{K^{-1}[y_1\ y_2\ 1]^\top}{\|K^{-1}[y_1\ y_2\ 1]^\top\|},$$

where $K$ is the instrinsic camera calibration matrix. Intuitively, eq. (16) minimizes the orthogonal distance between the the estimated point $\mathbf{X}_{R,\mathbf{t}}(\mathbf{y})$ and its projection on the ground truth projection ray $\mathbf{r}(\mathbf{y})$. We notice that eq. (16) is linear in $\mathbf{X}_{R,\mathbf{t}}$ on infinity and quadratic in the compact region around the optimum, hence the magnitude of the gradient is bounded from above. We found this addition important for convergence of C3DM.

**Perspective projection for C3DPO** In order to optimize eq. (16), a C3DPO model [44] trained using the perspective projection model is required. Since the original C3DPO codebase only admits orthographic cameras, we will describe additions to the pipeline that enable training a perspective model on Freiburg Cars.

C3DPO optimizes a combination of canonicalization and reprojection losses. To this end, we replace the original C3DPO reprojection loss (eq. (4) in [44]) with the ray-projection loss (16). Additionally, unlike in the orthographic case, one has to determine the full 3DoF position of the camera w.r.t. the object coordinate frame. While it is possible to let C3DPO predict translation as an additional output of the network, we avoid over-parametrization of the problem by estimating camera translation as a solution to a simple least-squares problem.

In more detail, we exploit the locally quadratic form of the ray-projection loss and formulate the translation estimation problem that allows for a closed-form solution. Assuming that C3DPO, given a list of input 2D landmarks $\mathbf{y}_1, ..., \mathbf{y}_K$, predicts a camera rotation matrix $R$, the translation can be obtained as a solution to the following problem:

$$\mathbf{t}^* = \text{argmin}_\mathbf{t} \sum_{i=1}^{K} \left\| \mathbf{X}_{R,\mathbf{t}}(\mathbf{y}_k) - \mathbf{r}(\mathbf{y}_k)^\top \mathbf{X}_{R,\mathbf{t}}(\mathbf{y}_k)\mathbf{r}(\mathbf{y}_k) \right\|^2.$$

After a few mathematical manipulations, we arrive at the following closed-form expression for $\mathbf{t}^*$:

$$\mathbf{t}^* = \left[ \sum_{k=1}^{K} (I - \Gamma_k) \right]^{-1} \left[ \sum_{k=1}^{K} (\Gamma_k - I)\mathbf{X}_{R,\mathbf{0}} \right], \tag{17}$$

where $\Gamma_k = \mathbf{r}(\mathbf{y}_k)\mathbf{r}(\mathbf{y}_k)^\top$ is an outer product of $\mathbf{r}(\mathbf{y}_k)$ with itself. Using eq. (17), we can estimate the camera translation online during the SGD iterations of the C3DPO optimization. Note that the matrix inverse in eq. (17) is not an issue because of the small size of the matrix being inverted ($3\times3$) and the possibility to backpropagate through matrix inversion using modern automatic differentiation frameworks (PyTorch).

## D Rotation loss

We use the distance between rotation matrices $d_\epsilon(R, R^*)$ as part of the loss (4). We aim to penalise large angular distance, while avoiding the exploding gradients of inverse trigonometric functions. First, we note that the relative rotation can be computed as $R^\top R^*$. Next, converting it to the axis-angle representation lets us compute the angular component as $\theta = \arccos\left(\frac{1}{2}(\text{Tr}(R^\top R^*) - 1)\right)$. Using the fact that $\arccos$ is monotonically decreasing, we strip it and apply an affine transform to make sure the loss achieves the minimum at 0:

$$d_\epsilon(R, R^*) = 1 - \cos\theta = \frac{3 - \text{Tr}(R^\top R^*)}{2}. \tag{18}$$

## E Datasets

**Freiburg Cars (FrC).** In order to test our algorithm in a low-noise setting, we consider the Freiburg cars dataset [52][5] containing walkaround videos of 52 cars. While this dataset contains videos of the cars, in order to test the ability of the photometric loss (7) to reconstruct objects even if the views are independent, we pair each pivot image $I$ with a selection of other images $\mathcal{P}_I$ extracted from *different* video sequences.

Following Novotny et al. [43, 42], we set out 5 sequences for validation (indexed 22, 34, 36, 37, 42). The training set contains 11,162 training frames and 1,427 validation frames. For evaluation, we also use their ground-truth 3D point clouds, but we only retain the 3D points that, after being projected into each image of a given test sequence, fall within the corresponding segmentation mask. Each point cloud is further normalized to zero-mean and unit variance along the 3 coordinate axes. Please refer to [43] for details.

[5] `https://github.com/lmb-freiburg/unsup-car-dataset`

As an input to our method, we use the pre-trained Mask R-CNN of [25] to extract the segmentation masks and the HRNet [32] trained on PASCAL 3D+ [68] to extract the 2D keypoints. Hence, all inputs to our method are extracted automatically. We excluded the frames where a car was detected with a confidence below a threshold.

We report the Chamfer distance $d_{\text{pcl}}$ between the ground truth and the predicted point clouds after rigid alignment via ICP [10]. The point cloud predictions are obtained as explained in the *Benchmarks* section of the main text, with $|\mathcal{B}| = 30\text{k}$. Furthermore, we evaluate the quality of our depth predictions by measuring the average depth distance $d_{\text{depth}}$ between the point cloud formed by un-projecting the predicted depth map and the visible part of the ground truth point cloud.

**CelebA and Florence faces (FF).** The FrC dataset contains deformation between object instances, but each object itself is rigid. In order to compare the ability of our method to handle instance-level non-rigid deformations with the CMR's, we also run the method on images of human faces; in particular, we train our algorithm on the training set of CelebA dataset [39][6] containing 161,934 face images and test it on the Florence 2D/3D Face dataset [9][7]. The latter contains videos of 53 people and their ground truth 3D meshes, which we can use to assess the quality of our 3D reconstructions. Following a standard practice, we crop each 3D mesh to retain points that lie within 100mm distance from the nose tip. We extract 98 semantic keypoints for each training and test face using the pre-trained HRNet detector of [55].

For evaluation on FF, five frames are uniformly sampled from each test sequence. We then use our network to reconstruct each test face in 3D and evaluate $d_{\text{pcl}}$ after ICP alignment. Since the extent of the predicted face differs from the ground truth, we first pre-align the prediction by registering a 3D crop that covers the convex hull of the 98 semantic keypoints. The 100mm nose-tip crop is then extracted from the pre-aligned mesh and is aligned for the second time. $d_{\text{depth}}$ is not reported for FF since the dataset does not contain ground truth per-frame depth.

**CUB-200-2011 Birds.** We evaluate our method qualitatively on the CUB Birds dataset [64][8], which consists of 11,788 still images of birds belonging to 200 species. Each image is annotated with 15 semantic keypoints. As done in [44], for evaluation we use detections of a pre-trained HRNet. The dataset is challenging mainly due to significant shape variations across bird species, in addition to instance-level articulation. Since there is no 3D ground truth for that dataset, we qualitatively compare the quality of 3D reconstruction to the ones of CMR [30]. We also use the same training/validation split as CMR.

**Pascal3D+.** We provide additional comparison to CMR on four categories of Pascal3D+ [67][9]: aeroplane, consisting of 1194 training and 1135 test images, bus (674 training / 657 test), car (2765 training / 2713 test), and chair (650 training / 666 test). It has been manually annotated by rigidly aligning one of category-specific CAD models, so the annotation has noisy and biased shape and pose. Since the original CMR codebase contains models for only two classes, we trained CMR models on all considered classes ourselves (using their codebase) and test on the corresponding validation sets. We report only $d_{\text{pcl}}$, since the depth maps obtained by projecting with noisy cameras are unreliable.

[6] `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`

[7] `http://www.micc.unifi.it/masi/research/ffd/` ©Copyright 2011–2019 MICC — Media Integration and Communication Center, University of Florence. The Florence 2D/3D Face Dataset.

[8] `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`

[9] `https://cvgl.stanford.edu/projects/pascal3d.html`

## F Hyperparameters used in experiments

To sum up, during training, we optimize the following weighted sum of loss functions:

$$\begin{aligned}\mathcal{L}(\Phi, B, \boldsymbol{\alpha}, R, \mathbf{t}, \hat{I}; \Omega, I, \mathcal{P}_I, \mathcal{A}^*) = \; & w_{\text{pr}}\mathcal{L}_{\text{pr}}(\Phi, B, \boldsymbol{\alpha}, R; I, Y, \mathcal{A}^*) + \\ & w_{\text{repro}}\mathcal{L}_{\text{repro}}(\Phi, B, \boldsymbol{\alpha}, R; \Omega, I) + \\ & w^{\text{min-k}}_{\text{percep}}\mathcal{L}^{\text{min-k}}_{\text{percep}}(\Phi, B, \boldsymbol{\alpha}, R, \mathbf{t}; \Omega, I, \mathcal{P}_I) + \\ & w_{\text{emb-align}}\mathcal{L}_{\text{emb-align}}(\Phi, R; \Omega, I) + \\ & w_{\text{mask}}\mathcal{L}_{\text{mask}}(B, \boldsymbol{\alpha}, R, \mathbf{t}; \Omega) + \\ & \mathcal{L}_{\text{tex}}(\hat{I}; \Omega, I).\end{aligned} \tag{19}$$

We set most weights such that the corresponding term has a magnitude of about 1 in the beginning of training. We set $w_{\text{pr}} = 1$, $w_{\boldsymbol{\alpha}} = 1$, $w_{\text{repro}} = 1$ for the perspective camera model and $w_{\text{repro}} = 0.01$ for the orthographic one, where the error is measured in pixels rather than world units. For the components of texture loss, we set $w^{\text{tex}}_{\text{photo}} = 1$, and $w^{\text{tex}}_{\text{percep}} = 0.1$. Likewise, we set the weight for the geometry perceptual loss $w^{\text{min-k}}_{\text{percep}} = 0.1$. We ran grid search for the camera-related parameters within the following ranges: $w_R \in \{1, 10\}$, and $w_{\text{emb-align}} \in \{1, 10\}$. We enable $\mathcal{L}_{\text{mask}}$ for CUB Birds, Faces, and Pascal3D+ aeroplanes and chairs with weight $w_{\text{mask}} = 1$.

## G Additional qualitative results

Figures VII and VIII contain additional single-view reconstruction results. We can see that C3DM is robust to occlusions and instance segmentation failures: the 3D shape is reasonably completed in those cases. Furthermore, Figures IX and X have been populated with supplemental texture transfer results. Note that all images are taken from the test set, and images from the same FrC sequence do not co-occur in training and test sets. We also invite the readers to watch the attached videos of the rendered reconstructions to better evaluate 3D reconstruction quality.

## References


[1] Antonio Agudo and Francesc Moreno-Noguer. Dust: Dual union of spatio-temporal subspaces for monocular multiple object 3d reconstruction. In *Proc. CVPR*, 2017.

[2] Antonio Agudo and Francesc Moreno-Noguer. Deformable motion 3D reconstruction by union of regularized subspaces. In *Proc. ICIP*, 2018.

[3] Antonio Agudo, Melcior Pijoan, and Francesc Moreno-Noguer. Image collection pop-up: 3D reconstruction and clustering of rigid and non-rigid categories. In *Proc. CVPR*, 2018.

[4] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. In *Proc. NIPS*, 2009.

[5] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *PAMI*, 33(7):1442–1456, 2011.

[6] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. In *ACM Trans. on Graphics*, 2005.

[7] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph description of edge segments. Technical Report 559, INRIA, 1986.

[8] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. Technical report, 2016. URL `https://arxiv.org/abs/1607.06450`.

[9] Andrew D. Bagdanov, Iacopo Masi, and Alberto Del Bimbo. The florence 2d/3d hybrid face datset. In *Proc. of ACM Multimedia Int.'l Workshop on Multimedia access to 3D Human Objects (MA3HO'11)*. ACM Press, December 2011.

[10] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992.

[11] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *Proc. ECCV*, 2016.

[12] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3D shape from image streams. In *Proc. CVPR*, 2000.

[13] Joao Carreira, Abhishek Kar, Shubham Tulsiani, and Jitendra Malik. Virtual view networks for object reconstruction. In *Proc. CVPR*, 2015.

[14] Thomas J Cashman and Andrew W Fitzgibbon. What shape are dolphins? building 3d morphable models from 2d images. *PAMI*, 35(1):232–244, 2013.

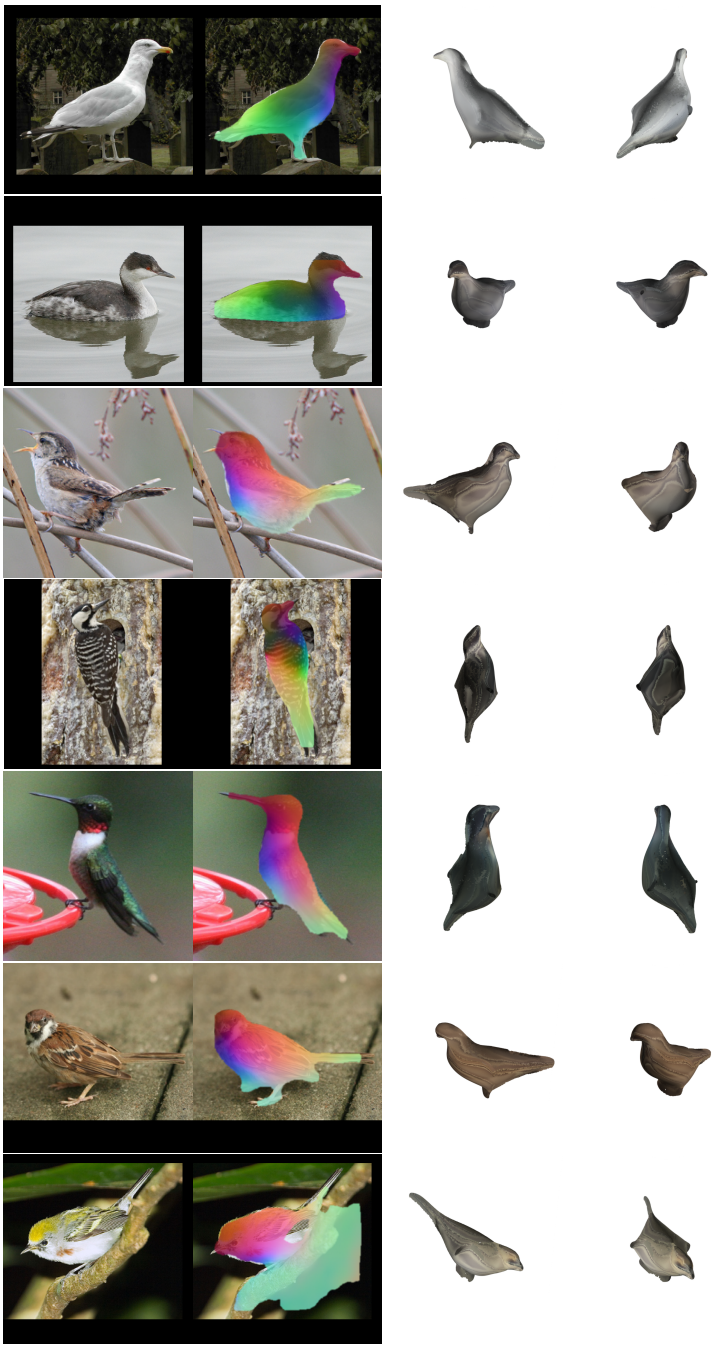

Figure VII: **Additional single-view reconstruction results on images from the test sequences of CUB Birds**. Columns: input image; canonical mapping; 3D reconstruction with the reconstructed texture from two viewpoints.


[15] Pierre Charbonnier, Laure Blanc-féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. Image Processing*, 6:298–311, 1997.

[16] Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Proc. NeurIPS*, 2019. URL `http://arxiv.org/abs/1908.01210`.

[17] Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014.

[18] Katerina Fragkiadaki, Marta Salas, Pablo Arbelaez, and Jitendra Malik. Grouping-based low-rank trajectory completion and 3D reconstruction. In *Proc. NIPS*, 2014.

Figure VIII: **Additional single-view reconstruction results on images from the test sequences of Freiburg Cars**. Columns: input image; canonical mapping; 3D reconstruction with the reconstructed texture from two viewpoints.


[19] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. CVPR*, 2017.

[20] Paulo FU Gotardo and Aleix M Martinez. Non-rigid structure from motion with complementary rank-3 spaces. In *Proc. CVPR*, 2011.

[21] P. Guan, A. Weiss, A. O. Balan, and M. J. Black. Estimating human shape and pose from a single image. In *Proc. ICCV*, 2009.

[22] A. Güler, N. Neverova, and I. Kokkinos. DensePose: Dense human pose estimation in the wild. In *Proc. CVPR*, 2018.

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proc. CVPR*, 2015.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. ICCV*, 2017.

[26] Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V. Gehler, Javier Romero, Ijaz Akhter, and Michael J. Black. Towards accurate marker-less human shape and pose estimation over time. In *Proc. 3DV*, 2017.

[27] Matthias Innmann, Kihwan Kim, Jinwei Gu, Matthias Niessner, Charles Loop, Marc Stamminger, and Jan Kautz. NRMVS: Non-Rigid Multi-View Stereo. 2019. URL `http://arxiv.org/abs/1901.03910`.

[28] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3D deformation model for tracking faces, hands, and bodies. In *Proc. CVPR*, 2018.

[29] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proc. CVPR*, 2018.

[30] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, 2018.

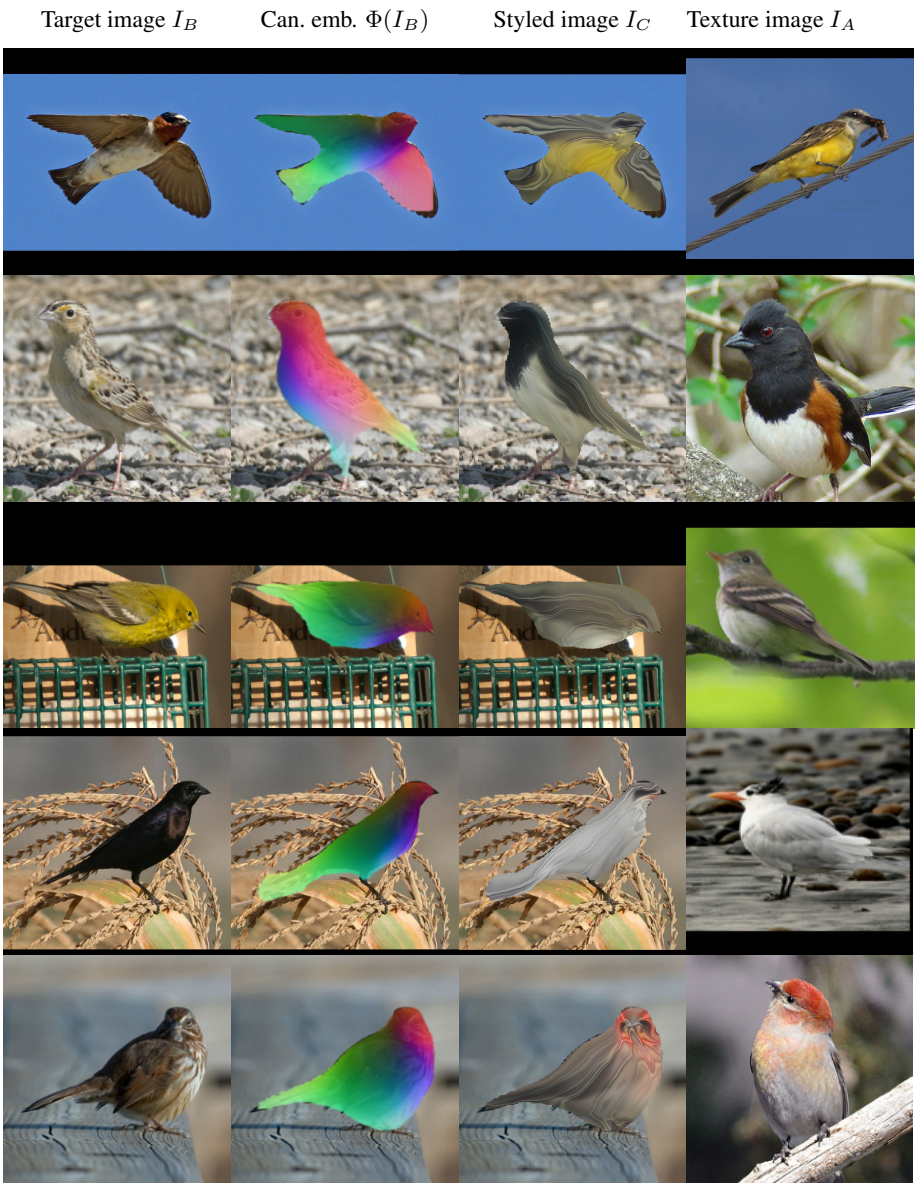


Figure IX: **Canonical mapping and texture transfer for CUB.** Given a target image $I_B$ (1st row), C3DM extracts the canonical embeddings $\kappa = \Phi(\mathbf{y}; I_B)$ (2nd row). Then, given the appearance descriptor $\boldsymbol{\beta}(I_A)$ of a texture image $I_A$ (4th row), the texture network $C$ transfers its style to get a styled image $I_C(\mathbf{y}) = C(\Phi_{\mathbf{y}}(I_B);\ \boldsymbol{\beta}(I_A))$ (3rd row), which preserves the geometry of the target image $I_B$.


Figure X: **Canonical mapping and texture transfer for Freiburg cars.** Given a target image $I_B$ (1st row), C3DM extracts the canonical embeddings $\kappa = \Phi(\mathbf{y}; I_B)$ (2nd row). Then, given the appearance descriptor $\boldsymbol{\beta}(I_A)$ of a texture image $I_A$ (4th row), the texture network $C$ transfers its style to get a styled image $I_C(\mathbf{y}) = C(\Phi_{\mathbf{y}}(I_B);\ \boldsymbol{\beta}(I_A))$ (3rd row), which preserves the geometry of the target image $I_B$.

[31] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Proc. CVPR*, 2015.

[32] Dong Liu Ke Sun, Bin Xiao and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proc. CVPR*, 2019.

[33] Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning Unsupervised Multi-View Stereopsis via Robust Photometric Consistency. 2019. URL `http://arxiv.org/abs/1905.02706`.

[34] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proc. CVPR*, 2019.

[35] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *Proc. ICCV*, 2019.

[36] Suryansh Kumar, Yuchao Dai, and Hongdong Li. Spatial-temporal union of subspaces for multi-body non-rigid structure-from-motion. *Pattern Recognition Journal*, 2017.

[37] Suryansh Kumar, Anoop Cherian, Yuchao Dai, and Hongdong Li. Scalable dense non-rigid structure from motion: A grassmannian perspective. In *Proc. CVPR*, 2018.

[38] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V. Gehler. Unite the people: Closing the loop between 3D and 2D human representations. In *Proc. CVPR*, 2017.

[39] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, December 2015.

[40] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black and. SMPL: A skinned multi- person linear model. *ACM Trans. on Graphics*, 2015.

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proc. CVPR*, 2019.

[42] David Novotny, Diane Larlus, and Andrea Vedaldi. Learning 3d object categories by looking around them. In *Proc. ICCV*, 2017.

[43] David Novotny, Diane Larlus, and Andrea Vedaldi. Capturing the geometry of object categories from video supervision. *PAMI*, 2018.

[44] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3DPO: Canonical 3d pose networks for non-rigid structure from motion. In *Proc. ICCV*, 2019.

[45] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehle, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *3DV*, 2018.

[46] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3D human pose and shape from a single color image. In *Proc. CVPR*, 2018.

[47] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proc. CVPR*, 2019.

[48] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *The IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009.

[49] Lawrence Gilman Roberts. *The Perception of Three-Dimensional Solids*. PhD thesis, Massachusets Institute of Technology, 1963. URL `https://dspace.mit.edu/bitstream/handle/1721.1/11589/33959125-MIT.pdf`.

[50] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. LCR-Net++: Multi-person 2D and 3D pose detection in natural images. *PAMI*, 2018.

[51] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2), 2017.

[52] Nima Sedaghat and Tomas Brox. Unsupervised generation of a viewpoint annotated car dataset from videos. In *Proc. ICCV*, 2015.

[53] Leonid Sigal, Alexandru Balan, and Michael J. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Proc. NIPS*. 2008.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[55] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-Resolution Representations for Labeling Pixels and Regions. 2019. URL `http://arxiv.org/abs/1904.04514`.

[56] V. Tan, I. Budvytis, and R. Cipolla. Indirect deep structured learning for 3D human body shape and pose prediction. In *Proc. BMVC*, 2017.

[57] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What Do Single-view 3D Reconstruction Networks Learn? In *Proc. CVPR*, 2019.

[58] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised object learning from dense invariant image labelling. In *Proc. NIPS*, 2017.

[59] J. Thewlis, S. Albanie, H. Bilen, and A. Vedaldi. Unsupervised learning of landmarks via vector exchange. In *Proc. ICCV*, 2019.

[60] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *PAMI*, 30(5):878–892, 2008.

[61] Hsiao-Yu Fish Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *Proc. NIPS*, 2017.
[62] G. Varol, D. Ceylan, B. Russel, J. Yang, E. Yumer, I. Laptev, and C. Schmid. BodyNet: Volumetric inference of 3D human body shapes. In *Proc. ECCV*, 2018.
[63] Sara Vicente, Joao Carreira, Lourdes Agapito, and Jorge Batista. Reconstructing PASCAL VOC. In *Proc. CVPR*, 2014.
[64] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.
[65] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *Proc. CVPR*, pages 2637–2646, 2019.
[66] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proc. CVPR*, 2019.
[67] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. *Proc. WACV*, 2014.
[68] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3d object detection in the wild. In *WACV*, 2014.
[69] Jing Xiao, Chai Jin-xiang, and Takeo Kanade. A closed-form solution to non-rigid shape and motion recovery. In *Proc. ECCV*, 2004.
[70] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *Proc. ECCV*, 2018.
[71] A. Zanfir, E. Marinoiu, and C. Sminchisescu. Monocular 3D pose and shape estimation of multiple people in natural scenes — the importance of multiple scene constraints. In *Proc. CVPR*, 2018.
[72] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. CVPR*, 2018.
[73] Xiaowei Zhou, Menglong Zhu, Kosta Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proc. CVPR*, 2016.
[74] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, and Kostas Daniilidis. Sparse representation for 3D shape estimation: A convex relaxation approach. *PAMI*, 2016.
[75] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the Continuity of Rotation Representations in Neural Networks. In *Proc. CVPR*, 2019. URL `http://arxiv.org/abs/1812.07035`.
[76] Yingying Zhu, Dong Huang, Fernando De La Torre, and Simon Lucey. Complex non-rigid motion 3D reconstruction by union of subspaces. In *Proc. CVPR*, 2014.