UNIVERSITY OF CALIFORNIA, MERCED

Learning Visual Correspondences across Instances and Video Frames

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering and Computer Science

by

Xueting Li

Committee in charge:

Professor Ming-Hsuan Yang, Chair
Professor Shawn Newsam
Professor Sungjin Im
Dr Sifei Liu

December 2021

The dissertation of Xueting Li is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

(Professor Shawn Newsam)

(Professor Sungjin Im)

(Dr Sifei Liu)

(Professor Ming-Hsuan Yang, Chair)

University of California, Merced

December 2021

DEDICATION

To my wonderful parents and brother.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

To begin with, I would like to express my deepest thanks to my advisor Professor Ming-Hsuan Yang. My Ph.D. journey has benefited massively from his support and advice. He has spent significant time patiently listening to my unbaked ideas and providing insightful feedback for each project. It has been a great pleasure to work with him, and I appreciate his consistent guidance and help in the past five years.

Next, I am grateful to all my mentors and co-authors, who have spent incredible time helping me with the projects and papers. I would like to express special thanks to my mentor and friend Sifei Liu. She has been supportive, patient and spent great efforts in mentoring me in the past few years. I couldn't stand where I am now without her.

I also would like to thank my friend and roommate, Jie Ren. As Ph.D. students, we have shared our happiness, joy, sorrow, and anxiety. I appreciate the memorable high-way trips and happy time spent with her.

Last but not least, I want to thank my parents and brother. I started this journey when my parents were in their 70s. Having them with me along the way is the luckiest thing that ever happened to me. I appreciate every minute spent with them, even when we are oceans apart. In addition, my big brother, who raised me as his own with my parents, has always been supportive whenever I needed him. They are and will always be my deepest love and drive. This dissertation would not be possible without them, but it is also my proof to show them that I have grown to be an independent adult with their care and love.

VITA

2009-2013	B. S. in Computer Science, Beijing University of Posts and Telecommunications
2013-2016	M. S. in Software Engineering, Tsinghua University
2017-2021	Ph. D. in Electrical Engineer and Computer Science, University of California, Merced

PUBLICATIONS

Xueting Li , Sifei Liu , Jan Kautz , Ming-Hsuan Yang, "Learning Linear Transformations for Fast Arbitrary Style Transfer", *CVPR*, 2019.

Xueting Li , Sifei Liu , Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, Jan Kautz, "Putting Humans in a Scene: Learning Affordance in 3D Indoor Environments", *CVPR*, 2019.

Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, Ming-Hsuan Yang, "Joint-task Self-supervised Learning for Temporal Correspondence", *NeurIPS*, 2019.

Xueting Li ,Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, Jan Kautz, "Self-supervised Single-view 3D Reconstruction via Semantic Consistency", *ECCV*, 2020.

Xueting Li ,Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, Jan Kautz, "Online adaptation for consistent mesh reconstruction in the wild", *NeurIPS*, 2020.

Sifei Liu, **Xueting Li** , Varun Jampani, Shalini De Mello, Jan Kautz, "Learning Propagation for Arbitrarily-structured Data.", *ICCV*, 2019.

An-Chieh Cheng, **Xueting Li**, Min Sun, Ming-Hsuan Yang, Sifei Liu,, "Learning 3D Dense Correspondence via Canonical Point Autoencoder.", *NeurIPS*, 2021.

Yijun Li , Ming-Yu Liu , **Xueting Li** , Ming-Hsuan Yang , Jan Kautz, "A Closed-form Solution to Photorealistic Image Stylization.", *ECCV*, 2018.

Xueting Li, Shalini De Mello, Xiaolong Wang, Ming-Hsuan Yang , Jan Kautz, Sifei Liu, "Learning Continuous Environment Fields via Implicit Functions", *a submission to ICLR*, 2022.

Xueting Li, Sifei Liu, Xiaolong Wang, Ming-Hsuan Yang , Alyosha Efros, "Scraping Textures from Natural Images for Synthesis and Editing.", *a submission to CVPR*, 2022.

ABSTRACT OF THE DISSERTATION

Learning Visual Correspondences across Instances and Video Frames

by

Xueting Li

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California Merced, December 2021

Professor Ming-Hsuan Yang, Chair

Correspondence is ubiquitous in our visual world. It describes the relationship of two images by pointing out which parts in one image relate to which parts in the other image. It is the fundamental task in many computer vision applications. For instance, object tracking essentially studies the correspondence of different parts on the same object through time, while semantic segmentation links the same semantic parts of different objects through space. Furthermore, the study of correspondence facilitates many applications such as structure from motion or label propagation through video frames. However, correspondence annotation is notoriously hard to harvest. Existing work either utilize synthesized data (e.g., optical flow from a game engine) or other human annotations (e.g., semantic segmentation), leading to domain limitation or tedious human efforts.

My research focuses on learning and applying correspondence in computer vision tasks in a self-supervised manner to resolve these limitations. I start by introducing a method that learns reliable dense correspondence from videos in a self-supervised manner. Next, I discuss two methods that utilize correspondence between images or video frames to facilitate 3D mesh reconstruction. To begin with, I present a work that learns a self-supervised, single-view 3D reconstruction model that predicts the 3D mesh shape, texture, and camera pose of a target object with a collection of 2D images and silhouettes. Then, based on the two methods discussed above, the intuitive question is that can we combine the correspondence learned in the first work and the mesh reconstruction model in the second work to solve mesh reconstruction from video frames? Thus, in the last work, I demonstrate an algorithm to reconstruct temporally consistent 3D meshes of deformable object instances from videos in the wild.

Chapter 1

Introduction

1.1 What is correspondence?

Visual correspondence is a fundamental concept in computer vision. It describes relationships between different images by determining the parts that relate to each other in the images. Correspondence exists ubiquitously in computer vision tasks. For instance, in image classification [36, 59], objects assigned to the same class (e.g., all humans) are correspondences. Or in pose estimation [88, 113], the same joint of different objects in the same category (e.g., hands of all humans) are correspondences. Specifically, in this thesis, we consider two types of correspondences – temporal and semantic correspondences.

Temporal correspondences [69, 131, 107, 21, 43] describe the movement and appearance changes of the same object at different frames in a video. Thanks to the continuity of our visual world (i.e., our visual world seldom changes abruptly), the same objects such as the bear claws in Figure 1.1(a), usually appear repeatedly in adjacent video frames, with slight movement and changes in appearance. In this thesis, we introduce a method [69] that learns such temporal correspondences from large-scale unlabeled videos as well as its application in helping mesh reconstruction from video frames [67].

Semantic correspondences [42, 76, 146] refer to the corresponding semantic parts in different objects in the same category. Taking the bird images in Figure 1.1(b) as an example, the wings of different birds are semantic correspondences. Such semantic correspondences provide abundant knowledge about the underline 3D shapes of objects in the images and we show how to utilize them for unsupervised single-view mesh reconstruction [68].

1.2 Self-supervised Correspondence Learning and Application

As a cornerstone task in computer vision, correspondence learning has been well studied in the past decades [131, 69, 42, 76]. However, annotating correspondences between images requires tedious human efforts, especially for pixel-wise dense correspondence labeling (e.g., optical flow). Existing work bypass this limitation either

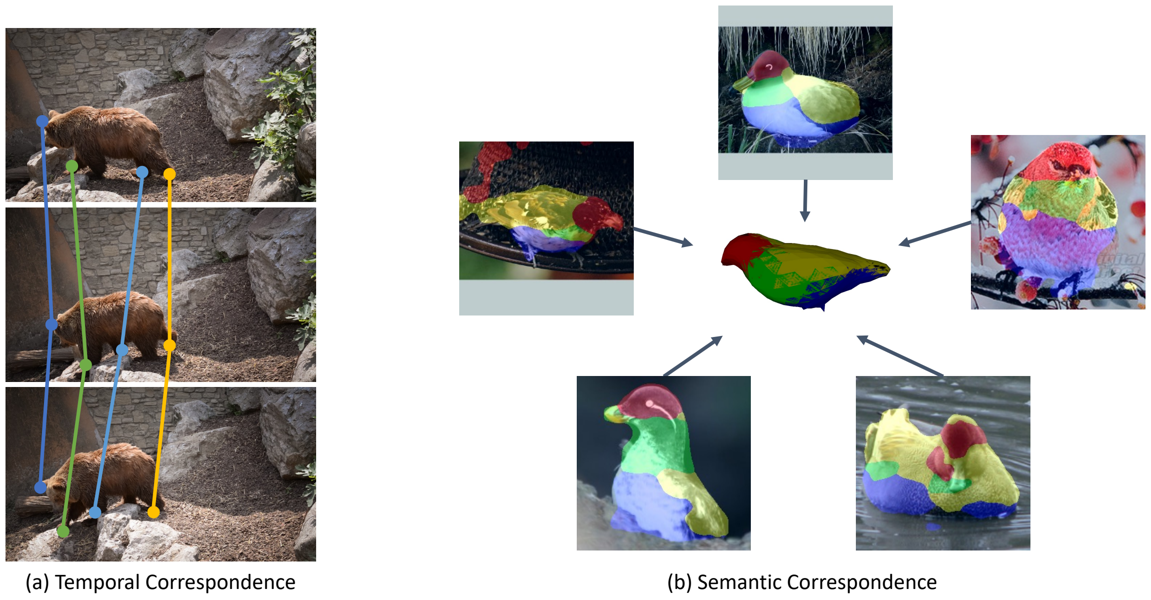


Figure 1.1: (a) Temporal correspondence. Image credit: The DAVIS-2017 dataset [95]. (b) Semantic correspondence. Image credit: The SCOPS method [42].

by harvesting correspondence annotations from game engines [6], or utilizing indirect annotations such as semantic segmentation [72, 30] as supervisions. Yet, the former presents a large domain gap when applied to real-world images, and the latter still requires a large number of labeling efforts. To resolve these limitations, my research focuses on learning and utilizing correspondences in a self-supervised manner.

Self-supervised methods have spurred great interest in the computer vision community due to their data efficiency [35, 11, 13]. The key and the difficulty for such methods lie in developing self-supervised signals and constraints for network training. This dissertation demonstrates how to learn temporal correspondences from unlabeled videos using color as the self-suerpvision signal. I also introduce how to utilize temporal and semantic correspondences to facilitate unsupervised 3D mesh reconstruction for images and videos.

1.3 Contributions

The contributions of this dissertation include:

- We propose a self-supervised method [69] to find accurate correspondences at different levels across video frames, which outperforms state-of-the-art methods on a variety of visual correspondence tasks, e.g., video instance and part segmentation, keypoints tracking, and object tracking.

- We introduce a self-supervised, single-view 3D reconstruction model that predicts the 3D mesh shape, texture, and camera pose of a target object with a collection of 2D images and silhouettes, by utilizing semantic correspondences across images [68]. Through various experiments on several categories of deformable and rigid objects, we demonstrate that our unsupervised method performs comparably if not better than existing category-specific reconstruction methods learned with supervision.

- We present an algorithm to reconstruct temporally consistent 3D meshes of deformable object instances from videos in the wild without requiring annotations of 3D mesh, 2D keypoints, or camera pose for each video frame by using temporal correspondences [67]. We demonstrate that our algorithm recovers temporally consistent and reliable 3D structures from videos of non-rigid objects, including those of animals captured in the wild – an extremely challenging task rarely addressed before.

1.4 Dissertation Overview

The rest of the text is organized as follows: In Chapter 2, we introduce a method that learns temporal correspondence from large-scale unlabeled videos. Next in Chapter 3, we apply semantic correspondence in unsupervised mesh reconstruction from images. We further demonstrate how to use learned temporal correspondence to reconstruct consistent shapes from videos in Chapter 4. Finally, we conclude this dissertation and discuss future work in Chapter 5.

Chapter 2

Self-supervised Temporal Correspondence Learning

2.1 Overview

In this chapter, we propose to learn reliable dense correspondence from videos in a self-supervised manner. Our learning process integrates two highly related tasks: tracking large image regions *and* establishing fine-grained pixel-level associations between consecutive video frames. We exploit the synergy between both tasks through a shared inter-frame affinity matrix, which simultaneously models transitions between video frames at both the region- and pixel-levels. While region-level localization helps reduce ambiguities in fine-grained matching by narrowing down search regions, fine-grained matching provides bottom-up features to facilitate region-level localization. Our method outperforms the state-of-the-art self-supervised methods on a variety of visual correspondence tasks, including video-object and part-segmentation propagation, keypoint tracking, and object tracking. Our self-supervised method even surpasses the fully-supervised affinity feature representation obtained from a ResNet-18 pre-trained on the ImageNet.

2.2 Introduction

Learning representations for visual correspondence is a fundamental problem that is closely related to a variety of vision tasks: correspondences between multi-view images relate 2D and 3D representations, and those between frames link static images to dynamic scenes. To learn correspondences across frames in a video, numerous methods have been developed from two perspectives: (a) learning region/object-level correspondences, via object tracking [3, 110, 118, 98, 128] or (b) learning pixel-level correspondences between multi-view images or frames, e.g., via stereo matching [87] or optical flow estimation [73, 107, 43, 79].

However, most methods address one or the other problem and significantly less effort has been made to solve both of them together. The main reason is that methods designed to address either of them optimize different goals. Object tracking focuses on learning object representations that are invariant to viewpoint and deformation changes, while learning pixel-level correspondence focuses on modeling detailed changes within an object over time. Subsequently, the existing supervised methods for these two problems often use different annotations. For example, bounding boxes are annotated in real videos for object tracking [137]; and pixel-wise associations are generated from synthesized data for optical flow estimation [6, 21]. Datasets with annotations for both tasks are scarcely available and supervision, here, is a further bottleneck preventing us from connecting the two tasks.

In this chapter, we demonstrate that these two tasks inherently require the same operation of learning an inter-frame transformation that associates the contents of two images. We show that the two tasks benefit greatly by modeling them jointly via a single transformation operation which can simultaneously match regions and pixels. To overcome the lack of data with annotations for both tasks we exploit self-supervision via the signals of (a) *Temporal Coherency*, which states that objects or scenes move smoothly and gradually over time; (b) *Cycle Consistency*, correct correspondences should ensure that pixels or regions match bi-directionally and (c) *Energy Preservation*, which preserves the energy of feature representations during transformations. Since all these supervisory signals naturally exist in videos and are task-agnostic, the transformation that we learn through them can generalize well to any video without restriction on domain or object category.

Our key idea is to learn *a single* affinity matrix for modeling *all* inter-frame transformations through a network that learns appropriate feature representations that model the affinity. We show that region localization and fine-grained matching can be carried out by sharing the affinity in a fully differentiable manner: the region localization module finds a pair of patches with matching parts in the two frames (Figure 2.1, mid-top), and the fine-grained module reconstructs the color feature by transforming it between the patches (Figure 2.1, mid-bottom), all through the same affinity matrix. These two tasks symbiotically facilitate each other: the fine-grained matching module learns better feature representations that lead to an improved affinity matrix, which in turn generates better localization that reduces the search space and ambiguities for fine-grained matching (Figure 2.1, right).

The contributions of this work are summarized as: (a) A joint-task self-supervision network is introduced to find accurate correspondences at different levels across video frames. (b) A general inter-frame transformation is proposed to support both tasks and to satisfy various video constraints – coherency, cycle, and energy consistency. (c)

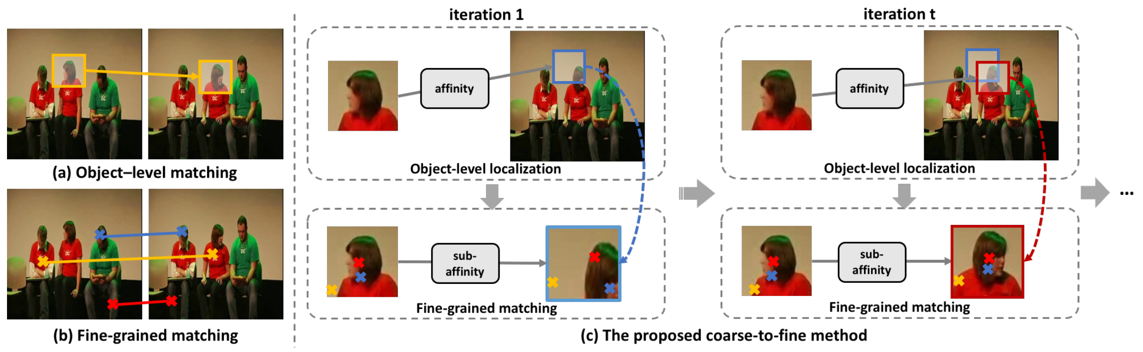

(a) Object–level matching
(b) Fine-grained matching
iteration 1
affinity
Object-level localization
sub-affinity
Fine-grained matching
iteration t
affinity
Object-level localization
sub-affinity
Fine-grained matching
(c) The proposed coarse-to-fine method

Figure 2.1: Our method (c) compared against (a) region-level matching (e.g., object tracking), and (b) pixel-level matching, e.g., matching by colorization [120]. We propose a joint-task framework which conducts region-level and fine-grained matching simultaneously and which are supported by a single inter-frame affinity matrix A. During training, the two tasks improve each other progressively. To illustrate this, we unroll two training iterations and illustrate the improvement with the red box and arrow.

Our method outperforms state-of-the-art methods on a variety of visual correspondence tasks, e.g., video instance and part segmentation, keypoints tracking, and object tracking. Our self-supervised method even surpasses the fully-supervised affinity feature representation obtained from a ResNet-18 pre-trained on the ImageNet [18].

2.3 Related Work

Learning correspondence in time is widely explored in visual tracking [3, 110, 118, 98, 128] and optical flow estimation [107, 73, 43]. Existing models are mainly trained on large annotated datasets, which require significant efforts. To overcome the limit of annotations, numerous methods have been developed to learn correspondences in a self-supervised manner [126, 131, 120]. Our work focuses on learning correspondence with self-supervision, and we discuss the most related methods here.

Object-level correspondence. The goal of visual tracking is to determine a bounding box in each frame based on an annotated box in the reference image. Most methods belong to one of the two categories that use: (a) the *tracking-by-detection* framework [1, 48, 124, 65], which models tracking as detection applied independently to individual frames; or (b) the *tracking-by-matching* framework that models cross-frame relations and includes several early attempts, e.g., mean-shift trackers [17, 143], kernelized correlation filters (KCF) [39, 70], and several works that model correlation filters as differentiable blocks [81, 82, 14, 127]. Most of these methods use annotated bounding boxes [137] in every frame of the videos to learn feature representations for tracking. Our work can be viewed as exploiting the *tracking-by-matching* framework in a self-supervised manner.

Fine-grained correspondence. Dense correspondence between video frames has been widely applied for optical flow and motion estimation [79, 107, 73, 43], where the goal is to track individual pixels. Most deep neural networks [43, 107] are trained with the objective of regressing the ground-truth optical flow produced by synthetic datasets [6, 21]. In contrast to many classic methods [79, 73] that model dense correspondence as a matching problem, direct regression of pixel offsets has limited capability for frames containing dramatic appearance changes [5, 104], and suffers from problems related to domain shift when applied to real-world scenarios.

Self-supervised learning. Recently, numerous approaches have been developed for correspondence learning via various self-supervised signals, including image [64] or color transformation [120] and cycle-consistency [131, 126]. Self-supervised learning of correspondence in videos has been explored along the two different directions – for region-level localization [131, 126] and for fine-grained pixel-level matching [120, 58]. In [126], a correlation filter is learned to track regions via a cycle-consistency constraint, and no pixel-level correspondence is determined. [131] develops patch-level tracking by modeling the similarity transformation of pixels within a fixed rectangular region. Conversely, several methods learn a matching network by transforming color/RGB information between adjacent frames [120, 63, 58]. As no region-level regularization is exploited, these approaches are less effective when color features are less distinctive (see Figure 2.1(b)). In contrast, our method learns object-level and pixel-level correspondence jointly across video frames in a self-supervised manner.

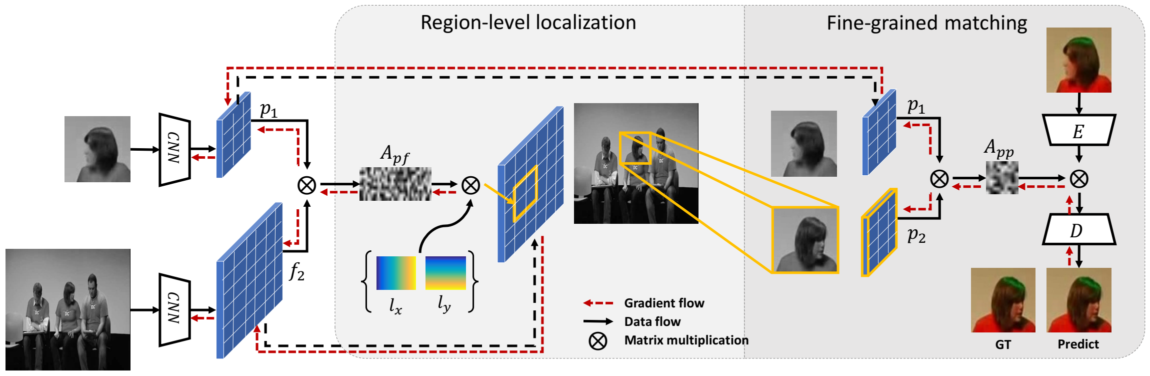


Figure 2.2: Main steps of proposed method. Blue grids represent the reference-patch p_1's and target-frame f_2's feature maps that are shared by the region-level localization (left box) and fine-grained matching (right box) modules. A_{pf} is the affinity between p_1 and f_2, and A_{pp} is that between p_1 and p_2. p_2 is a differentiable crop from the frame f_2. The maps l_x and l_y are the coordinates of pixels on a regular grid. All modules are differentiable, where the gradient flow is visualized via the red dashed arrows.

2.4 Approach

Video frames are temporally coherent in nature. For a pair of adjacent frames, pixels in a later frame can be considered as being copied from some locations of an earlier one with slight appearance changes conforming to object motion. This "copy" operator can be expressed via a linear transformation with a matrix A, in which $A_{ij} = 1$ denotes that the pixel j in the second frame is copied from pixel i in the first one. An approximation of A is the inter-frame affinity matrix [118, 77, 131]:

$$A_{ij} = \kappa(f_{1i}, f_{2j}) \tag{2.1}$$

where κ denotes some similarity function. Each entry A_{ij} represents the similarity of subspace pixels i and j in the two frames $f_1 \in \mathcal{R}^{C\times N_1}$ and $f_2 \in \mathcal{R}^{C\times N_2}$, where $f \in \mathcal{R}^{C\times N}$ is a vectorized feature map with C channels and N pixels. In this work, our goal is to learn the feature embedding f that optimally associates the contents of the two frames.

One free supervisory signal that we can utilize is color. To learn the inter-frame transformation in a self-supervised manner, we can slightly modify (2.1) to generate the affinity via features f learned only from gray-scale images. The learned affinity is then utilized to map the color channels from one frame to another [120, 77], while using the ground-truth color as the self-supervisory signal.

One strict assumption of this formulation is that the paired frames need to have the same contents – no new object or scene pixel should emerge over time. Hence, the existing methods [120, 77] sample pairs of frames either uniformly, or randomly within a specified interval, e.g., 50 frames. However, it is difficult to determine a "perfect" interval as video contents may change sporadically. When transforming color from a reference frame to a target one, the objects/scene pixels in the target frame may not exist in the reference frame, thereby leading to wrong matches and an adverse effect on feature learning. Another issue is that a large portion of the video frames are "static", in which the sampled pair of frames are almost the same and cause the learned affinity to be an identity matrix.

We show that the above problems can be addressed by incorporating a region-level localization module. Given a pair of reference and target frames, we first randomly sample a patch in the reference frame and localize this patch in the target frame (see Figure 2.2). The inter-frame color transformation is then estimated between the paired patches. Both localization and color transformation are supported by a single affinity derived from a convolutional neural network (CNN) based on the fact that the affinity matrix can simultaneously track locations and transform features discussed in this section.

2.4.1 Transforming Feature and Location via Affinity

We sample a pair of frames and denote the 1^{st} frame as the reference and the 2^{nd} one as the target. The CNN can be any effective model, e.g., ResNet-18 [36] with the first 4 blocks that takes a gray-scale image as input. We compute the affinity and conduct the feature transformation and localization on the top layer of the CNN, with features that are one-eighth the size of the input image. This ensures the affinity matrix to be memory efficient and each pixel in the feature space to contain considerable local contextual information.

Transforming feature representations. We adopt the dot product for κ in (2.1) to compute the affinity, where each column can be interpreted as the similarity score between a point in the target frame to all points in the reference frame. For dense correspondence, the inter-frame affinity needs to be sparse to ensure one-to-one mapping. However, it is challenging to model a sparse matrix in a deep neural network. We relax this constraint and encourage the affinity matrix to be sparse by normalizing each column with the softmax function, so that the similarity score distribution can be peaky and only a few pixels with high similarity in the reference frame are matched to each point in the target frame:

$$A_{ij} = \frac{\exp(f_{1i}^{\top} f_{2j})}{\sum_k \exp(f_{1k}^{\top} f_{2j})}, \quad \forall i \in [1, N_1], j \in [1, N_2] \tag{2.2}$$

where the variable definitions follow (2.1). The transformation is carried out as $\hat{c_2} = c_1 A$, where $A \in \mathcal{R}^{N_1 \times N_2}$, and c_i has the same number of entries as f_i and can be features of the reference frame or any associated label, e.g., color, segmentation mask or keypoint heatmap.

Tracing pixel locations. We denote $l_j = (x_j, y_j), l \in \mathcal{R}^{2\times N}$ as the vectorized location map for an image/feature with N pixels. Given a sparse affinity matrix, the location of an individual pixel can be traced from a reference frame to an adjacent target frame:

$$l_j^{12} = \sum_{k=1}^{N_1} l_k^{11} A_{kj}, \quad \forall j \in [1, N_2] \tag{2.3}$$

where l_j^{mn} represents the coordinate in frame m that transits to the j^{th} pixel in frame n. Note that l^{nn} (e.g., l^{11} in (2.3)) usually represents a canonical grid as shown in Figure 2.3.

2.4.2 Region-level Localization

In the target frame, region-level localization aims to localize a patch randomly selected from the reference frame by predicting a bounding box (denoted as "bbox") on a region that shares matching parts with the selected patch. In other words, it is a differential region of interest (ROI) with learnable center and scale. We compute an $N_1 \times N_2$ affinity A_{pf} according to (2.2) between feature representations of the patch in the reference frame, and that of the whole target frame (see Figure 2.2(a)).

Locating the center. To track the center position of the reference patch in the target frame, we first localize each individual pixel of the reference patch p_1 in the target frame f_2, according to (2.3). As we obtain the set l^{21}, with the same number of entries as p_1, that collects the coordinates of the most similar pixels in f_2, we can compute the average coordinate $C^{21} = \frac{1}{N_1}\sum_{i=1}^{N_1} l_i^{21}$ of all the points, as the estimated new position of the reference patch.

Scale modeling. For region-level tracking, the reference patch may undergo significant scale changes. Scale estimation in object tracking is challenging and existing methods mainly enumerate possible scales [3, 126] and select the optimal one. In contrast, the scale can be estimated by our proposed model. We assume that the transformed locations l^{21} are still distributed uniformly in a local rectangular region. By denoting w as the width of the new bounding box, the scale is estimated by:

$$\hat{w} = \frac{2}{N_1}\sum_{i=1}^{N_1} \left\| x_i - C^{21}(x) \right\|_1 \tag{2.4}$$

where the x_i is the x-coordinate of the i^{th} entry in the l^{21}. We note that (2.4) can be proved by using the analogous continuous space. Suppose there is a rectangle with scale $(2w, 2h)$ and with its center located at the origin of a 2D coordinate plane. By integrating points inside of it, we have:

$$\frac{1}{w}\int_{-w}^{w} \|x\|_1 \, dx = \frac{2}{w}\int_{0}^{w} x dx = w \tag{2.5}$$

This represents the average absolute distances w.r.t. the center when transforming to the discrete space. The estimation of height is conducted in the same manner.

Moving as a unit. An important assumption in the aforementioned ROI estimation in the target frame is that the pixels from the reference patch should move in unison – this is true in most videos, as an object or its parts typically move as one unit at the region level. We enforce this constraint with a *concentration* regularization [149, 42] term on the transformed pixels, with a truncated loss to penalize these points from moving too far away from the center:

$$L_c = \begin{cases} 0, & \left\| l_j^{12}(x) - C^{12}(x) \right\|_1 \le w \text{ and } \left\| l_j^{12}(y) - C^{12}(y) \right\|_1 \le h \\ \frac{1}{N_2} \sum_{j=1}^{N_2} \left\| l_j^{12} - C^{12} \right\|_2, & \text{otherwise} \end{cases} \tag{2.6}$$

This formulation encourages all the tracked pixels, originally from a patch, to be concentrated (see Figure 2.3) rather than being dispersed to other objects, which is likely to happen for methods that are based on pixel-wise matching only, e.g., when matching by color reconstruction, pixels of different objects having similar colors may match each other, as shown in Figure 2.1(b).

2.4.3 Fine-grained Matching

Fine-grained matching aims to reconstruct the color information of the located patch in the target frame, given the reference patch (see Figure 2.1). We re-use the inter-frame affinity A_{pf} by extracting a sub-affinity matrix A_{pp} containing the columns corresponding to the located pixels in the target frame, and by using it for the color transformation described in the formulations in Section 2.4.1. To make the color feature compatible with the affinity matrix, we train an auto-encoder that learns to reconstruct an image in the Lab space faithfully (see the encoder E and the decoder D in Figure 2.2). This network also encodes global contextual information from the color channels. We show that using the color feature instead of pixels significantly reduces the errors caused by reconstructing color directly in the image space [120] (see Table 2.1, ours vs. [120]). In the following, we introduce self-supervisory signals as regularization for fine-grained matching. For brevity, we denote A as the sub-affinity, l and f as the vectorized coordinate and feature map, respectively, for the paired patches.

Orthogonal regularization. Another important constraint, *cycle-consistency*, for the transformation of both location [131] and feature [77] is the orthogonal regularization. For a pair of patches, we encourage every pixel to fall into the same location after one cycle of forward and backward tracking, as shown in Figure 2.3 (middle and right):

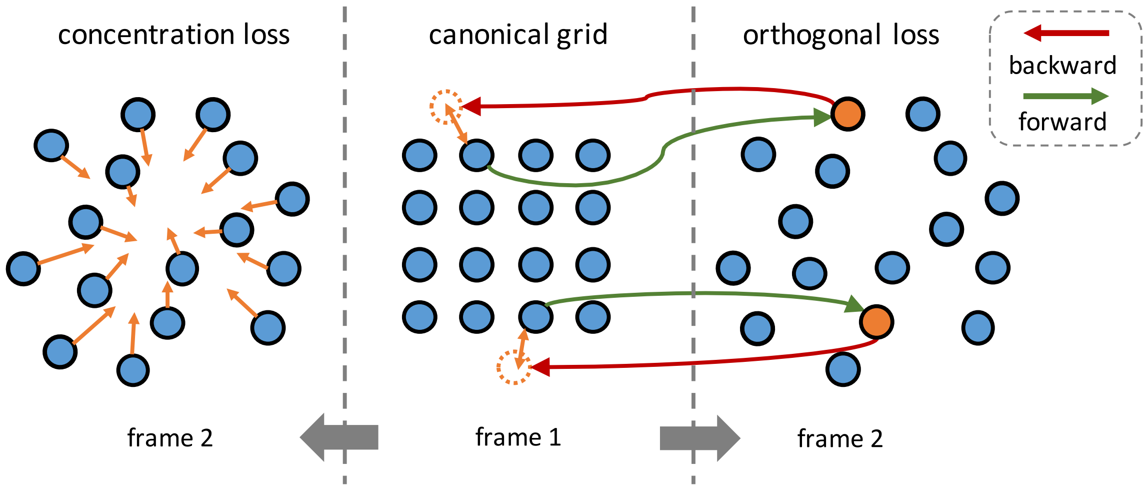


Figure 2.3: Concentration (left) and orthogonal (right) regularization. The dots denote pixels in feature space. The orange arrows show how they push the pixels.

$$\hat{l^{12}} = l^{11} A_{1\to 2}, \quad \hat{l^{11}} = \hat{l^{12}} A_{2\to 1} \tag{2.7}$$

Here we specifically add $m \to n$ to denote affinity transforming from the frame m to n, i.e., $A_{m\to n} = \kappa(f_m, f_n)$. Similarly, the *cycle-consistency* can be applied to the feature space:

$$\hat{f}_2 = f_1 A_{1\to 2}, \quad \hat{f}_1 = \hat{f}_2 A_{2\to 1} \tag{2.8}$$

We show that enforcing *cycle-consistency is equivalent to regularizing A to be orthogonal*: With (2.7) and (2.8), it is easy to show that the optimal solution is achieved when $A^{-1}_{1\to 2} = A_{2\to 1}$. Inspired by recent style transfer methods [26, 77], the color energy represented by the Gram-matrix should be consistent such that $f_1 f_1^\top = f_2 f_2^\top$, which derives that $A^\top_{1\to 2} = A_{2\to 1}$ is the goal to reconstruct the color information. Thus, it is easy to show that regularizing A as orthogonal automatically satisfies the cycle constraint. In practice, we switch the role of reference and target to perform the transformation, as described in (2.7) and (2.8). We use the MSE loss between both $\hat{l^{11}}$ and l^{11}, $\hat{f}_1$ and f_1, and specifically replace $A_{2\to 1}$ with $A^\top_{1\to 2}$ in Eq. (2.8) to enforce the regularization. Namely, the orthogonal regularization provides a concise mathematical formulation for many recent works [131, 126] that exploit *cycle-consistency* in videos.

Concentration regularization. We additionally apply the concentration loss (i.e., Eq.(2.6) without the truncation) in local, non-overlapping 8×8 grids of a feature map, to encourage local context or object parts to move as an entity over time. Unlike [131, 103] where local patches are regularized by similarity transformation via a spatial transformation network [46], this local concentration loss is more flexible by allowing arbitrary deformations within each local grid.

2.5 Experiments

We compare with state-of-the-art algorithms [120, 126, 131] on several tasks: instance mask propagation, pose keypoints tracking, human parts segmentation propagation and visual tracking.

2.5.1 Network Architecture

As shown in Figure 2.2, our model consists of a region-level localization module and a fine-grained matching module that share a feature representation network (see Figure 2.2). We use the ResNet-18 [36] as the network for fair comparisons with [120, 131]. The patch randomly cropped from the reference frame is of 256×256 pixels. We carry out all our experiments on servers equipped with four 16GB Tesla V100 GPUs.

Training. We first train the auto-encoder in the matching module (the encoder "E" and decoder "D" in Figure 2.2) to reconstruct images in the Lab space using the MSCOCO [72] dataset. We then fix it and train the feature representation network using the Kinetics dataset [54]. For all experiments, we train our model from scratch without any level of pre-training or human annotations. The objectives include: (a) concentration loss (Section 2.4.2 and 2.4.3), (b) color reconstruction loss and (c) orthogonal regularization (Section 2.4.3). Involving the localization module from the beginning in the training process prevents the network from converging because poor localization makes matching impossible. Thus we first train our network using patches cropped at the same location with the same size in the reference and target frame respectively. Fine-grained matching is conducted between the two patches for

10 epochs. We then jointly train the localization and matching module for another 10 epochs. We train our model using Adam [56] as the optimizer with a learning rate of 10^{-4} for the warm-up and 0.5×10^{-4} for the joint training of the localization and matching modules. We set the temperature in the softmax layer applied to the affinity matrix to 1 which empirically achieves best performance.

Inference. In the inference stage, we directly apply the affinity learned to transform color feature representations, on different types of inputs, e.g., segmentation masks and keypoint maps. We use the same testing protocol as Wang et al. [131] for all tasks. Similar to [131], we adopt a recurrent inference strategy by propagating the ground truth segmentation mask or keypoint heatmap from the first frame, as well as the predicted results from the preceding n frames onto the target frame. We average all $n+1$ predictions to obtain the final propagated map (n is 1 for the VIP, and 7 for all the other tasks). For fair comparisons, we also use the k-NN propagation schema as Wang et al. [131] and set $k = 5$ for all tasks. To compare with the ResNet-18 trained on the ImageNet with classification labels, we replace our learned network weights with it and leave other settings unchanged for fair comparisons.

2.5.2 Instance Segmentation Propagation on the DAVIS-2017 dataset

Figure 2.4 (a), Figure 2.5 and Figure 2.6 show the propagated instance masks. Table 2.1 lists quantitative results of all evaluated methods based on the Jacaard index $\mathcal{J}$ (IOU) and contour-based accuracy $\mathcal{F}$. We use the full 480p images during inference for our model. For fair comparisons we test the model by Wang et al. [131] with the resolution of 480p, in addition to the result reported using 400×400 images. Our model performs favorably against the self-supervised state-of-the-art methods. Specifically, our model outperforms Wang et al. [131] by 13.3% in $\mathcal{J}$ and 16.6% in $\mathcal{F}$. and is even 6.9% better in $\mathcal{J}$ and 4.1% better in $\mathcal{F}$ than the ResNet-18 model [36] trained on ImageNet [18] with classification labels.

Furthermore, we demonstrate that by including the localization module during inference, our model can exclude noise from background pixels. In Figure 2.7, we

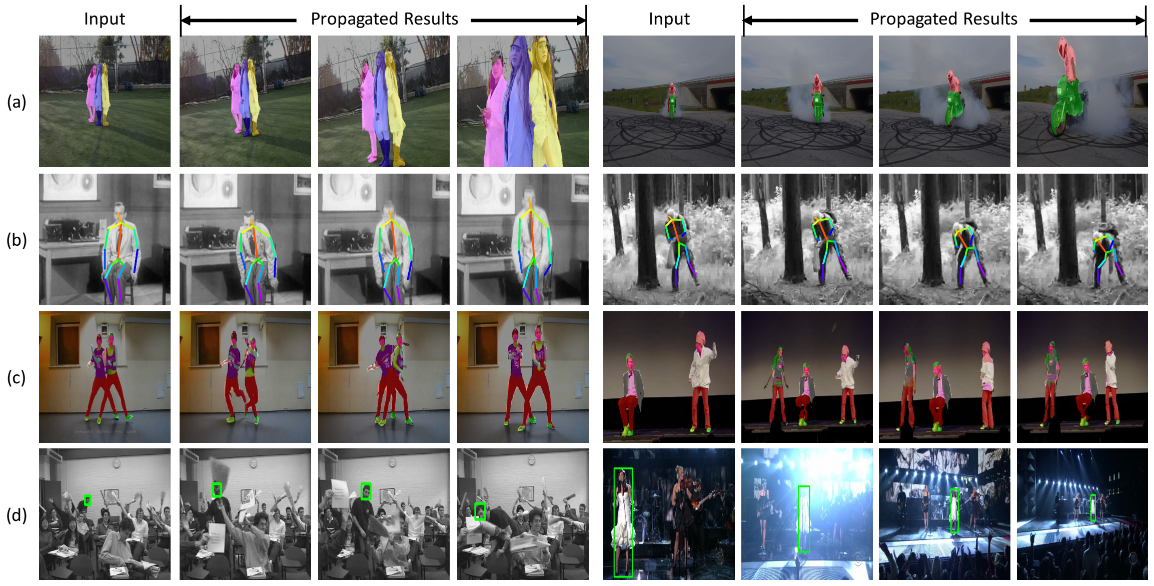


Figure 2.4: Visualization of the propagation results. (a) Instance mask propagation on the DAVIS-2017 [95] dataset. (b) Pose keypoints propagation on the J-HMDB [47] dataset. (c) Parts segmentation propagation on the VIP [150] dataset. (d) Visual tracking on the OTB2015 [137] dataset.

visualize the process of including the localization module during inference. Given the instance mask of the first frame, we first propagate each point (marked as green) from the reference frame to the target frame by localizing a bbox on it before matching. Instead of directly applying the center, we refine the center at inference by applying the mean-shift algorithm, i.e.,

$$C_t = \frac{\sum_{i=1}^{N} K(l_i - C_{t-1}) l_i}{\sum_{i=1}^{N} K(l_i - C_{t-1})} \tag{2.9}$$

where l_i is the coordinate of the i^{th} pixel, the C is the center of all l_i at the t^{th} iteration, and $K(a-b) = e^{\|a-b\|^2}$. Scale is estimated via Eq. 2.4 as well, see the bboxes in Figure 2.7. The green points in Figure 2.7 illustrate the individually propagated points and the red bounding box indicates the estimated bounding box of an object in the target frame. We then propagate the instance segmentation mask within the bounding box in the reference frame to the bounding box in the target frame. Since the propagation is carried out within two bounding boxes instead of the entire frames, we can minimize noise introduced by background pixels as shown in Figure 2.5 (d) and (e).

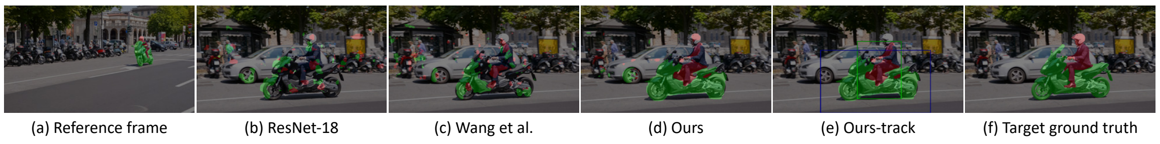


Figure 2.5: Qualitative comparison with other methods. (a) Reference frame with instance masks. (b) Results by the ResNet-18 trained on ImageNet. (c) Results by Wang et al. [131]. (d) Ours (global matching). (e) Ours with localization during inference. (f) Target frame with ground truth instance masks.

Table 2.1: Evaluation of instance segmentation propagation on the DAVIS-2017 dataset [95].

Model	Supervised	Dataset	$\mathcal{J}$(Mean)	$\mathcal{J}$(Recall)	$\mathcal{F}$(Mean)	$\mathcal{F}$(Recall)
SIFT Flow [73]	×	-	33.0	-	35.0	-
DeepCluster [8]	×	YFCC100M [112]	37.5	-	33.2	-
Transitive Inv [130]	×	-	32.0	-	26.8	-
Vondrick et al. [120]	×	Kinetics [54]	34.6	34.1	32.7	26.8
Wang et al. [131] (400 × 400)	×	VLOG [25]	43.0	43.7	42.6	41.3
Wang et al. [131] (480p)	×	VLOG [25]	46.4	50.1	50.0	48.0
mgPFF [58]	×	-	42.2	41.8	46.9	44.4
Lai et al. [63]	×	Kinetics [54]	47.7	-	51.3	-
ours	×	Kinetics [54]	56.8	65.7	59.5	65.1
ours-track	×	Kinetics [54]	**57.7**	**67.1**	**60.0**	**65.7**
ResNet-18(3 blocks)	✓	ImageNet [18]	49.4	52.9	55.1	56.6
ResNet-18(4 blocks)	✓	ImageNet [18]	40.2	36.1	42.5	36.6
FlowNet2 [43]	✓	FlyingThings3D [83]	26.7	-	25.2	-
PWC-Net [107]	✓	FlyingThings3D [83]	35.2	34.0	37.4	33.1
SiamMask [129]	✓	YouTube-VOS [141]	54.3	62.8	58.5	67.5
OSVOS [7]	✓	ImageNet,DAVIS [95]	56.6	63.8	63.9	73.8

The quantitative evaluation of this improved model outperforms the model that does not include the localization module during inference. (see “Ours-track” vs. “Ours” in Table 2.1)

2.5.3 Ablation Studies on the DAVIS-2017 Dataset

We carry out ablation studies to see the contributions of each term, as shown in Figure 2.8 and Table 2.2. Note that inference is conducted between a pair of full-size frames without localization.

Region-level Localization. Our model trained with the region-level localization module is able to place the individual points all within a reasonable local region (Figure 2.8 (c)). We show that the model can accurately capture both region-level shifts (e.g., person moving forward), and subtle deformations (e.g., movement of body

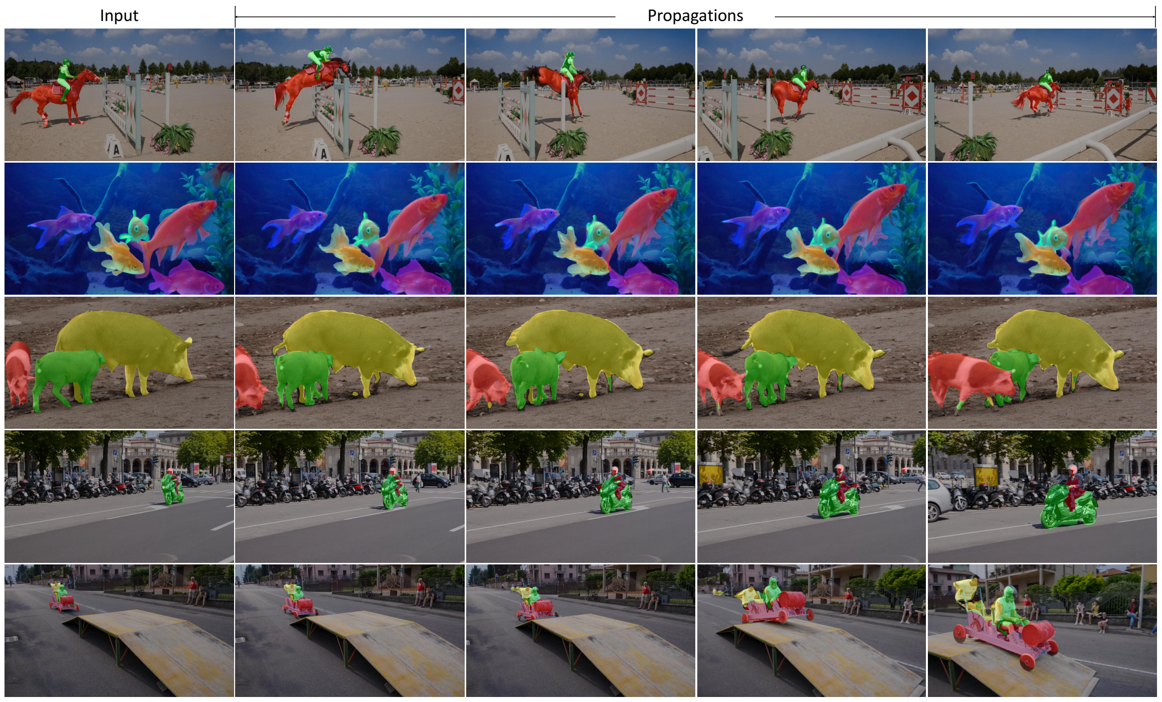


Figure 2.6: Instance mask propagation results.

parts), while preserving the correct spatial relations among all the points. In contrast, the model trained without the localization module tends to model global matching, leading to less accurate preservation of the local spatial relationships among points, e.g., the red points in Figure 2.8 (d) tend to cluster together as shown in the cyan circle. Consistent quantitative results can also be found in Table 2.2 (c), where the $\mathcal{J}$ and $\mathcal{F}$ measures drop 2.5% and 0.9%, respectively, when trained without the localization module. We also discover that the localization module should always be trained together with the concentration loss to satisfy the assumption in Section 2.4.2(Table 2.2(f)(g)).

Concentration regularization. The concentration regularization encourages locality during the transformation process, i.e. points within a neighbourhood in the reference frame stay together in the target frame. The model trained without it tends to introduce outliers, as shown in the cyan circle of Figure 2.8(e). Table 2.2 (b)(e) demonstrate the contribution of this concentration regularization term, e.g., compared to (b), the $\mathcal{J}$ in (e) decrease by 8% without this regularization term.

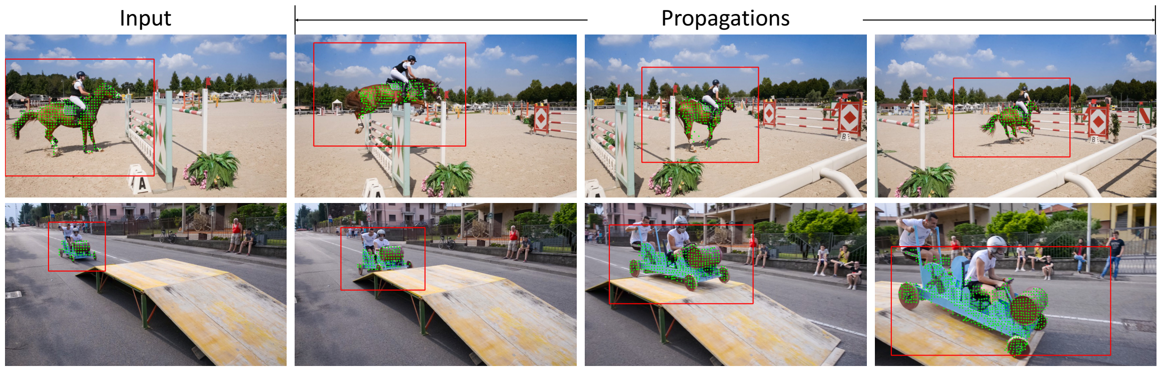


Figure 2.7: Visualization of the process of including the localization module during inference.

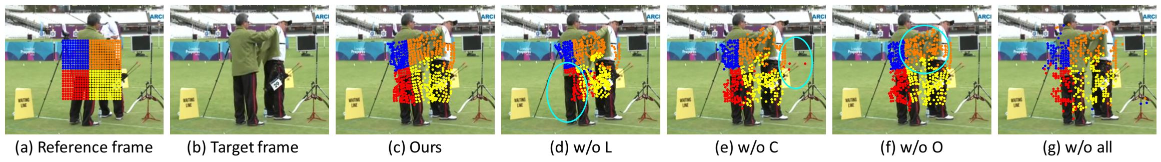


Figure 2.8: Visualization of the ablation studies. Given a set of points in the reference frame (a), we visualize the results of propagating these points on to the target frame (b). “L”, “C”, “O” and “all” correspond to the localization modules, concentration or orthogonal regularization, or all of them (d-g).

Orthogonal regularization. The orthogonal regularization term enforces points to match back to themselves after a cycle of forward and backward transformation. As shown in Figure 2.8 (f), the model trained without the orthogonal regularization term is less effective in preserving local structures. The effectiveness of the orthogonal regularization is also validated quantitatively at Table 2.2 (e) and (f).

Table 2.2: Ablation studies. The minus sign “-” indicates training without the specific module or regularization. “L”, “O” and “C” mean the localization module, orthogonal and concentration regularization, respectively. The last column (“(g) -all”) shows results of a baseline model trained without any of “L”, “O” or “C”.

Metric	(a) Ours-track	(b) Ours	(c) -L	(d) -O	(e) -C	(f) -O&C	(g) -all
$\mathcal{J}$ (Mean)	57.7	56.3	53.8	55.2	48.3	44.3	45.7
$\mathcal{F}$ (Mean)	61.3	59.2	58.3	58.7	52.4	49.6	52.3

Table 2.3: Tracking results on OTB2015 [137]

Model	Supervised	AUC score (%)
UDT [126]	×	59.4
Ours	×	59.2
ResNet-18	✓	55.6
Supervised [3]	✓	58.2

2.5.4 Tracking Pose Keypoint Propagation on the J-HMDB Dataset

We demonstrate that our model learns accurate correspondence by evaluating it on the J-HMDB dataset [47], which requires precise matching of points compared to the coarser propagation of masks. Given the 15 ground truth human pose keypoints in the first frame, we propagate them to the remaining frames. We quantitatively evaluate performance using the probability of correct keypoint (PCK) metric [145], which measures the ratio of joints that fall within a threshold distance from the ground truth joint locations. We show quantitative evaluations against the state-of-the-art methods in Table 2.5 and qualitative propagation results in Figure 2.4(b). Our model performs well versus all self-supervised methods [131, 120] and notably achieves better results than ResNet-18 [36] trained with classification labels [18].

2.5.5 Visual Tracking on the OTB Dataset

Other than the tasks that require dense matching, e.g., segmentation or keypoints propagation, the features learned by our model can be applied to object matching tasks such as visual tracking, because of its capability of localizing an object or a relatively global region. Without any fine-tuning, we directly integrate our network trained via self-supervision into a classic tracking framework [126, 98] based on correlation filters, by replacing the Siamese network in [126, 98] with our model, while keeping other parts in the tracking framework unchanged. Even without training with a correlation filter, our features are general and robust enough to achieve comparable performance on the OTB2015 dataset [137] to methods trained with this filter [126], as shown in Table 2.3. Figure 2.4(d) shows that our learned features are robust

Table 2.4: Segmentation propagation on VIP [150].

Model	Supervised	mIoU	AP^{r}_{vol}
DeepCluster. [8]	×	21.8	8.1
Wang et al. [131]	×	28.9	15.6
Ours	×	34.1	17.7
ResNet-18	✓	31.8	12.6
Fully Supervised [99]	✓	37.9	24.1

Table 2.5: Kepoints propagation on J-HMDB [47].

Model	Supervised	PCK@.1	PCK@.2
Vondrick et al. [120]	×	45.2	69.6
Wang et al. [131]	×	57.3	78.1
Ours	×	58.6	79.8
ResNet-18	✓	53.8	74.6
Fully Supervised [144]	✓	68.7	92.1

against occlusion (left), object scale, as well as illumination changes (right) and can track objects through a long sequence (hundreds of frames in the OTB2015 dataset).

2.5.6 Semantic and Instance Propagation on VIP Dataset

We evaluate our method on the VIP dataset [150], which includes dense human parts segmentation masks on both the *semantic* and *instance* levels. We use the same settings as Wang et al. [131] and resize the input frames to 560×560. For the semantic propagation task, we propagate the semantic segmentation maps of human parts (e.g., arms and legs) and evaluate performance via the mean IoU metric. For the part instance propagation task, we propagate the instance-level segmentation of human parts (e.g., arms of the first person or legs of the second person) and evaluate performance via the mean average precision of the instance-level human parsing metric [66]. Table 2.4 shows that our method performs favourably against all self-supervised methods and notably the ResNet-18 model trained on ImageNet with classification labels for both tasks. Figure 2.4(c) shows sample semantic segmentation propagation results. Interestingly, our model correctly propagates each part mask onto an unseen instance (the woman which does not appear in the first frame) in the second example.

2.5.7 Texture Propagation

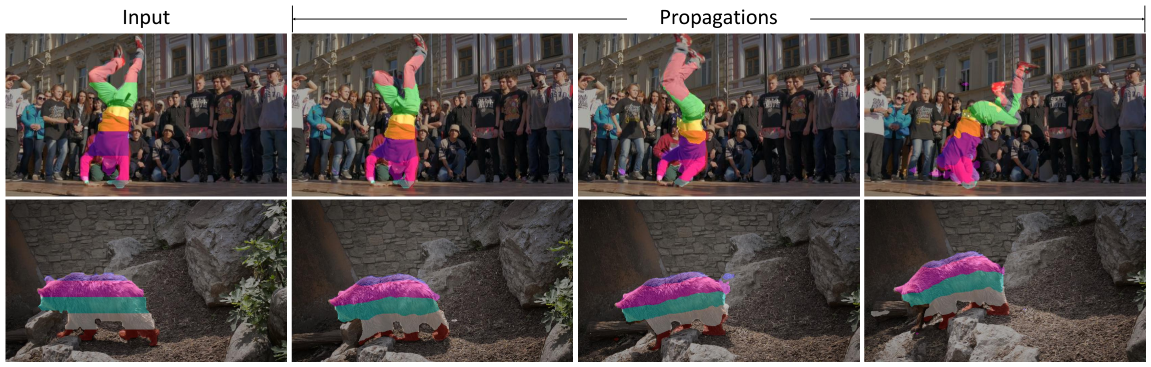


Figure 2.9: Texture Propagation.

In Figure 2.9, we show results of texture propagation. Following Wang et al. [131], we overlay a texture map on the object in the first video frame, then propagate this texture map across the rest of the video frames. As shown in Figure 2.9, our model is able to preserve the texture well during propagation, this indicates that our model is able to find precise correspondences between video frames.

2.6 Conclusions

In this chapter, we propose to learn correspondences across video frames in a self-supervised manner. Our method jointly tackles region-level and pixel-level correspondence learning and allows them to facilitate each other through a shared inter-frame affinity matrix. Experimental results demonstrate the effectiveness of our approach versus the state-of-the-art self-supervised video correspondence learning methods, as well as supervised models such as the ResNet-18 trained on ImageNet with classification labels.

Chapter 3

Single-view 3D Mesh Reconstruction via Semantic Correspondence

3.1 Overview

In this chapter, we learn a self-supervised, single-view 3D reconstruction model that predicts the 3D mesh shape, texture and camera pose of a target object with a collection of 2D images and silhouettes. The proposed method does not necessitate 3D supervision, manually annotated keypoints, multi-view images of an object or a prior 3D template. The key insight of our work is that objects can be represented as a collection of deformable parts, and each part is semantically coherent across different instances of the same category (e.g., wings on birds and wheels on cars). Therefore, by leveraging part segmentation of a large collection of category-specific images learned via self-supervision, we can effectively enforce semantic consistency between the reconstructed meshes and the original images. This significantly reduces ambiguities during joint prediction of shape and camera pose of an object, along with texture. To the best of our knowledge, we are the first to try and solve the single-view reconstruction problem without a category-specific template mesh or semantic keypoints. Thus our model can easily generalize to various object categories without such labels, e.g., horses, penguins, etc. Through a variety of experiments on several categories of deformable and rigid objects, we demonstrate that our unsupervised method performs comparably if not better than existing category-specific reconstruction methods learned with supervision.

3.2 Introduction

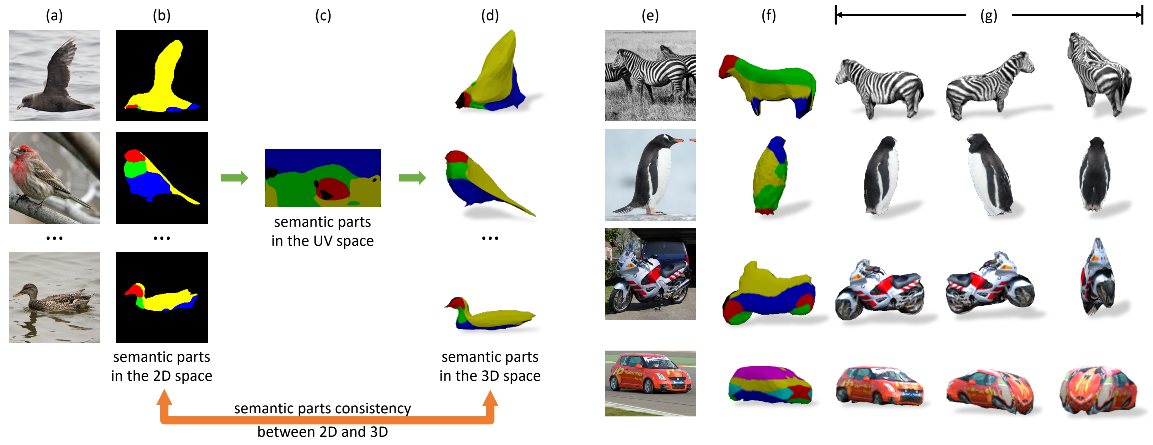

(a)
(b)
(c)
(d)
(e)
(f)
(g)
semantic parts in the UV space
semantic parts in the 2D space
semantic parts in the 3D space
semantic parts consistency between 2D and 3D

Figure 3.1: Self-supervision with semantic part consistency (a–d): (a) Images of different objects in the same category (e.g., birds in this example). (b) Semantic part segmentation for each image learned via self-supervision. (c) Canonical semantic UV map for the category. (d) Semantic part segmentation on meshes. Single-view 3D Mesh reconstruction (e–g): Reconstruction (inference) of each single-view image (e) is demonstrated in (g), along with semantic labels of the mesh in (f).

Recovering both 3D shape and texture, and camera pose from 2D images is a highly ill-posed problem due to its inherent ambiguity. Existing methods resolve this task by utilizing various forms of supervision such as ground truth 3D shapes [15, 132, 125], 2D semantic keypoints [49], shading [37], category-level 3D templates [61] or multiple views of each object instance [142, 53, 133, 100]. These types of supervision signals require tedious human effort, and hence make it challenging to generalize to many object categories that lack such annotations. On the other hand, learning to reconstruct by not using any 3D shapes, templates, or keypoint annotations, i.e., with only a collection of single-view images and silhouettes of object instances, remains challenging. This is because the reconstruction model learned without the aforementioned supervisory signals leads to erroneous 3D reconstructions. A typical failure case is caused by the "camera-shape ambiguity", wherein, incorrectly predicted camera pose and shape result in a rendering and object boundary that closely match the input 2D image and its silhouette, as shown in Figure 3.2 (c) and (d).

Interestingly, humans, even infants who have never been taught about objects in a category, tend to mentally reconstruct objects in that category by perceiving them as a combination of several basic parts, e.g., a bird has two legs, two wings, and one head, etc., and use the parts to associate all the divergent instances of the category. By observing object parts, humans can also roughly infer the relative camera pose and 3D shape of any specific instance. In computer vision, a similar intuition is formulated by the deformable parts model, where objects are represented as a set of parts arranged in a deformable configuration [23, 94].

Inspired by this intuition, we learn a single-view reconstruction model from a collection of images and silhouettes. We utilize the semantic parts in both the 2D and 3D space, along with their consistency to correctly estimate shape and camera pose. Specifically, we first leverage self-supervised co-part segmentation (SCOPS [42]) to decompose 2D images into a collection of semantic parts (Figure 3.1(b)). By exploiting the property of *semantic part invariance*, which states that the semantic part label of a point on the mesh surface does not change even when the mesh shape is deformed, we associate the semantic parts of *different* object instances with each other and build a category-level canonical semantic UV map (Figure 3.1(c)). The semantic part label of each point on the reconstructed mesh surface (Figure 3.1(d)) is then defined by this canonical semantic UV map. Finally, we resolve the aforementioned "camera-shape ambiguity" and learn the self-supervised reconstruction model by encouraging the consistency of semantic part labels in both the 2D and 3D space (Figure 3.1, orange arrow). Furthermore, we train our model by iteratively learning (a) instance-level reconstruction and (b) a category-level template mesh from scratch. Thus, our model also does not require a pre-defined 3D template mesh or any other shape prior. Our main contribution is a 3D reconstruction model that is able to:

- Conduct single-view mesh reconstruction *without* any of the following forms of supervision: category-level 3D template prior, annotated keypoints, camera pose or multi-view images. In other words, the model can be generalized to other categories which do not have well-defined keypoints, e.g., penguin.

- Leverage the *semantic part invariance* property of object instances of a category as a deformable parts model.

- Learn a category-level 3D shape template from scratch via iterative learning.
- Perform comparably to the state-of-the-art supervised methods [49, 61] trained with either pre-defined templates or annotated keypoints, while also improving the self-supervised semantic co-part segmentation model (SCOPS [42]).

3.3 Related Work

3D Shape Representation

Various representations have been explored for 3D processing tasks, including point clouds [22], implicit surfaces [84, 75], triangular meshes [49, 53, 74, 51, 125, 89, 132] and voxel grids [15, 27, 32, 116, 133, 142, 153, 34]. Among these, while both voxels and point clouds are more friendly to deep learning architectures (e.g., VON [134, 152], PointNet [96, 97], etc), they suffer either from issues of memory inefficiency or are not amenable to differentiable rendering. Hence, in this work, we adopt triangular meshes [49, 53, 74, 51, 125, 89, 132] for 3D reconstruction.

Single-view 3D Reconstruction

Single-view 3D reconstruction [15, 27, 32, 116, 133, 142, 153, 22, 37] aims to reconstruct a 3D shape given a single input image. One line of works have explored this ill-posed task with varying degree of supervision. Several methods [125, 89, 132] utilize image and ground truth 3D mesh pairs as supervision. This either requires significant manual annotation effort [139] or is restricted to synthetic data [9]. More recently, a few works [53, 74, 51, 12] avoid 3D supervision by taking advantage of differentiable renderers [53, 74, 12] and the "analysis-by-synthesis" approach, with either multiple views, or known ground truth camera poses.

To further relax constraints on supervision, Kanazawa et al. [49] explored 3D reconstruction from a collection of images of different instances. However, their method still requires annotated 2D keypoints to infer camera pose correctly. It is also the first work to propose a learnable category-level 3D template shape, which, however, needs to be initialized from a keypoint-dependent 3D convex hull. Similar problem settings have also been explored in other methods [109, 135, 38], but with object

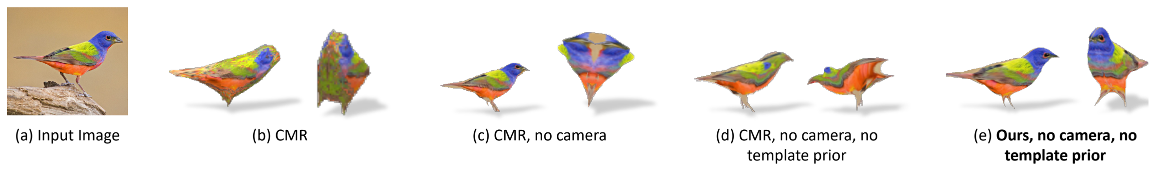

(a) Input Image
(b) CMR
(c) CMR, no camera
(d) CMR, no camera, no template prior
(e) Ours, no camera, no template prior

Figure 3.2: Comparison with baselines. Each reconstructed mesh is rendered in the original view of the input image and the frontal view of the bird. (b) Shows the result from CMR with camera pose and template prior supervision. (c) Shows CMR with only template prior. (d) Shows CMR without both types of supervision where the model completely fails to learn the texture and shape. In contrast, our model in (e) reconstructs correctly even without supervision from camera pose or a template prior.

categories restricted to rigid or structured objects, such as cars or faces. Different from all these works, we target both rigid and non-rigid objects (e.g., birds, horses, penguins, motorbikes and cars shown in Figure 3.1 (e)-(g)) and propose a method that jointly estimates a 3D mesh, texture, and camera pose from a single-view image, using only a collection of images with silhouettes as supervisions. In other words, we do not require 3D template priors, annotated keypoints, or multi-view images.

Self-supervised Correspondence Learning

Our work is also related to self-supervised cross-instance correspondence learning, via landmarks [111, 149, 40, 105], part segments [16, 42], or canonical surface mapping [61]. We utilize self-supervised co-parts segmentation [42] to enforce semantic consistency, which was originally proposed purely for 2D images. The work of [61] learns a mapping function that maps pixels in 2D images to a predefined category-level template in a self-supervised manner. However, it dose not use the learned correspondence for 3D reconstruction. We show that our work, despite having a focus on 3D reconstruction, outperforms [61] at learning 2D to 3D correspondences as well.

3.4 Approach

To fully reconstruct the 3D mesh of an object instance from an image, a network should be able to jointly predict the shape and texture of the object, and the camera pose of the image. We start with the existing network from [49] (CMR) as the baseline reconstruction network. Given an input image, CMR extracts the image features using an encoder E and jointly predicts the mesh shape, camera pose and mesh texture by three decoders D_{shape}, D_{camera} and D_{texture}. The mesh shape V is reconstructed by predicting vertex offsets ΔV to a category-specific shape template $\bar{V}$, while the camera pose θ is represented by a weak perspective transformation. To reconstruct mesh textures, the texture decoder outputs a UV texture flow (I_{flow}) that maps pixels from the input image to the UV space. A pre-defined mapping function Φ further maps each pixel in the UV space to a point on the mesh surface.

One of the key elements for the CMR method to perform well is to exploit *mannually annotated semantic keypoints* for (i) precisely pre-computing the ground truth camera pose for each instance, and (ii) estimating a category-level 3D template prior. However, annotating keypoints is tedious, not well-defined for most object categories in the world and impossible to generalize to new categories. Thus, we propose a method within a more scalable, but challenging self-supervised setting *without* using manually annotated keypoints to estimate camera pose or a template prior.

Not surprisingly, simply taking out the keypoints supervision, as well as all the related information (i.e., the camera pose and the template prior) from the CMR network makes it unable to predict camera pose and shape correctly, as shown in Figure 3.2(c) and (d). This is due to the inherent ambiguity of hallucinating 3D meshes from only single-view 2D observations, where the model trivially picks a combination of camera pose and shape that yields the rendering that matches the given image and silhouette. Consider an extreme case, where the model predicts the front view for all instances, but is still able to match the image and silhouette observations by deforming each instance mesh accordingly.

In this work, we propose a framework (Figure 3.3) designed for self-supervised mesh reconstruction learning, i.e., with only a collection of images and silhouettes as supervision. The framework consists of: (i) A reconstruction network (green box)

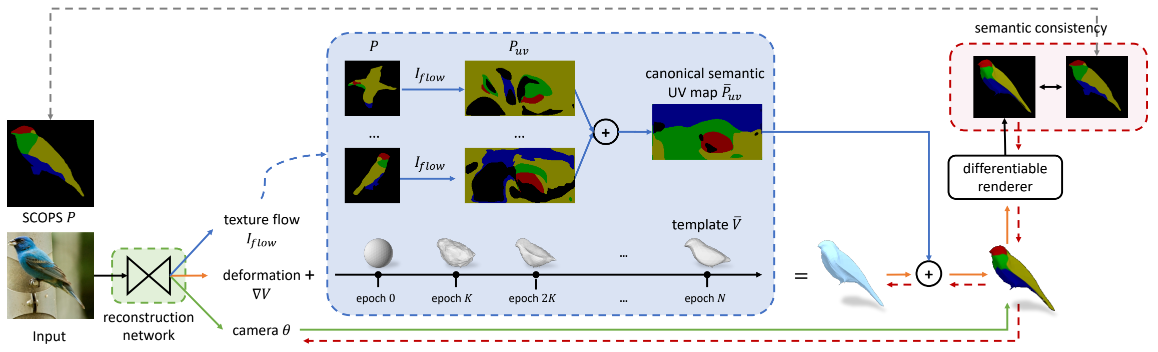


Figure 3.3: Overview.(a) Green box: The reconstruction network. (b) Red box: Semantic part consistency constraint, see Section 3.4.1 for more details. (c) Blue box: Computing the canonical semantic UV map and the template shape using the reconstruction network, see Section 3.4.2. The red dashed arrows show that the gradients from the semantic part consistency constraint facilitate shape and viewpoint estimation.

that has the same architecture as [49] – it consists of an image encoder E and three decoders D_{shape}, D_{camera} and D_{texture} that jointly predict the mesh deformation ΔV, texture flow I_{flow} and camera pose θ for the instance in the image. (ii) A semantic consistency constraint (red box in Figure 3.3) that regularizes the learning of module (i) and largely resolves the aforementioned "camera-shape ambiguity" under the self-supervised setting. We introduce this module in Section 3.4.1. (iii) A module that learns the canonical semantic UV map and category-level template from scratch (blue box in Figure 3.3). This module is iteratively trained with module (i) and discussed in Section 3.4.2.

3.4.1 Resolving Camera-Shape Ambiguity via Semantic Consistency

In this section, we show the key to solving the "camera-shape ambiguity" is to make use of the semantic parts of object instances in both 3D and 2D. Specifically, we exploit the fact that (i) in the 2D space, self-supervised co-part segmentation [42] provides correct part segments for a majority of the object instances, even for those with large shape variations (see Figure 3.1(b)); and (ii) in the 3D space, semantic parts are invariant to mesh deformations, i.e., the semantic part label of a specific point on the mesh surface is consistent across all reconstructed instances of a category. We demonstrate that this *semantic part invariance* allows us to build a category-level semantic UV map, namely the canonical semantic UV map, shared by all instances, which in turn allows us to assign semantic part labels to each point on the mesh. By enforcing consistency between the canonical semantic map and an instance's part segmentation in the 2D space, the camera-shape confusion can be largely resolved.

Part Segmentation in 2D via SCOPS [42]

SCOPS is a self-supervised method that learns semantic part segmentation from a collection of images of an object category (see Figure 3.1(b)). The model leverages concentration and equivariance loss functions, as well as part basis discovery to output a probabilistic map w.r.t. the discovered parts that are semantically consistent across different object instances.

Part Segmentation in 3D via Canonical Semantic UV Map

Given the semantic part segmentation of 2D images estimated by SCOPS, how can we obtain the semantic part labels for each point on the mesh surface? One intuitive way is to obtain a mapping from the 2D image space to the 3D shape space. Therefore, we propose to first utilize the learned texture flow I_{flow} by our reconstruction network that naturally forms a mapping from the 2D image space to the UV texture space, and then further map the semantic labels from the UV space to the mesh surface by the pre-defined mapping function Φ. We denote the semantic part segmentation of image i as $P^i \in \mathbb{R}^{H \times W \times N_p}$ (see Figure 3.3 in the blue bbox), where H and W are the height and width of the image, respectively and N_p is the number of semantic parts. By mapping P^i from the 2D image space to the UV space using the learned texture flow, we obtain a "semantic UV map" denoted as $P^i_{\text{uv}} \in \mathbb{R}^{H_{\text{uv}} \times W_{\text{uv}} \times N_p}$, where H_{uv} and W_{uv} are the UV map's height and width, respectively.

Ideally, all instances should result in the same semantic UV map – the canonical semantic UV map for a category, regardless of shape differences of instances. This is because: (i) the *semantic part invariance* states that the semantic part labels assigned to each point on the mesh surface are consistent across different instances; and (ii) the mapping function Φ that maps pixels from the UV space to the mesh surface is pre-defined and independent of deformations in the 3D space, such as face location or area changes. Thus, the semantic part labels of pixels in the UV map should also be consistent across different instances.

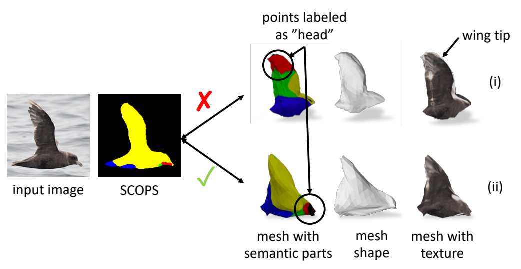
However, if we directly sample the individual P^i via the learned texture flow I_{flow}, the obtained semantic UV maps are indeed very different between instances, as shown in Figure 3.3 (blue box). This is caused by the fact that (i) the part segmentation predictions produced by the self-supervised SCOPS method are noisy, and (ii) texture flow prediction is highly uncertain for the invisible faces of the reconstructed mesh. Therefore, we approximate the canonical semantic UV map, denoted as $\bar{P}_{\text{uv}}$ by aggregating the individual semantic UV maps:

$$\bar{P}_{\text{uv}} = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} I^i_{\text{flow}}(P^i), \tag{3.1}$$

where $I^i_{\text{flow}}(P^i)$ indicates the sampling of P^i by I_{flow} and $\mathcal{U}$ is a subset of selected training samples with accurate texture flow prediction. Note that when we update the canonical semantic UV map using Eq 3.1, to avoid using training samples with outliers, e.g., those caused by inaccurate prediction of I^i_{flow}, we choose an exemplar training example with the smallest perceptual distance objective, and form the set $\mathcal{U}$ of the top k training samples that have the most similar semantic UV maps (as measured by the L2 norm) to the exemplar. Through this aggregation process, $\bar{P}_{\text{uv}}$ produces a mean semantic UV map, which effectively eliminates outliers (i.e., instances with incorrect SCOPS), as well as the noisy pixel-level predictions.

Semantic Consistency between 2D and 3D

As mentioned above, because our model learns via self-supervision and only relies on images and silhouettes that do not provide any semantic part information, it suffers from the "camera-shape ambiguity" introduced in Section 3.2. Take row (i) in Figure 3.4 as an example. The model erroneously forms the wing's tip in the reconstructed bird by deforming the mesh faces assigned to the "head part" (colored in red). This incorrect shape reconstruction, associated with an incorrect camera pose, however, can yield a rendering that matches the observed image and its silhouette.



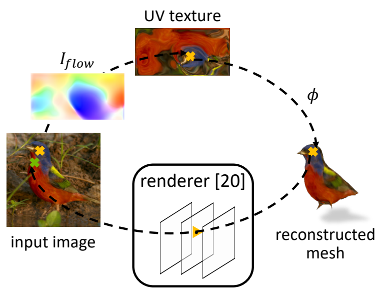


Figure 3.5: The process of texture cycle consistency constraint computation.

This ambiguity, although is not easy to spot by only comparing the rendering of the reconstruction with the input image, however, can be identified once the semantic part label for each point on the mesh surface is available. One can tell that the reconstruction in row (i) of Figure 3.4 is wrong by comparing the rendering of the semantic part labels on the mesh surface and the 2D SCOPS part segmentation. Only when the camera pose and shape are both correct, will the rendering and the SCOPS segmentation be consistent, as shown in row (ii) in Figure 3.4. This observation inspires us to propose a probability and a vertex-based constraint that facilitate correct camera pose and shape learning by encouraging the consistency of semantic part labels in both 2D images and in the mesh surface.

Probability-based constraint. For each reconstructed mesh instance i, we map the canonical semantic UV map $\bar{P}_{\text{uv}}$ onto its surface by the UV mapping Φ and render it using the predicted camera pose θ^i. We denote the projection from 3D to 2D as $\mathcal{R}$. We constrain the projected probability map to be close to the SCOPS part segmentation probability map P^i by computing the loss:

$$L_{\text{sp}} = \left\| P^i - \mathcal{R}(\Phi(\bar{P}_{\text{uv}}); \theta^i) \right\|^2. \tag{3.2}$$

We empirically found the mean squared error (MSE) metric to be more robust than the Kullback–Leibler divergence for comparing two probability maps.

Vertex-based constraint. We also propose a vertex-based constraint to enhance semantic part consistency by enforcing that 3D vertices assigned a part label p, after being projected to the 2D domain with the predicted camera pose θ^i, align with the area assigned to that part in the input image:

$$L_{\text{sv}} = \sum_{p=1}^{N_p} \frac{1}{|\bar{V}_p|} \text{Chamfer}(\mathcal{R}(\bar{V}_p; \theta^i), Y_p^i), \tag{3.3}$$

where $\bar{V}_p$ is the set of vertices on a learned category-level 3D *template* $\bar{V}$ (see Section 3.4.2) with the part label p, Y_p^i is the set of 2D pixels sampled from the part p in the original input image and N_p is the number of parts. Here we use the *Chamfer distance*, because the projected vertices and pixels with the same part label p in the input image do not have a strictly one-to-one correspondence.

Note that, $\bar{V}_p$ is a set of vertices on the category-level shape template $\bar{V}$ as opposed to each instance reconstruction V^i, since using V^i results in a degenerate solution where the network only alters 3D shape to satisfy this vertex-based constraint, rather than the camera pose. Instead, using $\bar{V}$ drives the network towards learning the correct camera pose, in addition to shape.

3.4.2 Progressive Training

We train the framework in Figure 3.3 via progressive training based on two considerations: (a) building the canonical semantic UV map, introduced in Section 3.4.1, requires reliable texture flows to map the SCOPS from images to the UV space. Thus the canonical semantic UV map can only be obtained after the reconstruction network is able to predict texture flow reasonably well, and (b) a canonical 3D shape template [49, 61] is desirable, since it speeds up the convergence of the network [49] and also avoids degenerate solutions when applying the *vertex-based constrain* as introduced in Section 3.4.1. However, jointly learning the category-level 3D shape template and the instance-level reconstruction network leads to undesired trivial solutions. Thus, we propose an expectation-maximization (EM) style progressive training procedure below. In the E-step, we train the reconstruction network with the current template and canonical semantic UV map fixed, and in the M-step, we update the template and the canonical semantic UV map using the reconstruction network learned in the E-step.

E-step: Learning Instance-specific Reconstruction

In the E-step, we fix the canonical semantic UV map as well as the category-level template and train the reconstruction network mainly with the following objectives. (i) A negative IoU objective [51] between the rendered and the ground truth silhouettes for shape learning. (ii) A perceptual distance objective [148, 49] between the rendered and the input RGB images for texture learning. (iii) The probability and vertex-based constraints introduced in Section 3.4.1 to resolve the "camera-shape ambiguity" under the self-supervised setting. (iv) A texture consistency constraint to facilitate accurate texture flow learning that will be introduced in Section 3.4.3. (v) Smoothness Term. We utilize a graph Laplacian constraint to encourage the reconstructed mesh surface to be smooth [49, 74], and adopt an edge regularization to penalize irregularly-sized faces as in [125, 28]. More details can be found in [49, 74, 125, 28]. (vi) Adversarial Training. To constrain the reconstructed meshes to look plausible from all views, we also introduce adversarial training [29] into the mesh reconstruction framework [51]. We render the reconstructed mesh from a randomly sampled camera pose to obtain an image I_{rd}, and pass it together with a random real image I_{rl} into a discriminator. By learning to discriminate between the real and rendered images, the discriminator learns shape priors and constrains the reconstruction model to generate meshes that are plausible from all viewpoints. The adversarial loss is:

$$L_{\text{adv}}(R, D) = \mathbb{E}_{I_{\text{rl}}}[\log D(I_{\text{rl}})] + \mathbb{E}_{I_{\text{rd}}}[\log(1 - D(I_{\text{rd}}))], \tag{3.4}$$

where R and D are the reconstruction and discriminator networks, respectively. Figure 3.6 illustrates the adversarial objective. Note that in the first E-step, the template is a sphere and hence the probability and vertex-based constraints are not used.

M-step: Canonical UV Map and Template Learning

In the M-step, we compute the canonical semantic UV map introduced in Section 3.4.1 and learn a category-level template from scratch, i.e., from a sphere primitive. As far as we know, we are the first method that learns a category-level template from scratch. This is in contrast to existing methods [61], where the template is either a readily available instance mesh from the category or is estimated from

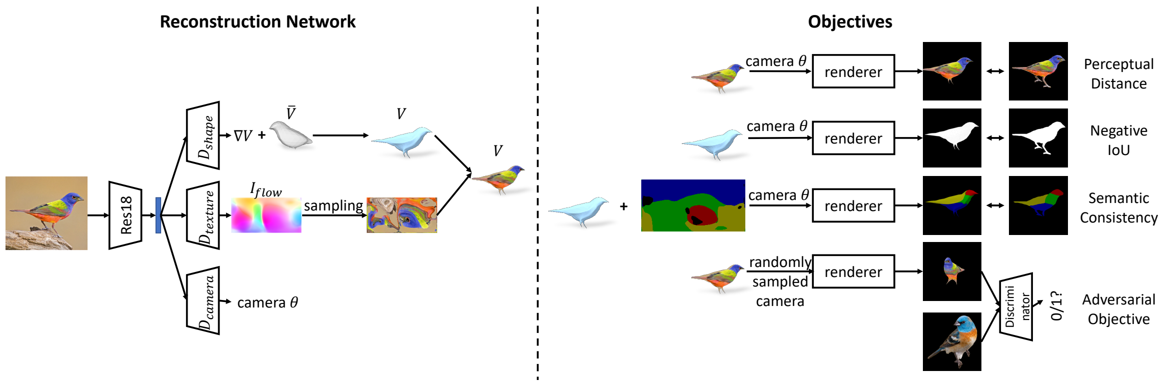


Figure 3.6: Detailed network Architecture and Objectives.

annotated keypoints [49]. Jointly learning the shape template along with the reconstruction network does not guarantee a meaningful "mean shape" which encapsulates the most representative characteristics of objects in a category. Instead, we propose a feed-forward template learning approach: the template starts out as a sphere and is updated every K training epochs by:

$$\bar{V}_t = \bar{V}_{t-1} + D_{\text{shape}}\left(\frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} E(I^i)\right), \tag{3.5}$$

where $\bar{V}_t$ and $\bar{V}_{t-1}$ are the updated and current templates, respectively, I^i is the input image passed to the image encoder E and D_{shape} is the shape decoder (see the beginning of Section 3.4). $\mathcal{Q}$ is a set of training images with consistent mesh predictions. Instead of using all training samples to obtain the averaged feature, we select a subset of the training samples to form a set $\mathcal{Q}$ and compute the averaged feature for the samples in this set. In the following, we explain why and how to form this set $\mathcal{Q}$ used in Eq. 3.5. Empirically we found that for several categories, there exist ambiguities that produce inconsistent mesh reconstructions, e.g., side-view images of horses could be reconstructed with their heads on either the left or the right side. Aggregating such instance meshes leads to incorrect estimation of the category-level template. To resolve this, we select a subset of reconstructed meshes whose viewpoints roughly match (e.g. horses with heads on the left side). To do so, from the meshes reconstructed for all the training images, we first choose the instance with the most "reliable" reconstruction results, i.e., the instance whose rendered silhouette has the largest intersection over union (IoU) with its corresponding ground truth silhouette, as an exemplar (e.g. a horse shape with its head on the left). We then use the top k training samples with meshes that are most similar to the exemplar mesh to form the subset $\mathcal{Q}$ in Eq. 3.5 (e.g., all chosen horse samples have heads on the left). We measure the similarity between an individual instance mesh and the exemplar mesh by computing the IoU between their rendered silhouettes. The template $\bar{V}_t$ is the mean shape of instances in a category for the current epoch, which enforces a meaningful shape (e.g., the template looks like a bird) rather than an arbitrary form for the category.

3.4.3 Texture Cycle Consistency Constraint

One issue with the learned texture flow is that the texture of 3D mesh faces with a similar color (e.g., black) can be incorrectly sampled from a single pixel location of the image. Thus we introduce a texture cycle consistency objective to regularize the predicted texture flow (i.e., 2D→3D) to be consistent with the camera projection (i.e., 3D→2D). As shown in Figure 3.5, considering the pixel marked with a yellow cross in the input image, it can be mapped to the mesh surface through the predicted texture flow I_{flow} along with the pre-defined mapping function Φ introduced in Section 3.4. Meanwhile, its mapping on the mesh surface can be re-projected back to the 2D image by the predicted camera pose, as shown by the green cross in Figure 3.5. If the predicted texture flow conforms to the predicted camera pose, the yellow and green crosses would overlap, forming a $2D \rightarrow 3D \rightarrow 2D$ cycle.

Formally, given a triangle face j, we denote the set of input image pixels mapped to this face by texture flow as Ω^j_{in}. We further infer the set of pixels (denoted as Ω^j_{out}) projected from the triangle face j in the rendering operation by taking advantage of the probability map $\mathcal{W} \in \mathcal{R}^{|F|\times(H\times W)}$ in the differentiable renderer [74] where $|F|, H, W$ are the number of faces, height and width of the input image, respectively. Each entry in $\mathcal{W}^m_j$ indicates the probability of face j being projected onto the pixel m. We compute the geometric center of both sets (Ω^j_{in} and Ω^j_{out}), denoted by $\mathcal{C}^j_{\text{in}}$ and $\mathcal{C}^j_{\text{out}}$, respectively as:

$$\mathcal{C}^j_{\text{in}} = \frac{1}{N_c}\sum_{m=1}^{N_c} \Phi(I_{\text{flow}}(\mathcal{G}^m))_j; \quad \mathcal{C}^j_{\text{out}} = \frac{\sum_{m=1}^{H\times W} \mathcal{W}^m_j \times \mathcal{G}^m}{\sum_{m=1}^{H\times W} \mathcal{W}^m_j}, \tag{3.6}$$

where $\mathcal{G} \in \mathbb{R}^{(H\times W)\times 2}$ is a standard coordinate grid of the projected image (containing pixel location (u, v) values), and Φ is the fixed UV mapping that, along with the texture flow I_{flow} maps pixels from the 2D input image to a mesh face j, as discussed in the beginning of Section 3.4. N_c is the number of pixels in the input image mapped to each triangular face and $\times$ indicates multiplication between two scalars. We constrain the predicted texture flow to be consistent with the rendering operation by encouraging $\mathcal{C}^j_{\text{in}}$ to be close to $\mathcal{C}^j_{\text{out}}$:

$$L_{\text{tcyc}} = \frac{1}{|F|}\sum_{j=1}^{|F|} \left\|\mathcal{C}^j_{\text{in}} - \mathcal{C}^j_{\text{out}}\right\|^2_F . \tag{3.7}$$

We note that while not targeting 3D mesh reconstruction directly, a similar intuition, but with a different formulation was also introduced in [61].

3.5 Experimental Results

We first introduce our experimental settings in Section 3.5.1, and present qualitative evaluations for the bird, horse, motorbike and car categories in Section 3.5.4. Quantitative evaluations and ablation studies for the contribution of each proposed module are discussed in Section 3.5.5 and Section 3.5.6, respectively.

3.5.1 Experimental Settings

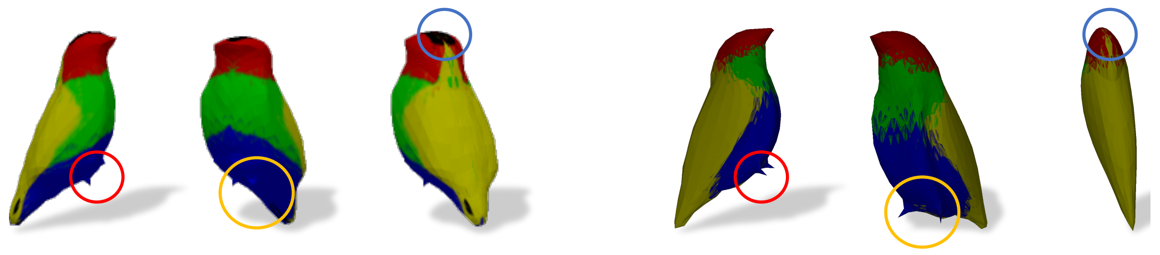
We validate our method on both rigid objects, i.e., *car* and *motorcycle* images from the PASCAL3D+ dataset [140], and non-rigid objects, i.e., *bird* images from the CUB-200-2011 dataset [122], *horse*, *zebra*, *cow* images from the ImageNet dataset [18] and *penguin* images from the OpenImages dataset [62]. We use progressive training (Section 3.4.2) to learn the model parameters. In each E-step, the reconstruction network is trained for 200 epochs and then used to update the template and the canonical semantic UV map in the M-step. The only exception is in the first round (a round consists of one E and M-step), where we train the reconstruction network without the semantic consistency constraint. This is because, at the beginning of training, I_{flow} is less reliable, which in turn makes the canonical semantic UV map less accurate.

3.5.2 Network Architecture

We present the details of our network architecture as well as training objectives in Figure 3.6. We use the same network as in CMR [49], where: (i) the encoder is the ResNet18 network [36] with four residual blocks and is pretrained on the ImageNet [18] dataset, (ii) the shape decoder consists of one fully connected layer to decode shape deformation ΔV, (iii) the texture decoder contains two fully-connected layers followed by eleven upsample and convolution layers to predict the texture flow I_{flow}, (iv) the camera pose decoder contains three parallel fully connected layers to predict the scale, translation and rotation, respectively and these three parameters together compose the camera pose θ. Note that we use the one camera hypothesis in the first EM training round and use the multiple camera hypothesis (eight camera hypothesis) as in [61, 44, 115] to avoid local minima in the subsequent rounds. To render the reconstructed meshes, we utilize the Soft Rasterizer [74] instead of the Neural Mesh Renderer [53] used in the CMR [49]. This is because it provides the probability map described in Section 3.4.3 for the texture cycle consistency constraint.

3.5.3 Network Training

We train the reconstruction network with an initial learning rate of 1e-4 and gradually decay it by a factor of 0.5 every 2000 iterations. The network is trained for two EM training rounds (each training round contains one E and M-step) on four NVIDIA Tesla V100 GPUs for two days. We found that two rounds of EM training are sufficient to generate high-quality reconstruction results. During the inference stage, the model takes 0.022 seconds to reconstruct a 3D mesh from a 256×256 sized single-view image on a single NVIDIA Tesla V100 GPU. In Figure 3.7, we show the learned template shape as well as the semantic parts after the first (left figure) and second M-steps (right figure), where both the template shape and the semantic parts after the second M-step are better than the first.



(a) Template with semantic parts after ***first*** EM training round

(b) Template with semantic parts after ***second*** EM training round

Figure 3.7: Visualization of the learned template and semantic parts. Notice the improvements of the template after the second M-step compared to the first, i.e., better feet shape in the red and yellow circles, and a part of the head (blue circle) that was mistakenly assigned to the background (colored in black) in the first step is corrected (colored in red) in the second M-step.

3.5.4 Qualitative Results

Thanks to the self-supervised setting, our model is able to learn from a collection of images and silhouettes (e.g., horse and cow images [18] and penguin images [62]), which cannot be achieved by existing methods [49, 125, 142, 53] that require extra supervisory signals.

Template and Semantic Parts on 3D Meshes

We show the learned templates for the bird, horse, motorbike and car categories in Figure 3.8 and Figure 3.9, which capture the shape characteristics of each category, including the details such as the beak and feet of a bird, etc. We also visualize the canonical semantic UV map by showing the semantic part labels assigned to each point on the template's surface. For instance, bird meshes have four semantic parts – head (red), neck (green), belly (blue) and back (yellow) in Figure 3.8, which are consistent with the part segmentation predicted by SCOPS [42].

Instance 3D Reconstruction

We show the results of 3D reconstruction from each single-view image in Figure 3.8 (b)-(d) and Figure 3.9 (b). Our model can reconstruct instances from an object category with highly divergent shapes, e.g., a thin bird in (b), a duck in (c) and a

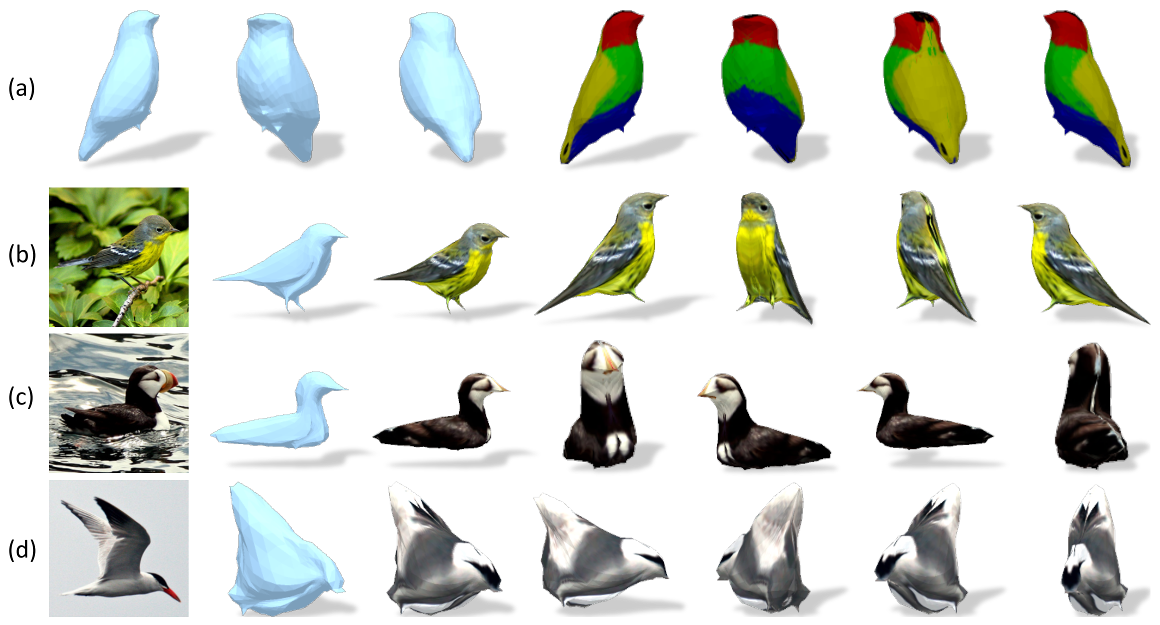


Figure 3.8: Learned template and instance reconstructions from single-view images. (a) The learned template shape (first three columns) and semantic parts (last four columns). (b)-(d) 3D reconstruction from a single-view image. In each row from left to right, we show the input image, reconstruction rendered using the predicted camera view and from four other views.

flying bird in (d). Our model also correctly maps the texture from each input image onto its 3D mesh, e.g., the eyes of each bird as well as fine textures on the back of the bird. Furthermore, the renderings of the reconstructed meshes under the predicted camera poses (2nd and 3rd columns in Figure 3.8 and Figure 3.9) match well with the input images in the first column, indicating that our model accurately predicts the original camera view.

We show more qualitative results for birds in Figure 3.11. We also show one application of our model to reconstruct 3D meshes of 2D bird paintings in Figure 3.10. Reconstruction of rigid objects (cars and motorbikes) is demonstrated in Figure 3.13, horses and cows in Figure 3.12, and penguins and zebras in Figure 3.14. Note that we use six semantic parts for the car category to encourage the SCOPS method [42] to differentiate between the fronts and the sides of cars. For other objects, we use four semantic parts.

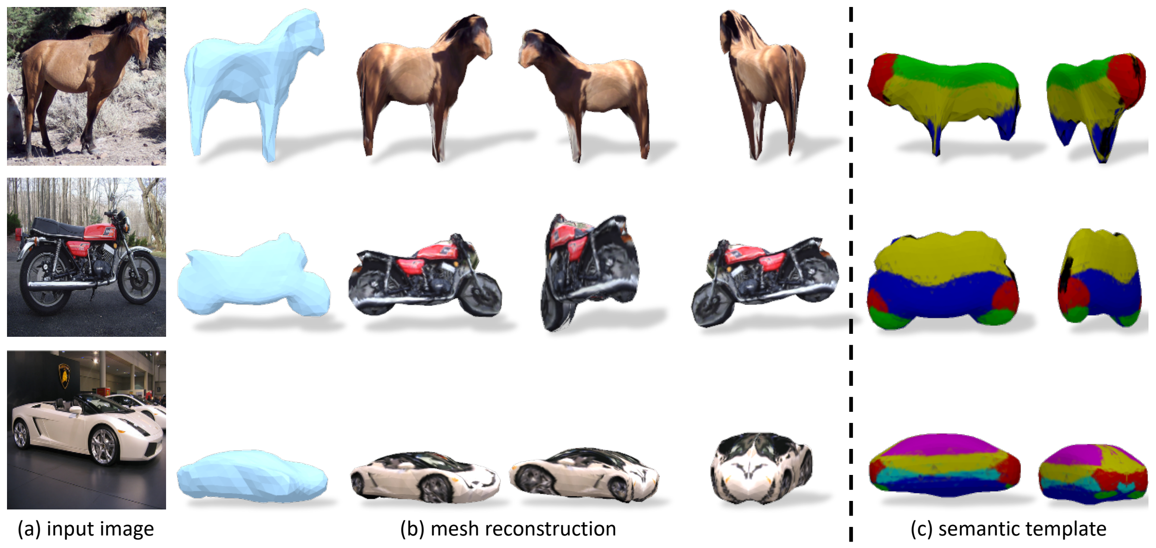


Figure 3.9: More reconstruction results. Visualization of instance-level reconstructions and semantic templates for the *horse*, *motorbike* and *car* categories.

3.5.5 Quantitative Evaluations

As a self-supervised approach, our model is more practically suited to reconstruct many non-rigid objects, e.g., animals captured in the wild that do not have 3D ground truth meshes available. Therefore, we treat the bird category [122] as the major one for qualitative evaluation, through the task of keypoint transfer following previous work [61]. Given a pair of source and target images of two different object instances from a category, we map a set of annotated keypoints from the source image to the target image by first mapping them onto the learned shape template and then to the target image. Each mapping can be carried out by either the learned texture flow or the camera pose, as explained below.

To validate 3D reconstruction results, we also evaluate our model on rigid objects, e.g., cars [140], in terms of 3D IoU. However, we note that reconstruction of such rigid objects for which the ground truth 3D meshes/CAD models are easy to obtain, is not the major focus of this self-supervised method.

We first evaluate shape reconstruction on the bird category. Due to a lack of ground truth 3D shapes in the CUB-200-2011 dataset [122], we follow [49] and compute the mask reprojection accuracy – the intersection over union (IoU) between rendered and ground truth silhouettes. As shown in Table 3.1, our model is able to

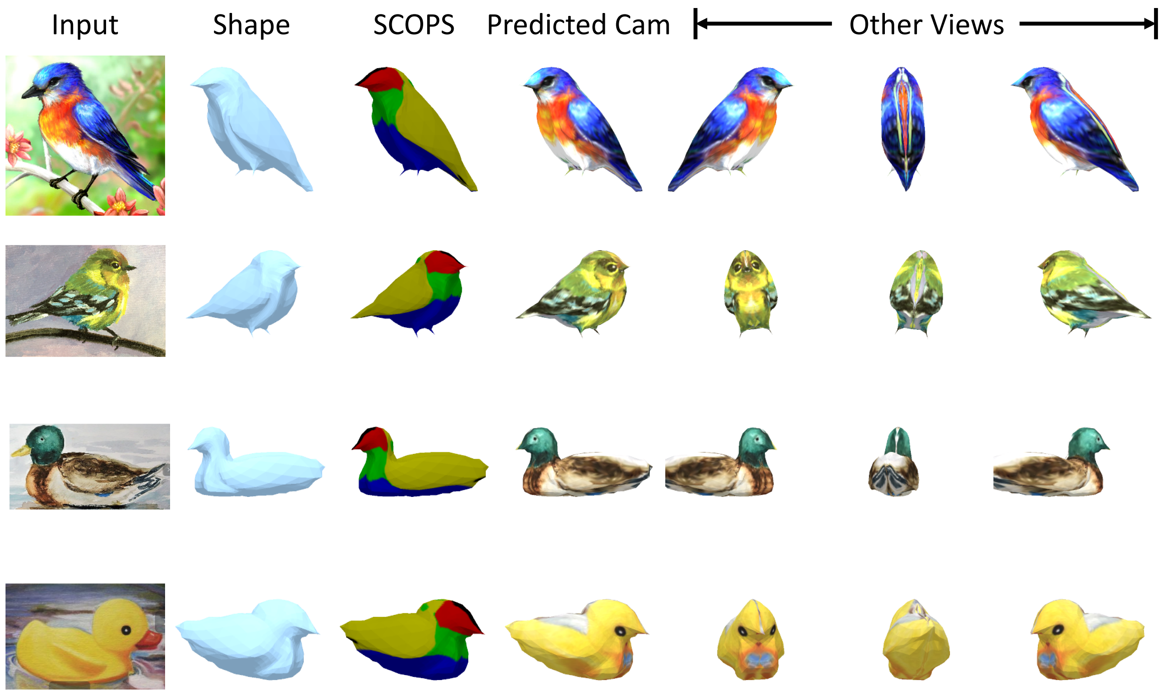


Figure 3.10: Results of applying our reconstruction model on bird paintings.

Table 3.1: Quantitative evaluation of mask IoU and keypoint transfer (KT) on the CUB dataset [122]. The comparisons are against the baseline supervised models [49, 61].

(a) Metric	(b) CMR [49]	(c) CSM [61]	(d) Ours
Mask IoU ↑	0.706	-	0.734
KT (Camera) ↑	47.3	-	51.2
KT (Texture Flow) ↑	28.5	48.0	58.2

achieve comparable if not better mask reprojection accuracy compared to CMR [49], which unlike our method is learned with additional supervision from semantic keypoints. This indicates that our model is able to predict 3D mesh reconstructions and camera poses that are well matched to the 2D observations.

Next, we evaluate shape reconstruction on the car category. Although PASCAL3D+ [140] provides "ground truth" meshes (the most similar ones to the image in a mesh library), our reconstructed meshes are not aligned with these "ground truth" meshes since our self-suerpvised model is free to learn its own "canonical reference frame". Thus, to quantitatively evaluate the intersection over union (IoU) between the two meshes, following CMR [49], we exhaustively search a set of scale, translation

Table 3.2: Ablation studies of each proposed module by evaluating mask IoU and keypoint transfer (KT) on the CUB-200-2011 dataset [122].

(a) Metric	(b) Ours	(c) w/o L_{tcyc}	(d) w/o L_{sv} & L_{sp}	(e) with original [42]
Mask IoU ↑	0.734	0.731	0.744	0.731
KT (Camera) ↑	51.2	48.5	29.0	48.7
KT (Texture Flow) ↑	58.2	51.0	32.8	52.9

and rotation parameters that best align to the "ground truth" meshes. Our method achieves an IoU (0.62) that is comparable to CMR [49] (0.64), even though the latter is trained with keypoints supervision.

Consider two different instances of a category as source and target images. To evaluate learned texture flow via keypoint transfer, given an annotated keypoint k^s in a source image (s), we map it to a triangle face (F_j) on the template using its learned flow I^s_{flow}. We then find all pixels (Ω_j) in the target image (t) that are mapped to the same triangle face F_j, by its texture flow I^t_{flow} and compute the geometric center of all pixels in Ω_j. We compare the location of the geometric center of Ω_j to the ground truth keypoint k^t and find the percentage of correct keypoints (PCK) as those that fall within a threshold distance $\alpha = 0.1$ of each other [61]. Figure 3.16 demonstrates qualitative visualizations of the keypoint transfer using texture flow and Table 3.1 shows that the texture flow learned by our method, even without supervision, outperforms the 2D$\rightarrow$3D mappings learned by the supervised methods [49, 61].

To evaluate the learned camera pose via keypoint transfer, we first find the 3D template's vertex v that corresponds to a source image's annotated 2D keypoint k^s by rendering all 3D vertices using its predicted pose θ^s. Then, v is the vertex whose 2D projection lies closest to the keypoint k^s. Next, we render the point v with a target image's predicted pose θ^t and compare it to its ground truth keypoint k^t to compute PCK. Figure 3.16(b) demonstrates the keypoint transfer results by the predicted camera pose. Table 3.1 shows that our model achieves favourable performance against the baseline method [49].

3.5.6 Ablation Studies

In this section, we first discuss the contribution of each proposed module: (i) The semantic consistency constraint discussed in Section 3.4.1. (ii) The texture cycle consistency introduced in Section 3.4.3. We evaluate on the CUB-200-2011 dataset [122] and use the mask reprojection accuracy as well as the keypoint transfer (via texture flow and via camera pose) accuracy discussed in Section 3.5.5 as our metrics.

As shown in Table 3.2 (b) *vs.* (d) our baseline model trained without the semantic consistency constraint performs much worse at the keypoint transfer task than our full model, indicating this baseline model predicts incorrect texture flow and camera views. We note that this baseline model achieves better mask IoU because the model trained without any constraint is more prone to overfit to the 2D silhouette observations.

Our model trained without the texture cycle consistency constraint achieves worse performance (Table 3.2 (b) *vs.*(c)) at transferring keypoints using the predicted texture flow. This proves the effectiveness of the texture cycle consistency constraint in encouraging the model to learn better texture flow.

Next, we show the results of three more baselines in Figure 3.15. The experimental settings for each are illustrated in Table 3.3 and are the following: (a) a basic model trained with only the texture cycle consistency constraint described in Section 3.4.3, but without any other proposed modules, i.e., the category-level template, the semantic consistency constraint and the adversarial training; (b) learning the model in (a) together with the category-level template; and (c) learning the model in (b) with the additional semantic consistency constraint.

As shown in Figure 3.15, the basic model (a) reconstructs meshes that only appear plausible from the observed view to match the 2D supervision (images and silhouettes). It fails to generate plausible results for unobserved views, e.g., for all the 3 examples. On adding template shape learning (see Section 3.4.2) to (a), the model in (b) learns more plausible reconstruction results across different views. This is because it is easier for the model to learn residuals w.r.t a category-level template compared to w.r.t a sphere, to match the 2D observations. However, without semantic part information, the model still suffers from the "camera-shape ambiguity" discussed in

Table 3.3: Settings of each baseline models in Section 3.5.6.

Module	category-level template	semantic consistency	adversarial training
baseline (a)	$\times$	$\times$	$\times$
baseline (b)	✓	$\times$	$\times$
baseline (c)	✓	✓	$\times$
Ours	✓	✓	✓

Table 3.4: Ablation studies of the probability and vertex-based semantic consistency constraints by evaluating the mask IoU and the keypoint transfer (KT) task on the CUB-200-2011 dataset [122].

(a) Metric	(b) Ours	(c) w/o L_{sv}	(d) w/o L_{sp} original [42]
Mask IoU $\uparrow$	0.734	0.6069	0.6418
KT (Camera) $\uparrow$	51.2	30.7	51.0
KT (Texture Flow) $\uparrow$	58.2	29.5	53.3

Section 3.2. For instance, the head of the template is deformed to form the tail and the wing's tip in the first and second examples, respectively in Figure 3.15. By additionally including the semantic consistency constraint in the model (c), the network is able to reduce the "camera-shape ambiguity" and predict the correct camera pose as well as the correct shape. Furthermore, adding adversarial training introduces better reconstruction details, as shown in Figure 3.15 (d). For instance, the bird may have more than two feet without the adversarial training constraint as demonstrated in the third example in Figure 3.15.

In addition, we demonstrate the effectiveness of the texture flow consistency constraint by visualizing the keypoint transfer results in Figure 3.16. The model trained without this constraint performs worse than our full model, especially when the bird has a uniform color, e.g., the second and the last examples in Figure 3.16. Figure 3.16 also shows that the proposed method performs favourably against the baseline CSM [61] method.

Finally, we show an ablation study of the probability and vertex-based semantic consistency constraints in Table 3.4, where both constraints contribute to the reconstruction network.

3.6 Failure Case and Limitations

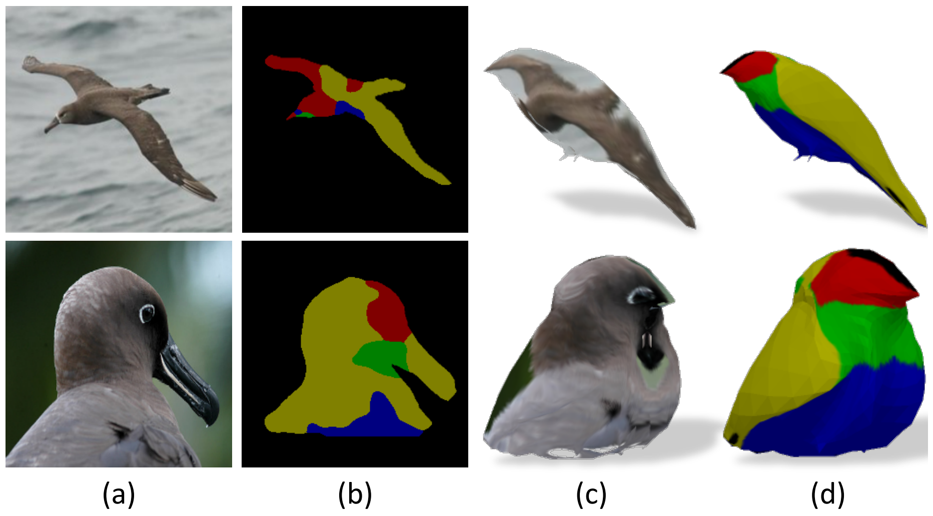


Figure 3.17: Failure cases. (a) Input images. (b) Semantic part segmentations predicted by the SCOPS method. (c) Reconstructed meshes. (d) Reconstructed meshes with the canonical semantic UV map.

Our method performs sub-optimally for objects with large concavities and objects with a genus greater than 0, such as horses and chairs. It captures the major shape characteristics of each instance but ignores some details, e.g., the two wings of flying birds, and the legs of zebras or horses are not separated, as shown in Figure 3.8 and Figure 3.9. Moreover, our method utilizes the SCOPS method to provide semantic part segmentation, and so it suffers when the semantic part segmentation is not accurate, as shown in the first row of Figure 3.17 or if the SCOPS model fails to discover meaningful parts for a certain category, such as airplanes, as shown in the supplementary document of [42]. We leave these failure cases and limitations to future works.

3.7 Conclusion

In this work, we learn a model to reconstruct 3D shape, texture and camera pose from single-view images, with only a category-specific collection of images and silhouettes as supervision. The self-supervised framework enforces semantic consistency between the reconstructed meshes and images and largely reduces ambiguities in the joint prediction of 3D shape and camera pose from 2D observations. It also creates a category-level template and a canonical semantic UV map, which capture the most representative shape characteristics and semantic parts of objects in each category, respectively. Experimental results demonstrate the efficacy of our proposed method in comparison to the state-of-the-art supervised category-specific reconstruction methods.

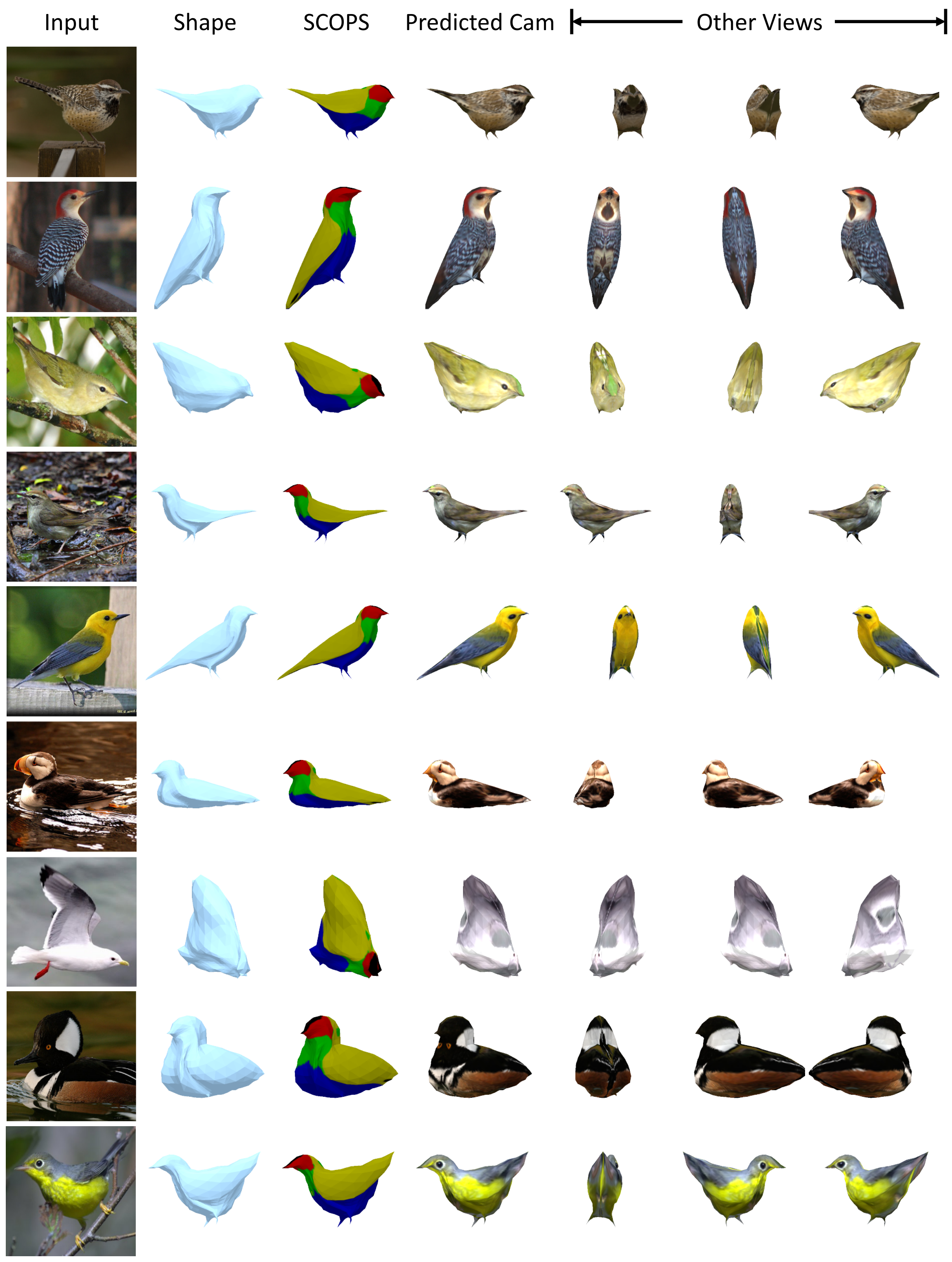


Figure 3.11: More qualitative results of birds.

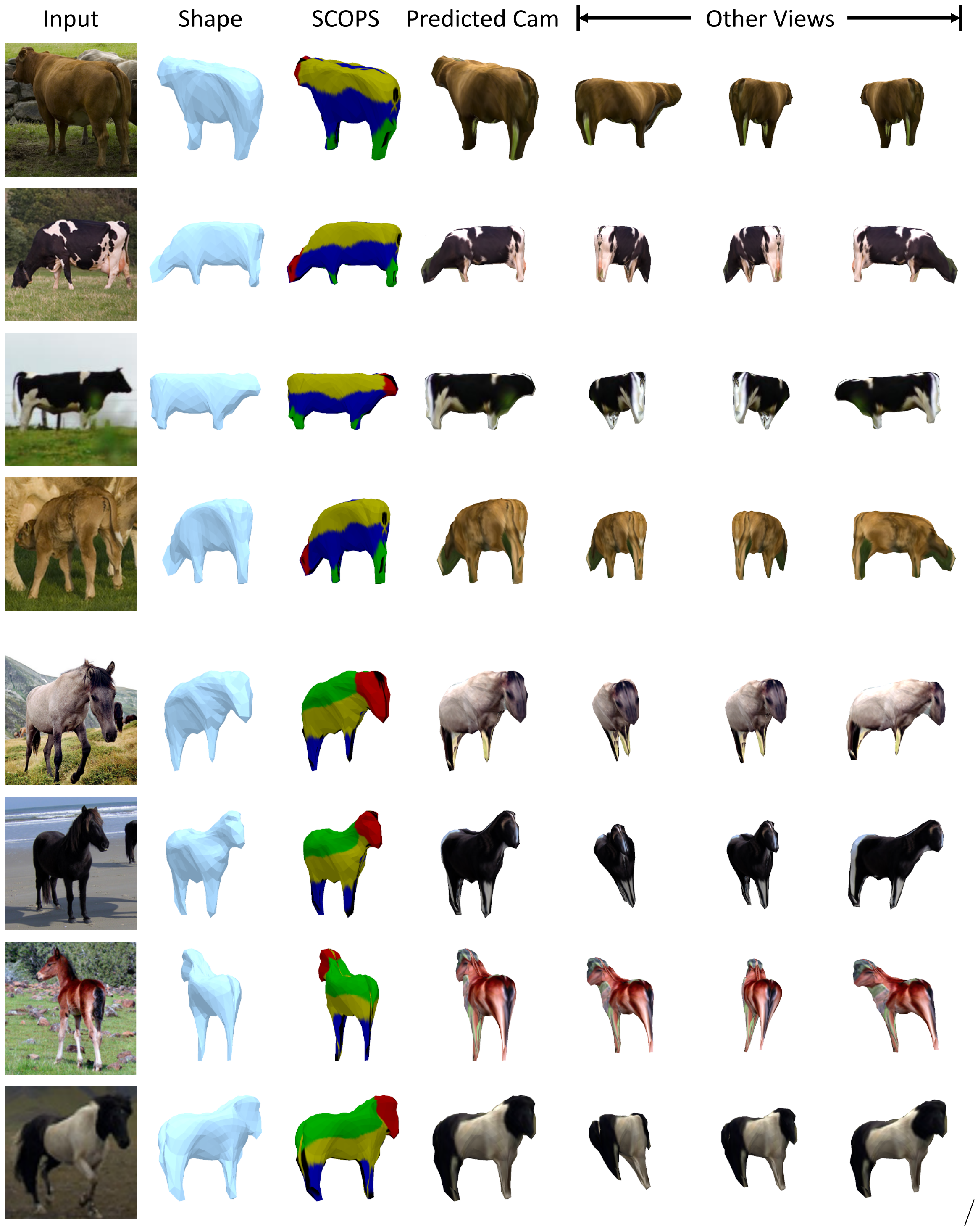


Figure 3.12: More qualitative results of horses and cows.

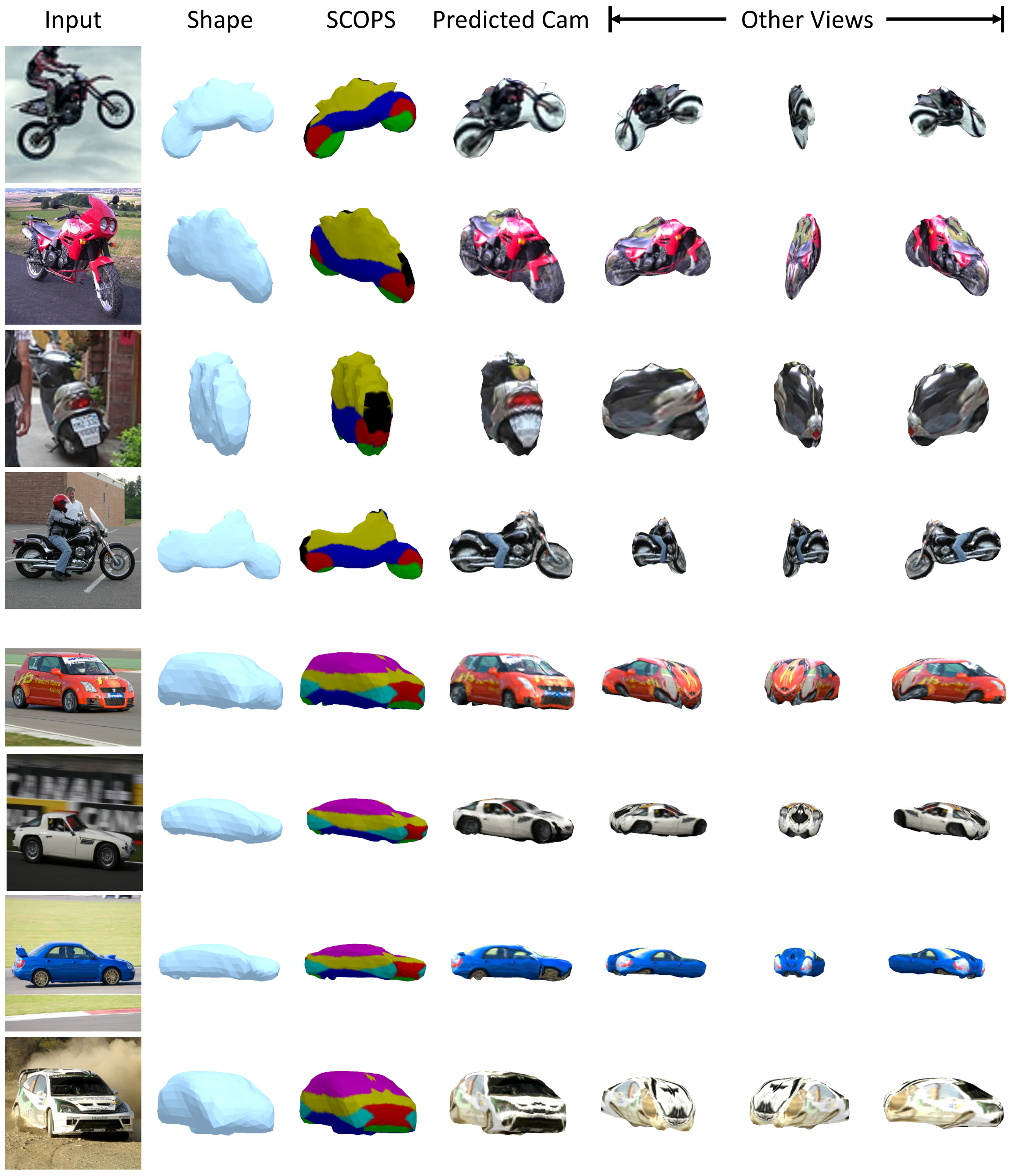


Figure 3.13: More qualitative results of motorbikes and cars.

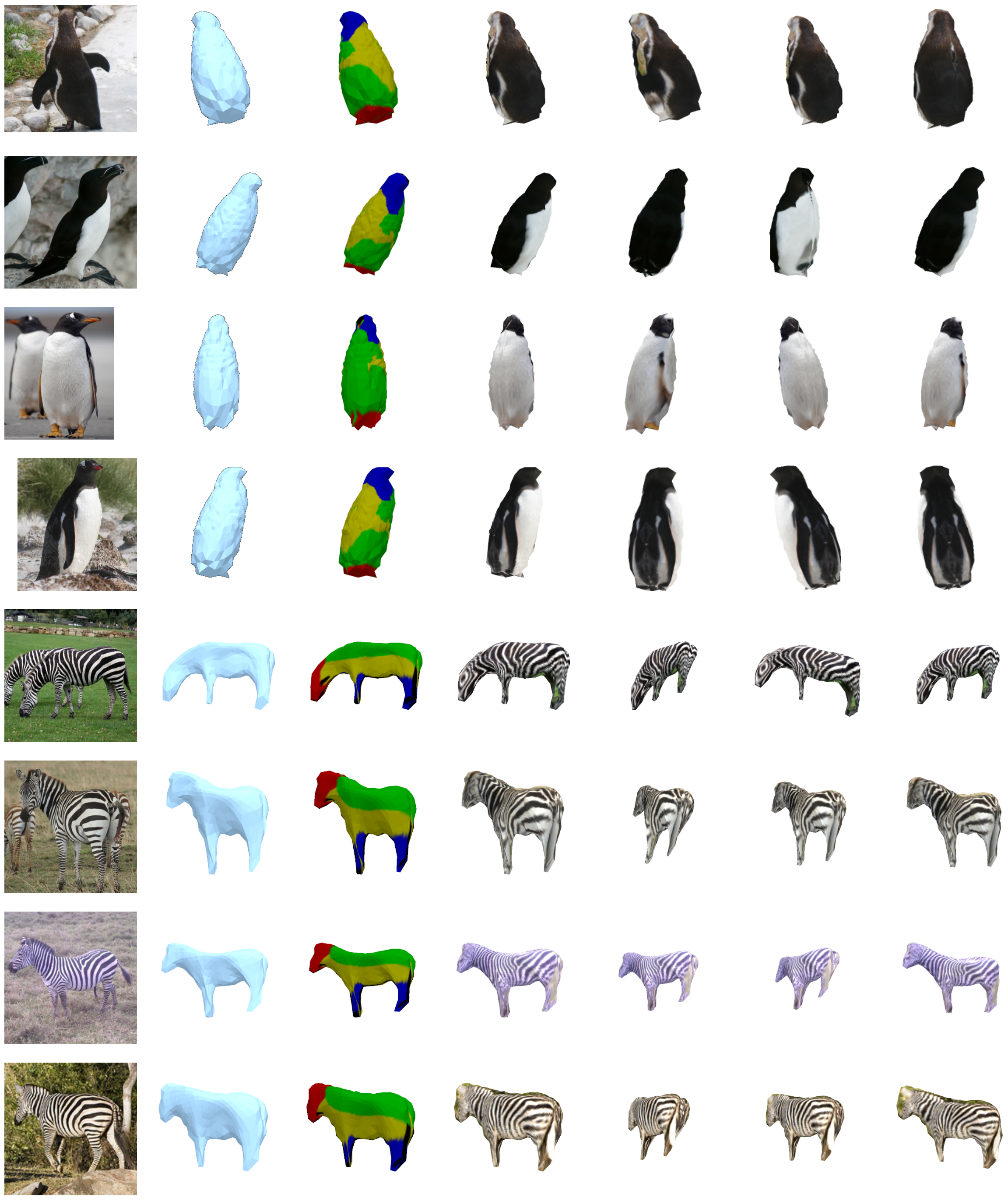


Figure 3.14: More qualitative results of zebras and penguins.

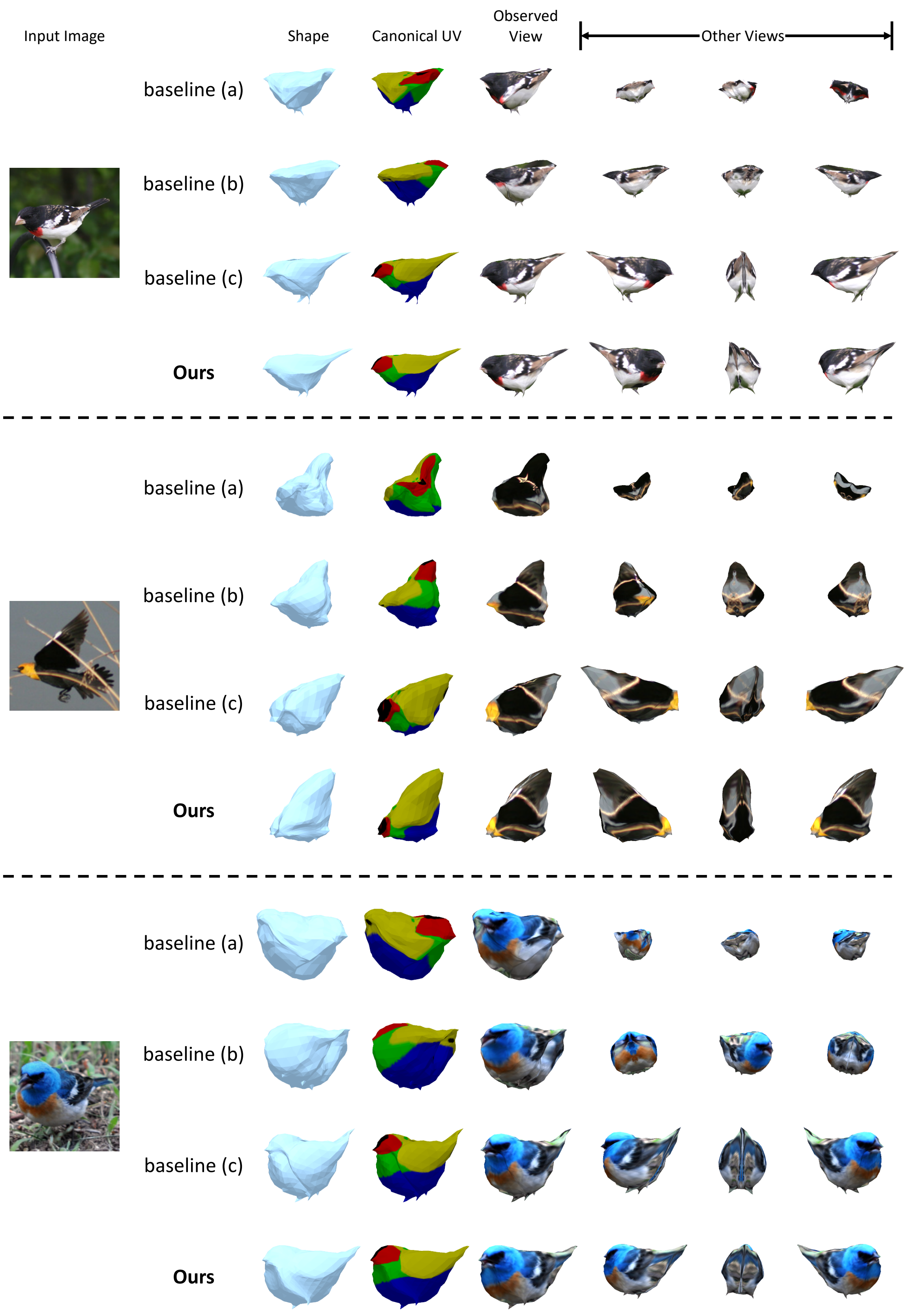


Figure 3.15: Visualization of the contribution of each module. The settings of baselines (a), (b), (c) can be found in Table 3.3

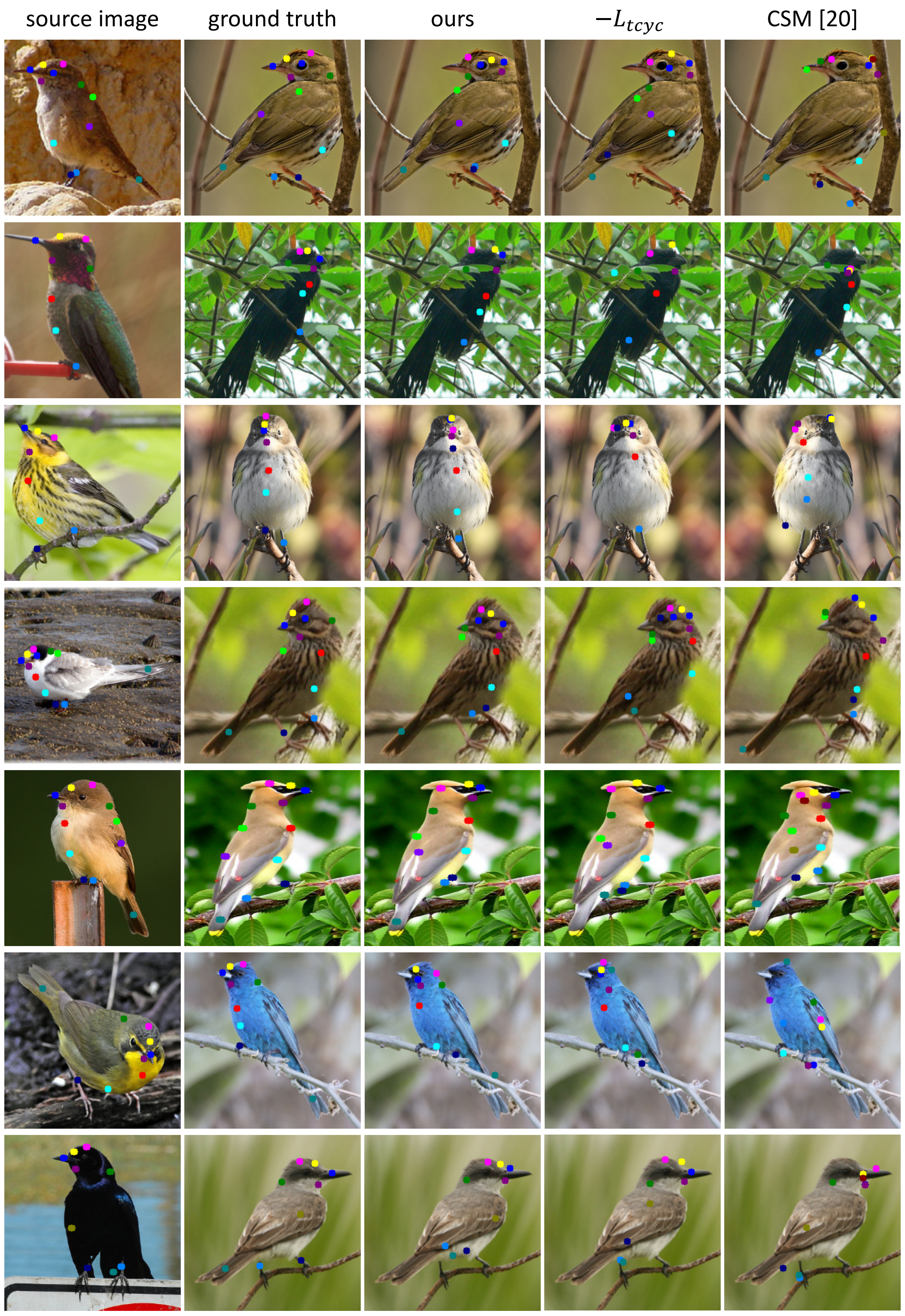


Figure 3.16: Visualization of keypoint transfer using texture flow.

Chapter 4

Mesh Reconstruction from Videos via Temporal Correspondence

4.1 Overview

In this chapter, we present an algorithm to reconstruct temporally consistent 3D meshes of deformable object instances from videos in the wild. Without requiring annotations of 3D mesh, 2D keypoints, or camera pose for each video frame, we pose video-based reconstruction as a self-supervised online adaptation problem applied to any incoming test video. We first learn a category-specific 3D reconstruction model from a collection of single-view images of the same category that jointly predicts the shape, texture, and camera pose of an image. Then, at inference time, we adapt the model to a test video over time using self-supervised regularization terms that exploit temporal consistency of an object instance to enforce that all reconstructed meshes share a common texture map, a base shape, as well as parts. We demonstrate that our algorithm recovers temporally consistent and reliable 3D structures from videos of non-rigid objects including those of animals captured in the wild – an extremely challenging task rarely addressed before.

4.2 Introduction

When we humans try to understand the object shown in Figure 4.1(a), we instantly recognize it as a "duck". We also instantly perceive and imagine its shape in the 3D world, its viewpoint, and its appearance from other views. Furthermore, when we see it in a video, its 3D structure and deformation become even more apparent to us. Our ability to perceive the 3D structure of objects contributes vitally to our rich understanding of them.

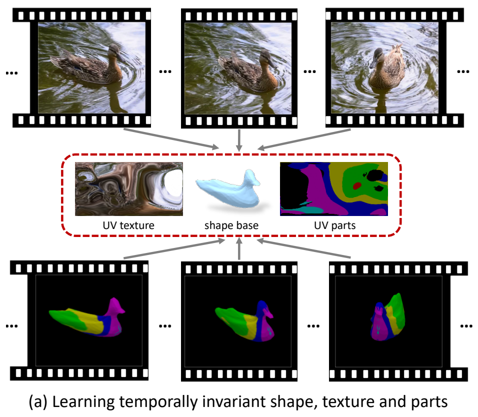
While 3D perception is easy for humans, 3D reconstruction of deformable objects remains a very challenging problem in computer vision, especially for objects in the wild. For learning-based algorithms, the key bottleneck is the lack of supervision. It is extremely challenging to collect 3D annotations such as 3D shape and camera pose [15, 53]. Consequently, existing research mostly focuses on limited domains (e.g., rigid objects [71], human bodies [50, 147] and faces [136]) for which 3D annotations can be captured in constrained environments. However, these approaches do not generalize well to non-rigid objects captured in naturalistic environments (e.g., animals).

In non-rigid structure from motion methods [4, 86], the 3D structure can be partially recovered from correspondences between multiple viewpoints, which are also hard to label. Due to constrained environments and limited annotations, it is nearly impossible to generalize these approaches to the 3D reconstruction of non-rigid objects (e.g., animals) from images and videos captured in the wild.

Instead of relying on 3D supervision, weakly supervised or self-supervised approaches have been proposed for 3D mesh reconstruction. They use annotated 2D object keypoints [49], category-level templates [61, 60] or silhouettes [68]. However, to scale up learning with 2D annotations to hundreds of thousands of images is still non-trivial. This limits the generalization ability of current models to new domains. For example, a 3D reconstruction model trained on single-view images, e.g., [49], produces unstable and erratic predictions for video data. This is unsurprising, due to perturbations over time. However, the temporal signal in videos should provide us an advantage instead of a disadvantage, as recently shown on the task of optimizing a 3D *rigid* object mesh w.r.t. a particular video [154, 71]. The question is, can we also take advantage of the redundancy in temporal sequences as a form of self-supervision in order to improve the reconstruction of dynamic non-rigid objects?

In this work, we address this problem with two important innovations. First, we strike a balance between model generalization and specialization. That is, we train an image-based network on a set of images, while at test time we adapt it online to an input video of a particular instance. Test-time training [108] is non-trivial since no labels are provided for the video. The key is to introduce self-supervised objectives that can continuously improve the model. To do so, we exploit the UV texture space, which provides a parameterization that is invariant to object deformation. We encourage the sampled texture, as well as a group of object parts, to be consistent among all the individual frames in the UV space, as shown in Figure 4.1(a). Using this constraint of temporal consistency, the recovered shape and camera pose are stabilized considerably and are adapted to the current video.

One bottleneck of existing image-based 3D mesh reconstruction methods [49, 68] is that the predicted shapes are assumed to be symmetric. This assumption does not hold for most non-rigid animals, e.g., birds tilting their heads, or walking horses, etc. Our second innovation is to remove this assumption and to allow the reconstructed



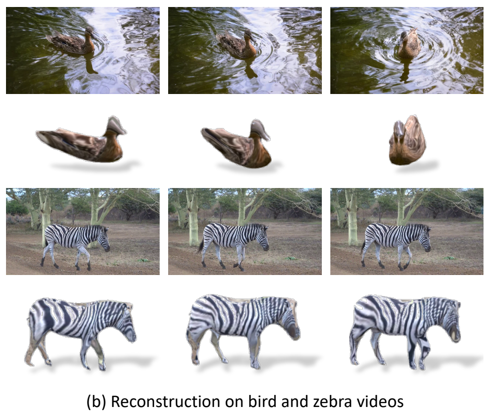


Figure 4.1: By utilizing the consistency of texture, shape and object parts correspondences in videos (red box) as self-supervision signals in (a), we learn a model that reconstructs temporally consistent meshes of deformable object instances in videos in (b).

meshes to fit more complex, non-rigid poses via an as-rigid-as-possible (ARAP) constraint. As another constraint that does not require any labels, we enforce ARAP during test-time training as well, to substantially improve shape prediction. We use two image-based 3D reconstruction models for training (i) a weakly supervised one (i.e., with object silhouettes and 2D keypoints provided), and (ii) a self-supervised one where only object silhouettes are available. The image-based models are then adapted to in-the-wild bird and zebra videos collected from the internet. We show that for both models, our innovations lead to an effective and robust approach to deformable, dynamic 3D object reconstruction of non-rigid objects captured in the wild.

4.3 Related Work

3D object reconstruction from images. A triangular mesh has long been used for object reconstruction [49, 53, 74, 51, 125, 89, 132]. It is a memory-efficient representation with vertices and faces, and is amenable to differentiable rendering techniques [53, 74]. The task of 3D reconstruction entails the simultaneous recovery of the 3D shape, texture, and camera pose of objects from 2D images. It is highly ill-posed due to the inherent ambiguity of correctly estimating both the shape and camera pose together. A major trend of recent works is to gradually reduce supervision from 3D vertices [15, 132, 125], shading [37], or multi-view images [142, 53, 133, 100, 71] and move towards weakly supervised methods that instead use 2D semantic keypoints [49], or a category-level 3D template [61]. This progress makes the reconstruction of objects, e.g., birds, captured in the wild possible. More recently, self-supervised methods [68, 136, 52] have been developed to further remove the need for annotations. Our method exploits different levels of supervision: weak supervision (i.e., using 2D semantic keypoints) and self-supervision to learn an image-based 3D reconstruction network from a collection of images of a category (Section 4.4.1).

Non-rigid structure from motion (NR-SFM). NR-SFM aims to recover the pose and 3D structure of a non-rigid object, or object deforming non-rigidly over time, solely from 2D landmarks without 3D supervision [4]. It is a highly ill-posed problem and needs to be regularized by additional shape priors [4, 155]. Recently, deep networks [57, 86] have been developed that serve as more powerful priors than the traditional approaches. However, obtaining reliable landmarks or correspondences for videos is still a bottleneck. Our method bears resemblances to deep NR-SFM [86], which jointly predicts camera pose and shape deformation. Differently from them, we reconstruct dense meshes instead of sparse keypoints, without requiring labeled correspondences from videos.

3D object reconstruction from videos. Existing video-based object reconstruction methods mostly focus on specific domains, e.g., videos of faces [24, 114] or human bodies [117, 2, 19, 50, 147], where dense labelling is possible [119]. To augment video labels, [50] formulates dynamic human mesh reconstruction as an omni-supervision task, where a combination of labeled images and videos with pseudo-ground truth are used for training. For human video-based 3D pose estimation, [92] introduces semi-supervised learning to leverage unlabeled videos with a self-supervised component. Dealing with specific application domains, all the aforementioned works rely on predefined shape priors, such as a parametric body model (e.g., SMPL [78]) or a morphable face model. While our work also exploits unlabeled videos, we do not assume any predefined shape prior, which, practically, is hard to obtain for the majority of objects captured in the wild.

Optimization-based methods. Optimization-based methods have also been extensively explored for scene or object reconstruction from videos. Several works [151, 123, 101, 93] first obtain a single-view 3D reconstruction and then optimize the mesh and skeletal parameters. Another line of methods is developed to optimize the weights of deep models instead, to render more robust results for a video of a particular instance [117, 71, 80, 156]. Our method falls into this category. While [117] enforces consistency between observed 2D and a re-projection from 3D, [71, 80] take a further step and encourage consistency between frames via a network that inherently encodes an entire video into an invariant representation. In this work, instead of limiting to rigid objects as [71], or depth estimation as [80], we recover dynamic meshes from videos captured in the wild – a much more challenging problem that is rarely explored.

4.4 Approach

Our goal is to recover coherent sequences of mesh shapes, texture maps and camera poses from unlabeled videos, with a two-stage learning approach: (i) first, we learn a 3D mesh reconstruction model on a collection of single-view images of a category, described in Section 4.4.1; (ii) at inference time, we adapt the model to fit the sequence via temporal consistency constraints, as described in Section 4.4.2. We focus on the weakly-supervised setting in Section 4.4.1 and 4.4.2, where both silhouettes and keypoints are annotated in the image dataset. We then describe how to generalize the approach to a self-supervised setting, where only silhouettes are available in the image dataset in Section 4.4.3.

Notations. We represent a textured mesh with $|V|$ vertices ($V \in \mathbb{R}^{|V|\times 3}$), $|F|$ faces ($F \in \mathbb{R}^{|F|\times 3}$) and a UV texture image ($I_{\text{uv}} \in \mathbb{R}^{H_{\text{uv}}\times W_{\text{uv}}\times 3}$) of height H_{uv} and width W_{uv}. Similarly to [49], we use a weak perspective transformation to represent the camera pose $\theta \in \mathbb{R}^7$ of an input image. We denote $\mathcal{R}(\cdot)$ as a general projection, which can represent (i) a differentiable renderer [74, 53] to render a mesh to a 2D silhouette as $\mathcal{R}(V, \theta)$, or a textured mesh to an RGB image as $\mathcal{R}(V, \theta, I_{\text{uv}})$ (we omit mesh faces F for conciseness); (ii) or a projection of a 3D point v to the image space as $\mathcal{R}(v, \theta)$. The Soft Rasterizer [74] is used as the differentiable renderer in this work.

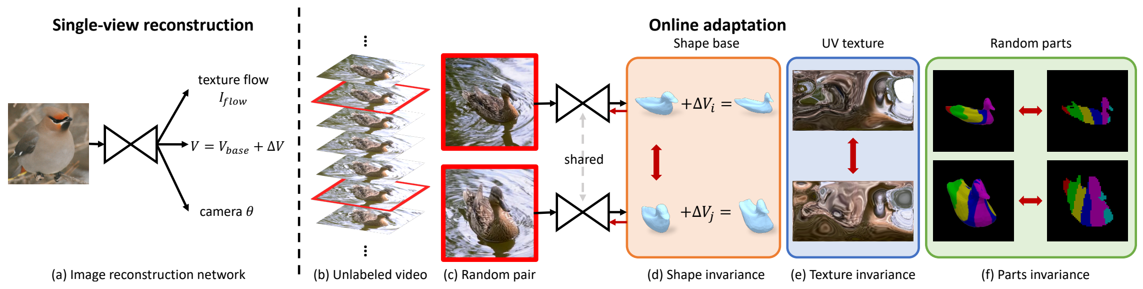


Figure 4.2: Overview. We show the single-view image reconstruction network on the left and the test-time training procedure to adapt it to a video on the right. Bold red arrows indicate invariance constraints in Section 4.4.2.

4.4.1 Single-view Mesh Reconstruction

In the first stage, we train a network with a collection of category-specific images that jointly estimates the shape, texture, and camera pose of an input image. Similarly to [49], we predict a texture flow $I_{\text{flow}} \in \mathbb{R}^{H_{\mathbf{uv}} \times W_{\mathbf{uv}} \times 2}$ that maps pixels from the input image to the UV space. A predefined UV mapping function Φ [41, 49] is then used to map these pixels from the UV space to the mesh surface. With a differentiable renderer [74], we train the network with supervision from object silhouettes, texture, and the Laplacian objectives as in [49, 68]. More details of learning texture and camera pose can be found in [49]. An overview of our reconstruction framework is shown in Figure 4.2(a).

Recovering asymmetric shapes. We propose a novel shape reconstruction module as shown in Figure 4.2(a). The key idea is to remove the symmetry requirement of object shapes, which is employed by many prior works [49, 68]. This is particularly important for recovering dynamic meshes in sequences, e.g., when a bird rotates its head as shown in Figure 4.5, its mesh is no longer mirror-symmetric. Prior works [49, 68] model object shape by predicting vertex offsets from a jointly learned 3D template. Simply removing the symmetry assumption for the predicted vertex offsets leads to excessive freedom in shape deformation, e.g., see Figure 4.5(f). To resolve this, we learn a group of N_b shape bases $\{V_i\}_{i=1}^{N_b}$, and replace the template by a weighted combination of them, denoted as the base shape V_{base}. Compared to a single mesh template, the base shape V_{base} is more powerful in capturing the object's identity and saves the model from predicting large motion deformation, e.g., of deforming a standing bird template to a flying bird. The full shape reconstruction can be obtained by:

$$V = V_{\text{base}} + \Delta V, \qquad V_{\text{base}} = \sum_{i=1}^{N_b} \beta_i V_i, \tag{4.1}$$

where the ΔV encodes the object's asymmetric non-rigid motion and $\{\beta_i\}_{i=1}^{N_b}$ are learned coefficients.

The computation of our shape bases is inspired by parametric models [78, 157, 158], where the basis shapes are extracted from an existing mesh dataset [102] or toy scans [158]. However, we make our model completely free of 3D supervision and obtain the bases by applying K-Means clustering to all meshes reconstructed by CMR [49]. We use each cluster center as a basis shape in our model.

Keypoint re-projection. In the weakly-supervised setting, the 2D keypoints are provided that semantically associate different instances. When projected onto the mesh surface, the same semantic keypoint (e.g., the tail keypoint in the orange circle in Figure 4.3(a)) for different object instances should be matched to the same face on the mesh (the tail keypoint in the orange circle in Figure 4.3(d)). To model the mapping between the 3D mesh surface and the 2D keypoints, prior work [49] learns an affinity matrix that describes the probability of each 2D keypoint mapping to each vertex on the mesh. The affinity matrix is shared among all instances and is independent of individual shape variations. However, this approach is sub-optimal because: (i) Mesh vertices are a subset of discrete points on a continuous mesh surface and so their weighted combination defined by the affinity matrix may not lie on it, leading to inaccurate mappings of 2D keypoints. (ii) The mapping from the image space to the mesh surface described by the affinity matrix, in our case, however, is already modeled by the texture flow. Hence, it is potentially redundant to learn both of them independently.

In this work, we re-utilize texture flow to map 2D keypoints from each image to the mesh surface. We first map each 2D keypoint to the UV space that is independent of shape deformation (Figure 4.3(b)). Ideally, each semantic keypoint from different instances should map to the same point in the UV space as discussed above. In

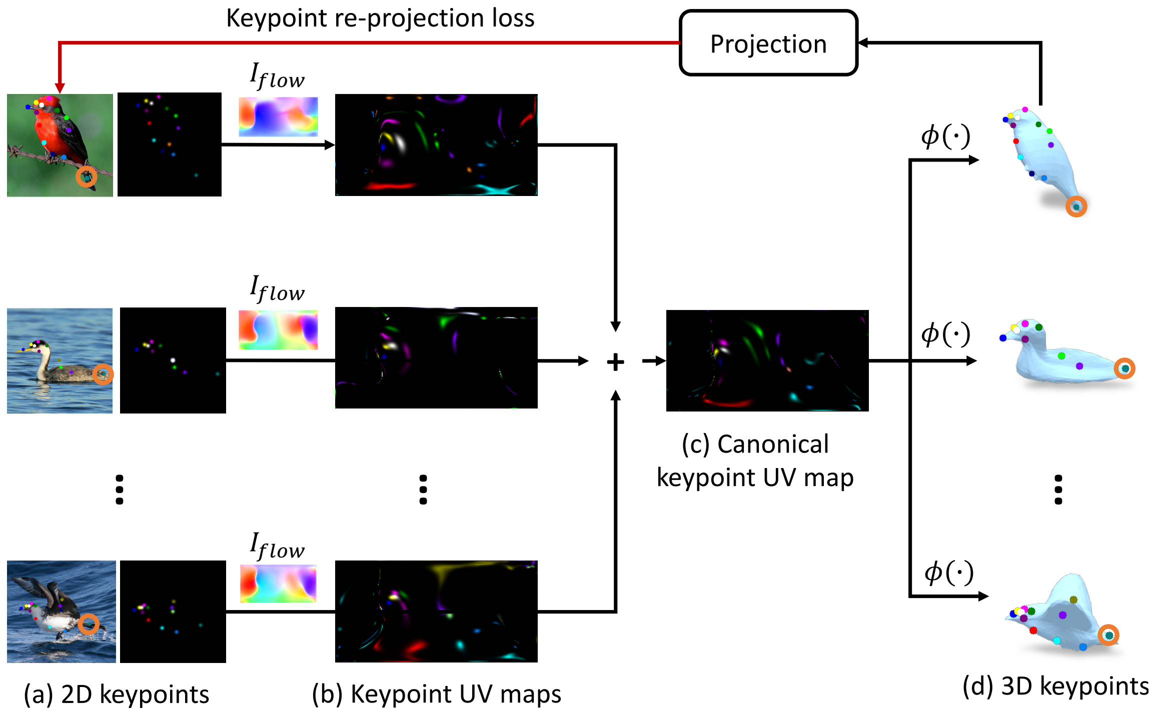


Figure 4.3: 3D canonical keypoints computation: (a) annotated 2D keypoints and their location heatmaps; (b) keypoint heatmaps mapped to the UV space using learned texture flows; (c) aggregated canonical keypoint heatmaps in the UV space; (d) canonical keypoints on different instance mesh surface. $\Phi(\cdot)$ is the UV mapping function discussed in Section 4.4.1

practice, this does not hold due to inaccurate texture flow prediction. To accurately map each keypoint to the UV space, we compute a canonical keypoint UV map as shown in Figure 4.3(c) by: (i) mapping the keypoint heat map in Figure 4.3(a) for each instance to the UV space via its predicted texture flow, and (ii) aggregating these keypoint UV maps in Figure 4.3(b) across all instances to eliminate outliers caused by incorrect texture flow prediction.

We further utilize the pre-defined UV mapping function Φ discussed above to map each semantic keypoint from the UV space to the mesh surface. Given the 3D correspondence (denoted as K^i_{3D}) of each 2D semantic keypoint K^i_{2D}, the keypoint re-projection loss enforces the projection of the former to be consistent with the latter by:

$$L_{kp} = \frac{1}{N_k} \sum_{i=1}^{N_k} \left\| \mathcal{R}(K_{3D}^i, \theta) - K_{2D}^i \right\|, \tag{4.2}$$

where N_k is the number of keypoints.

As-rigid-as-possible (ARAP) constraint. Without any pose-related regularization, the predicted motion deformation ΔV often leads to erroneous random deformations and spikes as shown in Figure 4.5(f), which do not faithfully describe the motion of a non-rigid object. Therefore, we introduce an as-rigid-as-possible (ARAP) constraint [106, 33] to encourage rigidity of local transformations and the preservation of the local mesh structure. Instead of solving the optimization in [106, 33], we reformulate it as an objective that ensures that the predicted shape V is a locally rigid transformation from the predicted base shape V_{base} by:

$$L_{\text{arap}}(V_{\text{base}}, V) = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| (V^i - V^j) - R_i(V_{\text{base}}^i - V_{\text{base}}^j) \right\|, \tag{4.3}$$

where $\mathcal{N}(i)$ represents the neighboring vertices of a vertex i, w_{ij} and R_i are the cotangent weight and the best approximating rotation matrix, respectively, as described in [106].

Objectives summary for image reconstruction model We summarize the objectives used in the single-view reconstruction model (Figure 4.6(a)) as follows: (i) *foreground mask loss:* a negative intersection over union objective between rendered and ground truth silhouettes [49, 68, 51]; (ii) *foreground RGB texture loss:* a perceptual metric [49, 68, 148] between rendered and input RGB images; (iii) *mesh smoothness:* a Laplacian constraint [49, 68] to encourage smooth mesh reconstruction; (iv) *keypoint re-projection loss:* as discussed in Section 4.4.1; and (v) *the ARAP constraint:* described in Section 4.4.1. The weight for each objective is set to 3.0, 3.0, 0.0008, 5.0 and 10.0.

4.4.2 Online Adaptation for Videos

Applying the image-based model developed in Section 4.4.1 independently to each frame of an unseen video usually results in inconsistent mesh reconstruction (Fig-

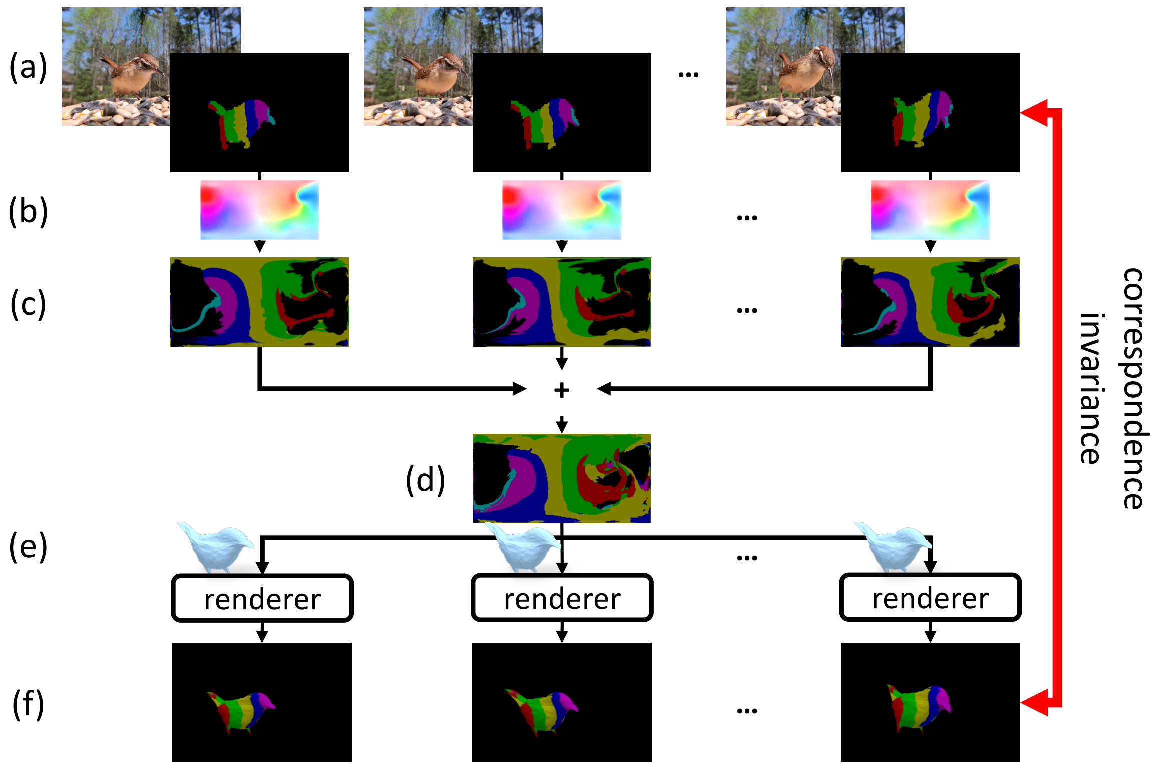


Figure 4.4: Part correspondence constraint. (a) Input frame and part propagations. (b) Predicted texture flows. (c) Part UV map. (d) Aggregated video-level part UV map. (e) Base shape and differentiable renderer. (f) Part rendering.

ure 4.8(a)), mainly due to the domain differences in video quality, lighting conditions, etc. In this section, we propose to perform online adaptation to fit the model to individual test video, which contains a single object instance that moves over time. Inspired by the keypoint re-projection constraint described in Section 4.4.1, we resort to the UV space, where the (i) RGB texture, and (ii) object parts of an instance should be constant when mapped from 2D via the predicted texture flow, and invariant to shape deformation. By enforcing the predicted values for (i) and (ii) to be consistent in the UV space across different frames, the adapted network is regularized to generate coherent reconstructions over time. In the following, we describe how to exploit the aforementioned temporal invariances as self-supervisory signals to tune the model.

Part correspondence constraint. We propose a part correspondence constraint that utilizes corresponding parts of each video frame to facilitate camera pose learning. The idea bears resemblance to NR-SFM methods [86, 57], but in contrast, we do not know the ground truth correspondence between frames. Instead, we resort to an unsupervised video correspondence (UVC) method [69]. The UVC model learns an affinity matrix that captures pixel-level correspondences among video frames. It can be used to propagate any annotation (e.g, segmentation labels, keypoints, part labels, etc.), from an annotated keyframe to the other unannotated frames. In this work, we generate part correspondence within a clip: we ”paint” a group of random parts on the object, e.g., the vertical stripes in Figure 4.2(f), on the first frame and propagate them to the rest of the video using the UVC model.

Given the propagated part correspondences in all the frames, we map them to the UV space via the texture flow, similar to our approach for the canonical keypoint map (Section 4.4.1). We then average all part UV maps to obtain a video-level part UV map ("UV parts" in Figure 4.1(a)) for the object depicted in the video. We map the part UV map to each individually reconstructed mesh, and render it via the predicted camera pose of each frame (see Figure 4.1(a), bottom). Finally, we penalize the discrepancy between the parts being rendered back to the 2D space, and the propagated part maps, for each frame. As the propagated part maps are usually temporally smooth and continuous, this loss implicitly regularizes the network to predict coherent camera pose and shape over time. In practice, instead of minimizing the discrepancy between the rendered part map and the propagation part map of a frame, we found that it is more robust to penalize the geometric distance between the projections of vertices assigned to each part with 2D points sampled from the corresponding part as:

$$L_c = \sum_{j=1}^{N_f} \sum_{i=1}^{N_p} \frac{1}{|V_i^j|} \text{Chamfer}(\mathcal{R}(V_i^j, \theta^j), Y_i^j), \tag{4.4}$$

where N_f is the number of frames in the video, $N_p = 6$ is the number of parts and V_i^j are vertices assigned to part i. Here we utilize the Chamfer distance because the vertex projections $\mathcal{R}(V_i^j, \theta^j)$ do not strictly correspond one-to-one to the sampled 2D points Y_i^j.

We summarize the part correspondence constraint in details in Figure 4.4. Given the propagated parts in each frame in Figure 4.4(a), we map them to the UV space with the predicted texture flow in Figure 4.4(b) and obtain part UV maps in Figure 4.4(c). By aggregating these part UV maps, i.e., averaging, we minimize noise in each individual part UV map and obtain a video-level part UV map in Figure 4.4(d). This video-level part UV map is shared by all frames in the video. Thus, for each frame, we wrap the video-level part UV map onto the base shape prediction and render it under the predicted camera pose as shown in Figure 4.4(f). Finally, we encourage consistency between part renderings and part propagations, as shown by the red arrow in Figure 4.4. Through the differentiable renderer, the loss implicitly improves both the predicted camera pose.

Texture invariance constraint. Based on the observation that object texture mapped to the UV space should be invariant to shape deformation and stay constant over time, we propose a texture invariance constraint to encourage consistent texture reconstruction from all frames. However, naively aggregating the UV texture maps from all the frames via a scheme similar to the one described for keypoints and parts, leads to a blurry video-level texture map. We instead enforce texture consistency between random pairs of frames, via a swap loss. Given two randomly sampled frames I^i and I^j, we swap their texture maps $I^{\mathbf{i}}_{\text{uv}}$ and $I^{\mathbf{j}}_{\text{uv}}$, and combine them with the original mesh reconstructions V^i and V^j as:

$$L_t = \text{dist}(\mathcal{R}(V^i, \theta^i, I^{\mathbf{j}}_{\text{uv}}) \odot S^i, I^i \odot S^i) + \text{dist}(\mathcal{R}(V^j, \theta^j, I^{\mathbf{i}}_{\text{uv}}) \odot S^j, I^j \odot S^j), \quad (4.5)$$

where S^i and S^j are the silhouettes of frame i and j, respectively and $\text{dist}(\cdot, \cdot)$ is the perceptual metric used in [148, 49, 68].

Base shape invariance constraint. As discussed in Section 4.4.1, our shape model is represented by a base shape V_{base} and a deformation term ΔV, in which the base shape V_{base} intuitively corresponds to the "identity" of the instance, e.g., a duck, or a flying bird, etc. During online adaptation, we enforce the network to predict consistent V_{base} to preserve the identity, via a swapping loss function:

$$L_s = \text{niou}(\mathcal{R}(V^{\mathbf{j}}_{\text{base}} + \Delta V^i, \theta^i), S^i) + \text{niou}(\mathcal{R}(V^{\mathbf{i}}_{\text{base}} + \Delta V^j, \theta^j), S^j), \quad (4.6)$$

where $V_{\text{base}}^{\mathbf{i}}$ and $V_{\text{base}}^{\mathbf{j}}$ are the base shapes for frame i and j; ΔV^i and ΔV^j are the motion deformations for frame i and j; and $\text{niou}(\cdot, \cdot)$ denotes the negative intersection over union (IoU) objective [51, 68]. All other notations are defined in Eq. 4.5.

As-rigid-as-possible (ARAP) constraint. Besides the consistency constraints, we keep the ARAP objective, as discussed in Section 4.4.1, during online adaptation since it also does not require any form of supervision. We found that the ARAP constraint can obviously improve the qualitative results, as visualized for the online adaptation procedure in Figure 4.7.

Online adaption. During inference, we fine-tune the model on a particular video with the invariance constraints discussed above, along with a silhouette and a texture objective, a Laplacian term as in [49, 68], and the ARAP constraint discussed in Section 4.4.1. The foreground masks used for the silhouette and texture objective are obtained by a segmentation model [10] trained with the ground truth foreground masks available for the image collection.

To obtain accurate part propagation of object parts by the UVC [69] model, we employ two strategies. Firstly, we fine-tune all parameters in the reconstruction model on sliding windows instead of all video frames. Each sliding window includes $N_w = 50$ consecutive frames and the sliding stride is set to $N_s = 10$. We tune the reconstruction model for $N_t = 40$ iterations with frames in each sliding window. We show the test-time tuning process in Figure 4.6(b). Within each sliding window, we encourage the consistency of UV texture, UV parts as well as base shape of all frames. Secondly, instead of "painting" random parts onto the first frame and propagating them to the rest of the frames sequentially in a window, we "paint" random parts onto the *middle* frame (i.e. the $\frac{N_w}{2}$th frame) in the window and propagate the parts backward to the first frame as well as forward to the last frame in the window. This strategy improves the propagation quality by decreasing the propagation range to half of the window size.

We summarize the objectives used in the online adaptation process (See Figure 4.6(b)) in the following. Since it is feasible to predict a segmentation mask via a pretrained segmentation model, we make use of the predicted foreground mask and compute the (i), (ii), and (iii) losses (mentioned above) similarly to the image-based training. We also adopt the the ARAP constraint described in Section 4.4.1, and the three invariance constraints as discussed in Section 4.4.2 for online adaptation. The weight for each objective is set to 0.1, 0.5, 0.0006, 2.0 and 2.0 (texture invariance), 1.0 (part correspondence), 1.0 (base shape invariance).

4.4.3 Self-supervised Setting

Our model can also be easily generalized to a self-supervised setting in which keypoints are not provided for the image datasets. In this setting, the template prior as well as camera poses in the CMR method [49] computed from the keypoints are no longer available. This self-supervised setting is trained differently from the weakly-supervised one in the following: (i) The first stage still assumes shape symmetry to ensure stability when training without keypoints. (ii) It learns a single template from scratch via the progressive training in [68]. (iii) We train this model without the keypoints re-projection and the ARAP constraints in Section 4.4.1. (iv) Without the shape bases, the base shape invariance constraint is thus removed in the online adaptation procedure. Other structure and training settings in the self-supervised model are the same as in the weakly-supervised model discussed in Section 4.4.1 and 4.4.2.

4.5 Experiments

We conduct experiments on animals, i.e., birds and zebras. We evaluate our contributions in two aspects: (i) the improvement of single-view mesh reconstruction, and (ii) the reconstruction of a sequence of frames via online adaptation. Due to the lack of ground truth meshes for images and videos captured in the wild, we evaluate the reconstruction results via mask and keypoint re-projection accuracy, e.g., we follow, and compare against [49] to evaluate the model trained on the image dataset. We also describe a new bird video dataset that we curate and evaluate the test-time tuned model on it in the following.

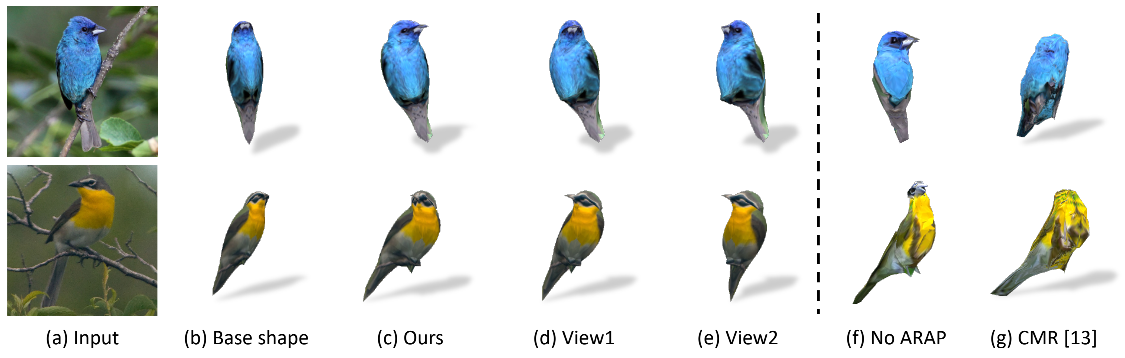


Figure 4.5: Mesh reconstructions from single-view images. All meshes are visualized from the predicted camera pose except for (d) and (e), where the reconstructions in (c) are visualized from two extra views. Meshes in (f) are reconstructed by a model trained without the ARAP constraint.

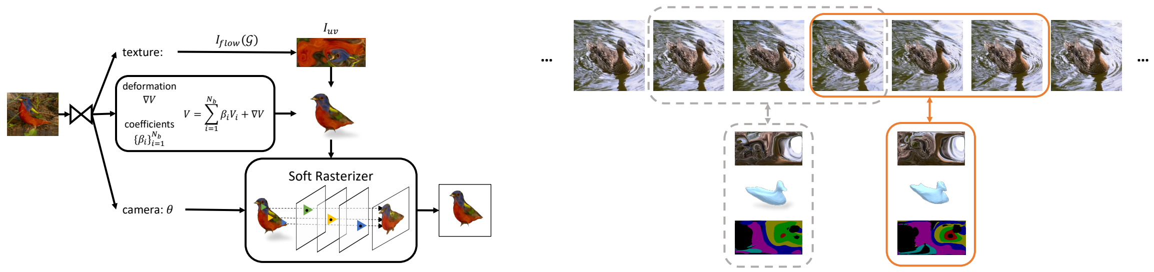
4.5.1 Experimental Settings

Datasets. We first train image reconstruction models, discussed in Section 4.4.1, for the CUB bird [122] and the synthetic zebra [157] datasets. For test-time adaptation on videos, we collect a new bird video dataset for quantitative evaluation. Specifically, we collect 19 slow-motion, high-resolution bird videos from YoutTube, and 3 bird videos from the DAVIS dataset [55]. For each slow-motion video collected from the Internet, we apply a segmentation model [10] trained on the CUB bird dataset [122] to obtain its foreground segmentation for online adaptation.

Evaluation metrics. We evaluate the image-based model on the testing split of the CUB dataset. Note that for keypoint re-projection, instead of using the keypoint assignment matrix in [49], we apply the canonical keypoint UV map to obtain the 3D keypoints (Section 4.4.1). For the video dataset, we annotate frame-level object masks and keypoints via a semi-automatic procedure. We train a segmentation model and a keypoint detector [31] on the CUB dataset. Then, we manually adjust and filter out inaccurate predictions to ensure the correctness of the ground-truth labels. To evaluate the accuracy of mask re-projection, we compute the Jacaard index $\mathcal{J}$ (IoU) and contour-based accuracy $\mathcal{F}$ proposed in [95], between the rendered masks and the ground truth silhouettes of all annotated frames.

In addition, to further quantitatively evaluate shape reconstruction quality, we animate a synthetic 3D bird model and create a video with 520 frames in various poses such as flying, landing, walking etc., as shown in Figure 4.12. We then compare the predicted mesh with the ground truth mesh using Chamfer distance every 10 frames.

Network architecture. For fair comparisons to the baseline method [49], we train our model using the same network as [49], i.e., ResNet18 [36] with batch normalization layers [45] as the encoder, we call this model "ACMR" in the following, which is short for "asymmetric CMR". However, ACMR cannot be well adapted to test videos due to the batch normalization layers and the domain gap between images and videos (see Table 4.2(d)). Thus, we train a variant model where we use ResNet50 [36] as our encoder and replace all the batch normalization layers in the network with group normalization layers [138]. We call this variant model "ACMR-vid". All test-time training is carried out on the ACMR-vid model unless otherwise specified. In



(a) Image reconstruction model

(b) Illustration of the sliding window scheme

Figure 4.6: Frameworks. For the purposes of illustration only, we show the test-time tuning process in (b) with a sliding widow size of 3 and sliding stride of 2. In all our experiments, we use a sliding window size of 50 and stride of 10. The gray dashed box shows the previous sliding window while the orange box shows the current sliding window.

Figure 4.6(a), we show details of our single-view reconstruction network. Given an input image, the network jointly predicts texture, shape and camera pose. By utilizing a differentiable renderer [74], we are able to utilize 2D supervision, i.e. silhouettes and input images.

Network training. Taking the weakly-supervised setting as an example, to train the image reconstruction model, we first warm up the model without the motion deformation branch, the keypoint re-projection objective, or the ARAP constraint for 200 epochs. This warm-up process effectively avoids the trivial solution where the model solely depends on the motion deformation branch for shape deformation while ignoring the base shape branch. We then train the full image reconstruction network with all objectives for another 200 epochs.

Training on zebra images and videos We adopt a different scheme to train a single-view reconstruction model on zebras: (i) since natural zebra images labeled with keypoints are not publicly available, we adopt a synthetic dataset [156], (ii) zebras have more complex shapes with large concavities. Therefore, it is not suitable to learn the shape by deforming from a sphere primitive. Instead, we utilize a readily available zebra mesh as a template and learn motion deformation on top of it. We first train an image reconstruction model using the synthetic dataset provided by [156]. Similarly as [156], we utilize the silhouettes, keypoints, texture maps as well as partially available UV texture flows as supervision. For shape reconstruction, instead of the utilizing the SMAL parametric model [158], we use the proposed shape module, i.e. combination of base shapes and motion deformation. Due to the limited motion of zebras, we only use one base shape, which is a readily available zebra mesh with 3889 vertices and 7774 faces. For camera pose prediction, we use the perspective camera pose discussed in Section 4.4.1 as well as in [49]. Due to the limited capacity of a single UV texture map, we also model the texture map by cutting the UV texture map into four pieces and stitch them together similarly as in [156]. We note that this “cutting and stitching” operation does not influence the mapping and aggregation of the part UV maps discussed in Section 4.4.2.

4.5.2 Qualitative Results

Mesh reconstruction from images. In Figure 4.5 and Figure 4.18, we show visualizations of reconstructed bird meshes from single-view images. Thanks to the “motion deformation branch” discussed in Section 4.4.1, the proposed ACMR model is able to capture asymmetric motion of the bird such as head rotation (Figure 4.5(c)),

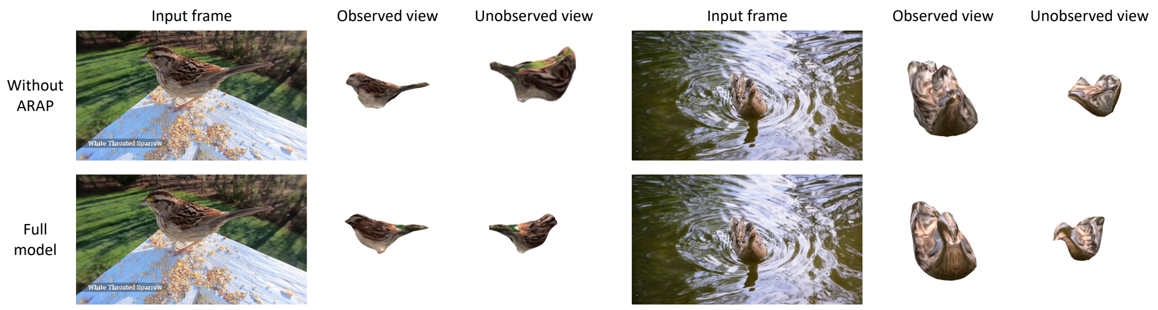


Figure 4.7: Qualitative comparison of online adaptation with/without the ARAP constraint.

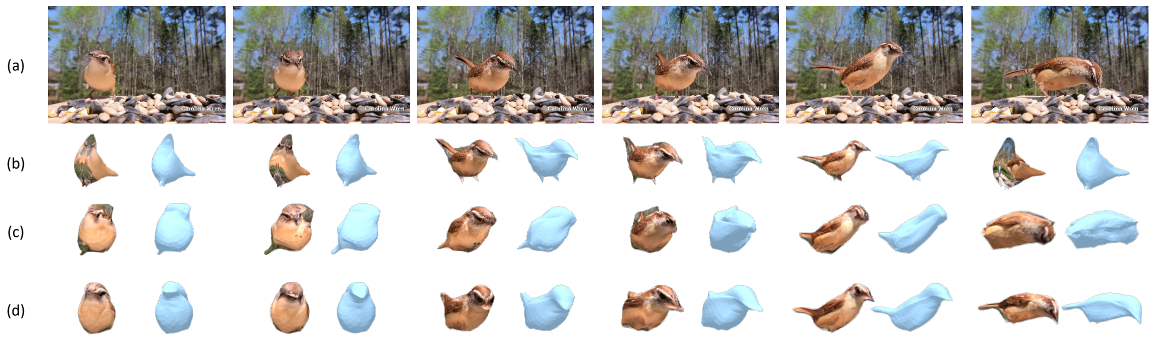


Figure 4.8: Mesh reconstruction from video frames. (a) Input video frames. (b) Reconstruction from the model trained only on single-view images. (c) Reconstruction from the model test-time trained on the video without the invariance constraints in Section 4.4.2. (d) Reconstruction from the proposed video reconstruction model.

which cannot be modeled by the baseline method [49] (Figure 4.5(g)). We also demonstrate mesh reconstruction for the zebra category in Figure 4.17.

Online adaptation on a video. We visualize the reconstructed meshes by our ACMR-vid model for video frames in Figure 4.8. Without online adaptation, the ACMR-vid model independently applied to each frame suffers from a domain gap and shows instability over time (Figure 4.8(b)). With online adaption as discussed in Section 4.4.2, the ACMR-vid model reconstructs plausible meshes for each video frame as shown in Figure 4.8(c) and (d). Specifically, to demonstrate the effectiveness of the proposed invariance constraints, we also show reconstructions of an ACMR-vid model trained without all the invariance constraints in Figure 4.8(c), which predicts less reliable shape, camera pose as well as texture compared to our full ACMR-vid model. Finally, we visualize the effectiveness of ARAP for online adaptation

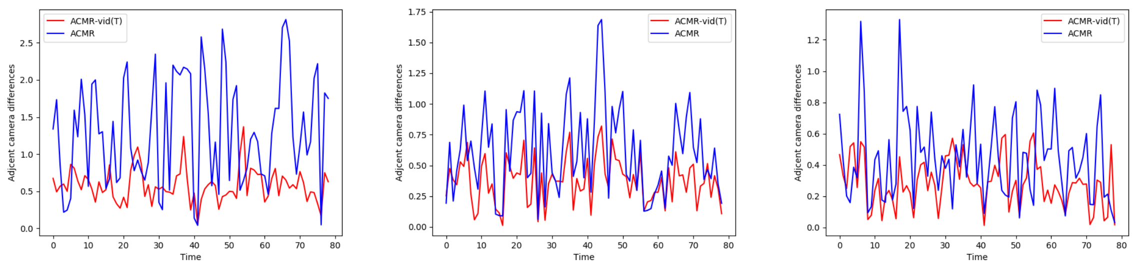


Figure 4.9: Camera stability visualization. We visualize differences between adjacent camera pose predictions. The blue and red lines represent the model trained only with images and the test-time tuned model, respectively.

in Figure 4.7. Without this constraint, the reconstructed meshes are less plausible, especially from unobserved views.

Camera Pose Stability To visually demonstrate the effectiveness of the test-time training process that stabilizes camera pose prediction, we visualize the differences of camera pose predictions between adjacent frames in Figure 4.9. Compared to the model that is only trained on images, the proposed method predicts more stable camera poses that change smoothly over time.

Keypoints Re-projection for Image-based Reconstruction We visualize re-projected keypoints on test images in Figure 4.10, where the corresponding quantitative results are presented in Section 4.5.3, Table 4.1. The proposed ACMR model is able to predict more accurate keypoints compared to the CMR [49] method, especially when the bird performs an asymmetric pose, e.g. first row in Figure 4.10.

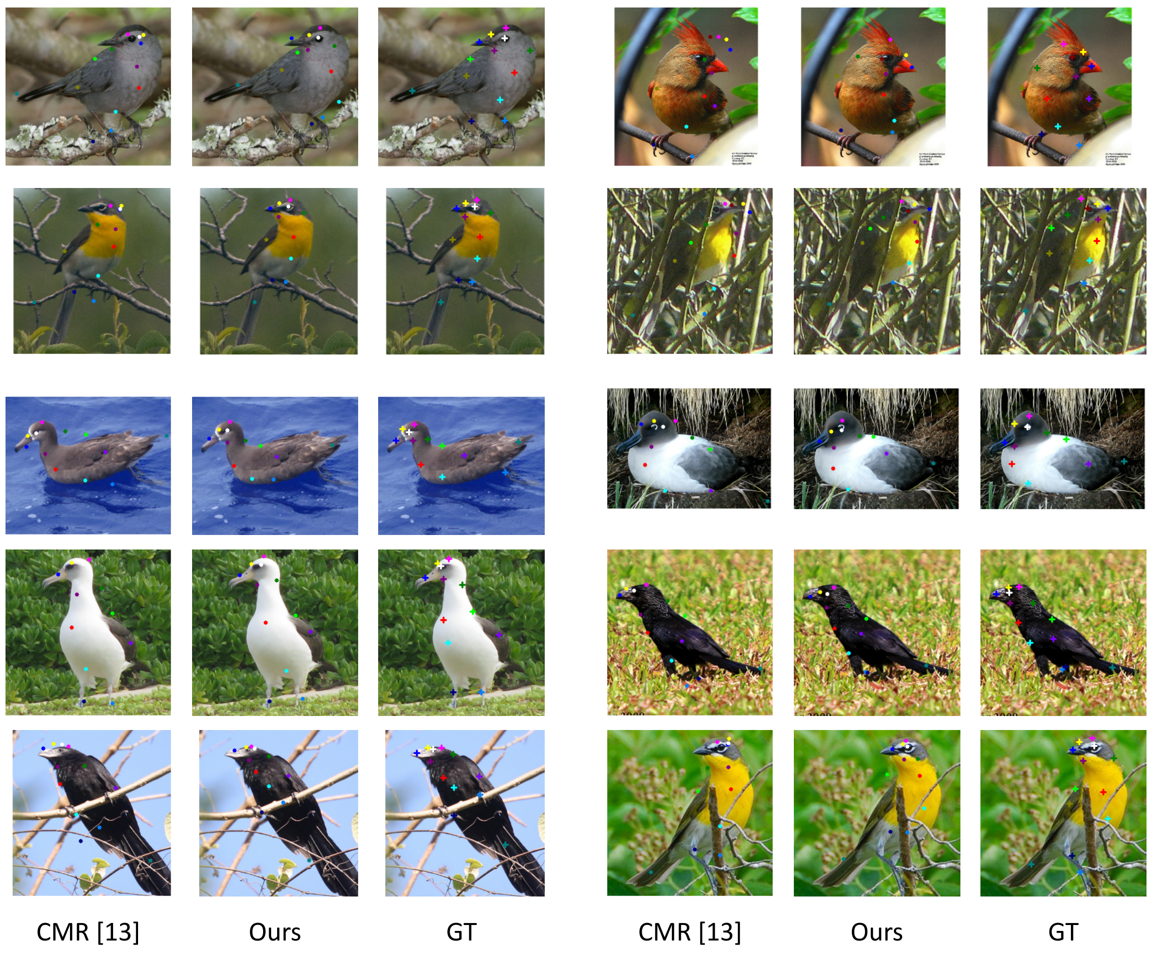
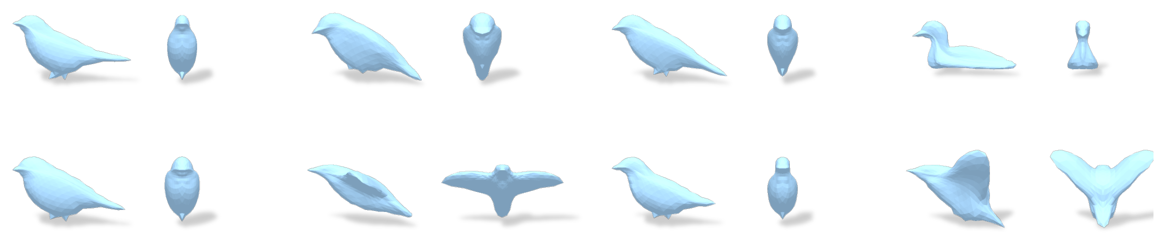


Figure 4.10: Visualization of re-projected keypoints of the single-view image reconstruction model.

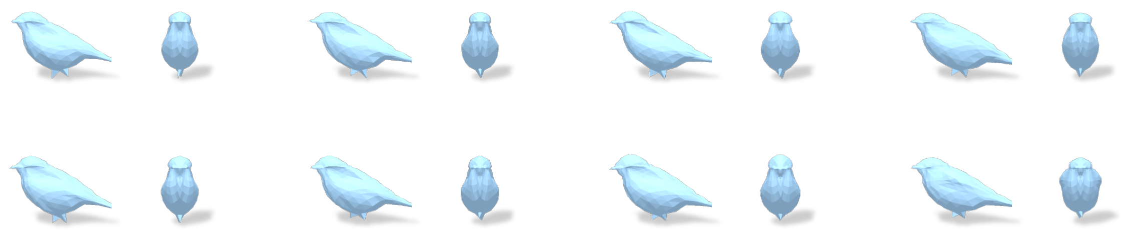
Bases Visualization We visualize the eight shape bases obtained by applying KMeans clustering to all reconstructed meshes by the CMR [49] method for birds in Figure 4.11(a). We also show the bases obtained by applying PCA to the bottleneck features of the image encoder. Note that the latter fails to discover rare shape modes (e.g., duck and flying bird) in the dataset as shown in Figure 4.11(b). Thus we choose to use KMeans to obtain shape bases.

4.5.3 Quantitative Results

Evaluations on the image dataset. As shown in Table 4.1(b) *vs.* (c), our ACMR model achieves comparable mask IoU and higher keypoints re-projection accuracy



(a) Applying KMeans to reconstructed meshes by CMR [13]



(b) Applying PCA to feature space of CMR [13]

Figure 4.11: Bases visualization.

Table 4.1: Quantitative evaluation of mask IoU and keypoint re-projection (PCK@0.1) on the CUB dataset [122].

(a) Metric	(b) CMR [49]	(c) ACMR	(d) ACMR, no ΔV	(e) ACMR, no ARAP	(f) ACMR-vid
Mask IoU ↑	0.706	0.708	0.647	0.758	**0.773**
PCK@0.1 ↑	0.810	0.855	0.790	0.857	**0.895**

compared to the baseline model [49] with the same network architecture. This confirms the correctness of both the reconstructed meshes as well as that of the predicted camera poses. In addition, our ACMR-vid model achieves even better performance as shown in Table 4.1(f). We note that our full ACMR model does not quantitatively outperform the model trained without the ARAP constraint, because the motion deformation ΔV freely over-fits to the mask and keypoint supervision without any regularization. However, the model without the ARAP constraint visually shows spikes and unnatural deformations as shown in Figure 4.5(f).

Evaluations on the video dataset. As shown in Table 4.2(b) and (c) *vs.* (e), by using the proposed online adaptation method discussed in Section 4.4.2, the model tuned on videos achieves higher $\mathcal{J}$ and $\mathcal{F}$ scores compared to the model trained only on images. This indicates that the test-time trained model successfully adapts to

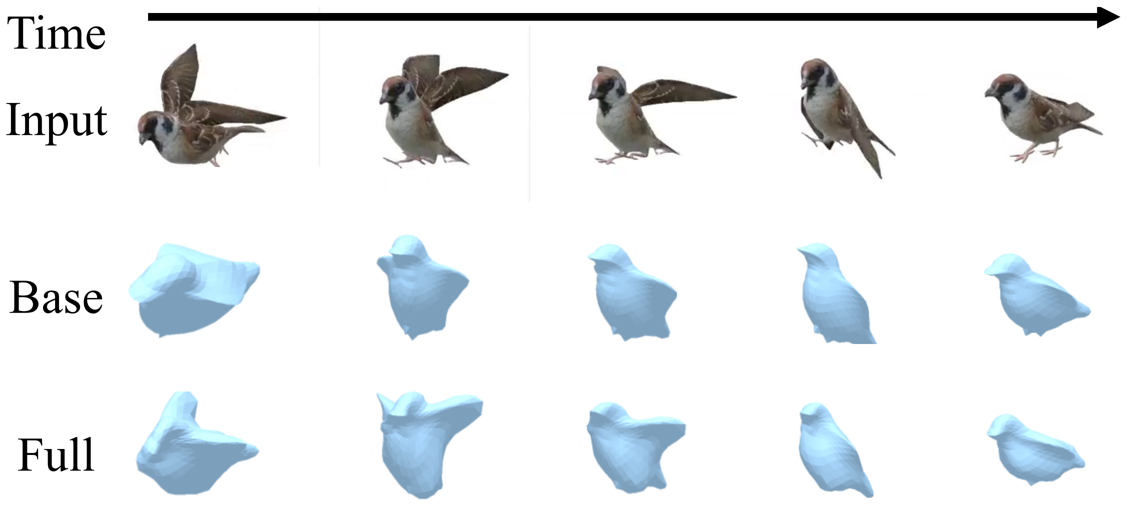


Figure 4.12: Reconstructions of an animated video clip.

Table 4.2: Quantitative evaluation of mask re-projection accuracy on the bird video dataset. "(T)" indicates the model is test-time trained on the given video., L_c, L_t, L_s are defined in Eq. 4.4, 4.5, 4.6 respectively.

(a) Metric	(b) CMR [49]	(c) ACMR	(d) ACMR (T)	(e) ACMR-vid (T), no L_c, L_t, L_s	(f) ACMR-vid (T)
$\mathcal{J}(Mean) \uparrow$	0.554	0.686	0.706	0.836	**0.868**
$\mathcal{F}(Mean) \uparrow$	0.209	0.363	0.406	0.666	**0.756**

unlabeled videos and can reconstruct meshes that conform well to the frames. The performance of the model is further improved by adding the correspondence, texture, and shape invariance constraints discussed in Section 4.4.2 during online adaptation, as shown in Table 4.2(f).

Evaluation on animated sequences. We apply the proposed model to an animated video clip and compare the predicted mesh with the ground truth mesh using Chamfer distance every 10 frames. We show the qualitative reconstructions in Figure 4.12 and quantitative evaluation results in Table 4.4. The proposed ACMR method outperforms the baseline CMR [49] model and is further improved via the proposed online adaptation strategy discussed in Section 4.4.2.

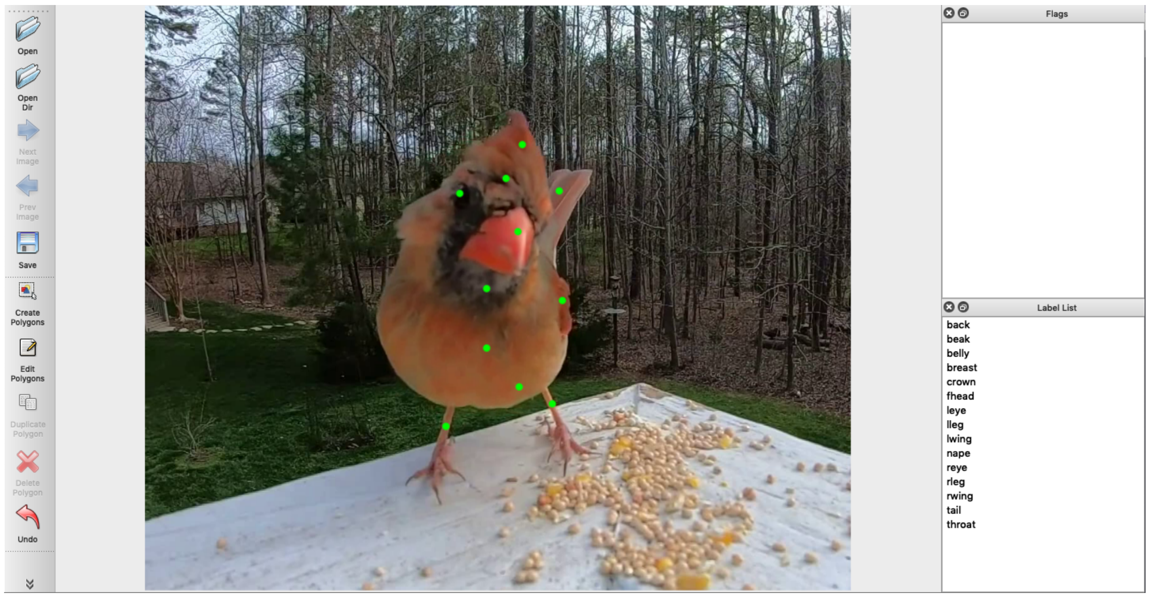


Figure 4.13: Keypoints annotation using the Labelme [121] toolkit.

Keypoint Re-projection Accuracy on Videos We evaluate the keypoint re-projection accuracy on the 22 videos we collected. To create the ground truth keypoints, we follow the protocol of the CUB dataset [122] to annotate 15 semantic keypoints every five frames in each video, via the Labelme [121] toolkit (see Figure 4.13 for the annotation interface.) Visualizations of the re-projected keypoints by different methods are visualized and compared in Figure 4.14.

Since we do not have the keypoint assignment matrix proposed in [49], we employ the canonical keypoint UV map to obtain the 3D keypoints (Section 4.4.1). The keypoint re-projection is done by (i) warping the canonical keypoint UV map to each individual predicted mesh surface; (ii) projecting the canonical keypoint back to the 2D space via the predicted camera pose; (iii) comparing against the ground truth keypoints in 2D. This evaluation implicitly reveals the correctness of both the predicted shape and camera pose for the mesh reconstruction algorithm, especially for objects that do not have 3D ground truth annotations.

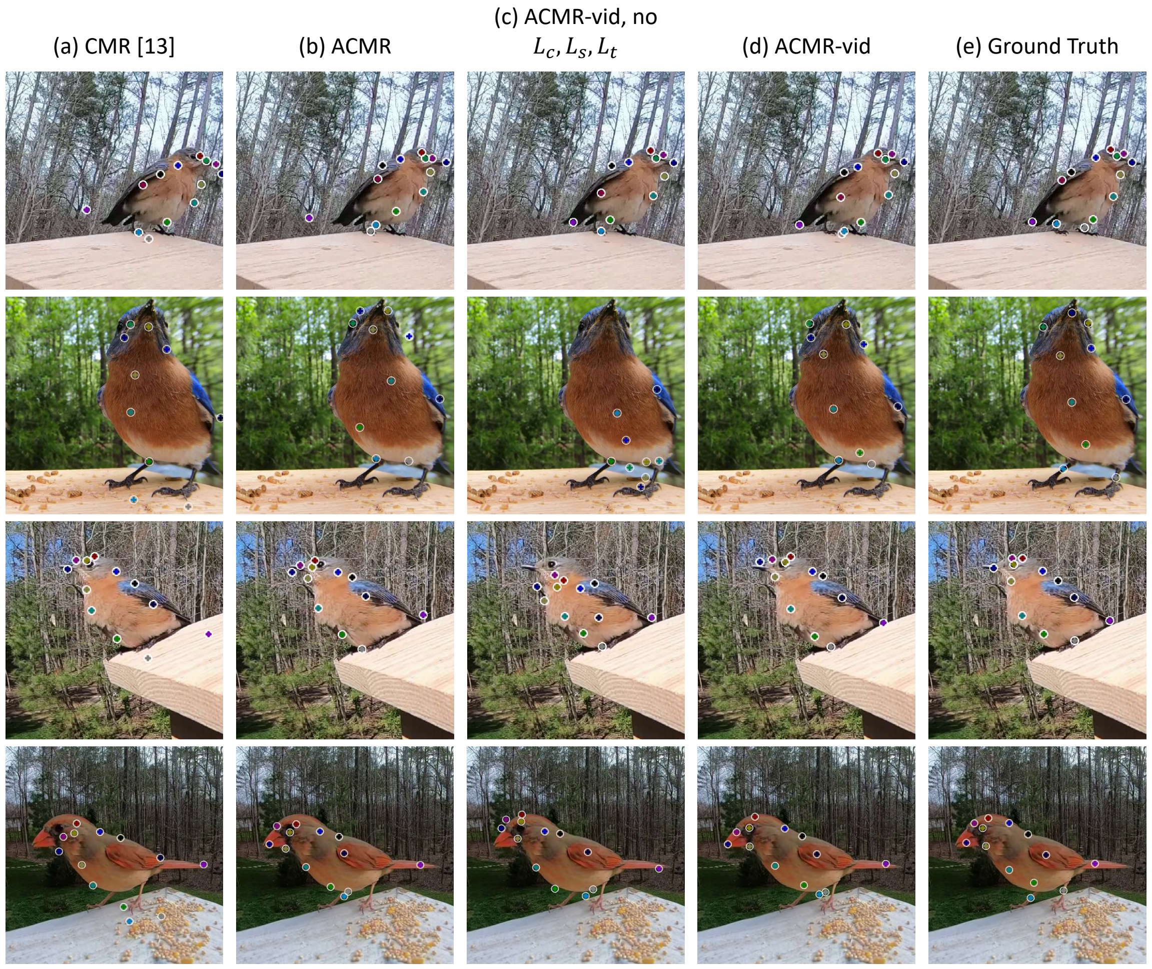


Figure 4.14: Visualization of re-projected keypoints on videos. We use white circles to highlight the keypoints for better visualization.

Compared to frame-wisely applying the CMR [49] (Table 4.3 (b)) or the ACMR model (Table 4.3 (c)) discussed in Section 4.4.1, the test-time tuned model achieves higher PCK score, as shown in Table 4.3 (f). It verifies the effectiveness of the proposed test-time training scheme and the invariance constraints. Essentially, as we noted in Section 4.4.1, although the original ACMR, i.e., using the ResNet-18 [36] as image encoder with batch normalization layers [45] in Table 4.3 (c), achieves relatively promising results, it is hard to adapt this model to new domains like low-quality videos (e.g., when switching from the *.eval()* mode to the *.train()* mode in PyTorch [91]). The performance drops significantly after test-time tuning as shown in Table 4.3 (d).

Table 4.3: Keypoint re-projection evaluation on videos. "(T)" indicates the model is test-time trained on the given video., L_c, L_t, L_s are defined in Eq. 4.4, Eq. 4.5 and Eq. 4.6, respectively.

(a) Metric	(b) CMR [49]	(c) ACMR	(d) ACMR (T)	(e) ACMR-vid (T), no L_c, L_t, L_s	(f) ACMR-vid (T)
PCK@0.1 ↑	0.514	0.751	0.424	0.644	**0.794**

Table 4.4: Evaluation on synthetic data.

Metric	CMR [49]	ACMR	ACMR (T)
Chamfer↓	0.016	0.015	**0.012**

Evaluations for self-supervised setting (Section 4.4.3). After online adaptation, this model too, achieves both a higher $\mathcal{J}$ score (0.843 *vs.* 0.582) and $\mathcal{F}$ score (0.678 *vs.* 0.256) compared to the model pre-trained on the image dataset, i.e., our method reconstructs promising dynamic 3D meshes without annotating any keypoint for training.

4.6 Ablation Studies

4.6.1 The Role of Shape Base

To demonstrate the superiority of using a set of shape bases versus using a single template, we train a baseline model where we replace the shape combination branch with the template obtained by the CMR approach [49]. This setting is equivalent to a using a single shape base (denoted as *single base*). We show quantitative and qualitative comparisons with the proposed ACMR model in Table 4.5 and Figure 4.15, respectively. As shown in Table 4.5(b) *vs.* (c), the model trained with a single base

Table 4.5: Quantitative comparison of the single base model with the proposed ACMR model.

(a) Metric	(b) Single base	(c) ACMR
Mask IoU ↑	0.605	0.708
PCK@0.1 ↑	0.655	0.855

Table 4.6: Ablation study on the ARAP constraint in online adaptation.

(a) Metric	(b) without ARAP	(c) ACMR-vid (T)
$\mathcal{J}(Mean)\uparrow$	0.875	0.868
$\mathcal{F}(Mean)\uparrow$	0.782	0.756
PCK@0.1↑	0.815	0.794

template struggles to fit the final shape when the instance is largely different from the given template (also shown in Figure 4.15). In contrast, the proposed ACMR model with 8 shape bases performs favorably against the single base model.

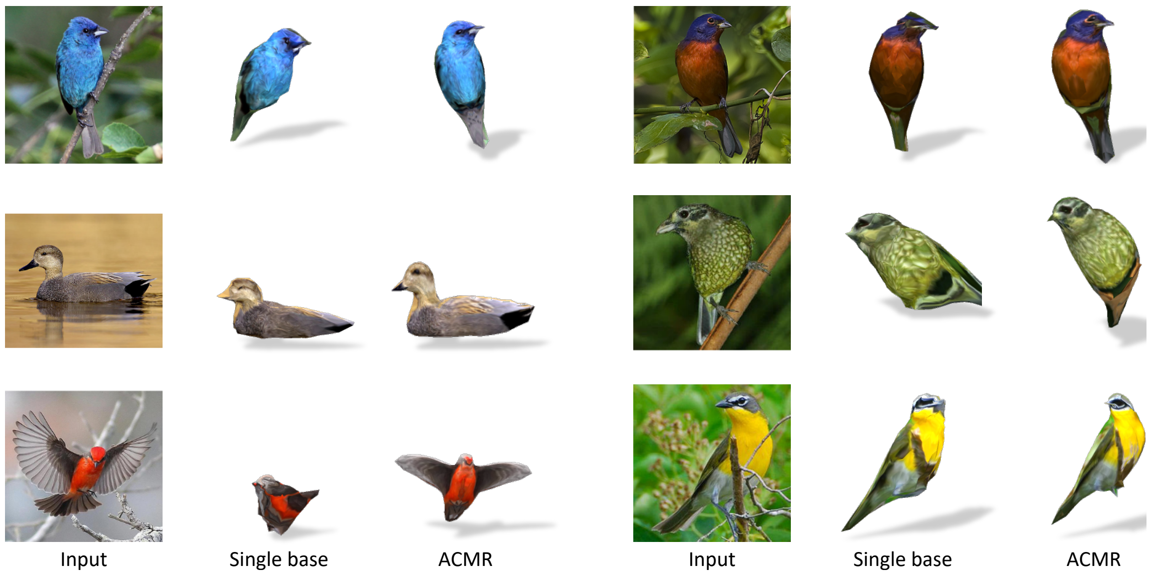


Figure 4.15: Qualitative comparison of the single base model with the proposed ACMR model with multiple shape bases. The single base model suffers when the instance is largely different from the template, e.g. flying bird or a duck.

4.6.2 ARAP Constraint in Online Adaptation

To verify the effectiveness of using the ARAP constraint in the online adaptation process, we test-time tune on the videos without this constraint. Although performing online adaptation without the ARAP constraint yields better quantitative evaluations as shown in Table 4.6, the reconstructed meshes are not plausible from unobserved views, as shown in Figure 4.7.

4.7 Failure Cases

Our work is the first to explore the challenging task of reconstructing 3D meshes of deformable object instances from videos in the wild. Impressive as the performance is, this challenging task is far from being fully solved. We discuss failure cases and limitations of the proposed method in the following. To begin with, we focus on genus-0 objects such as birds and zebras in this work. Thus our model suffers when it is generalized to objects with large concave holes such as chairs, humans etc. Second, our work struggles to reconstruct meshes from videos with large motion and lighting changes as well as occlusion, (see Figure 4.16 and the accompanying video). This is mainly due to the failure in correctly propagating parts by the self-supervised UVC model [69], which is out of scope of this work. We leave all these failure cases and limitations to future works.

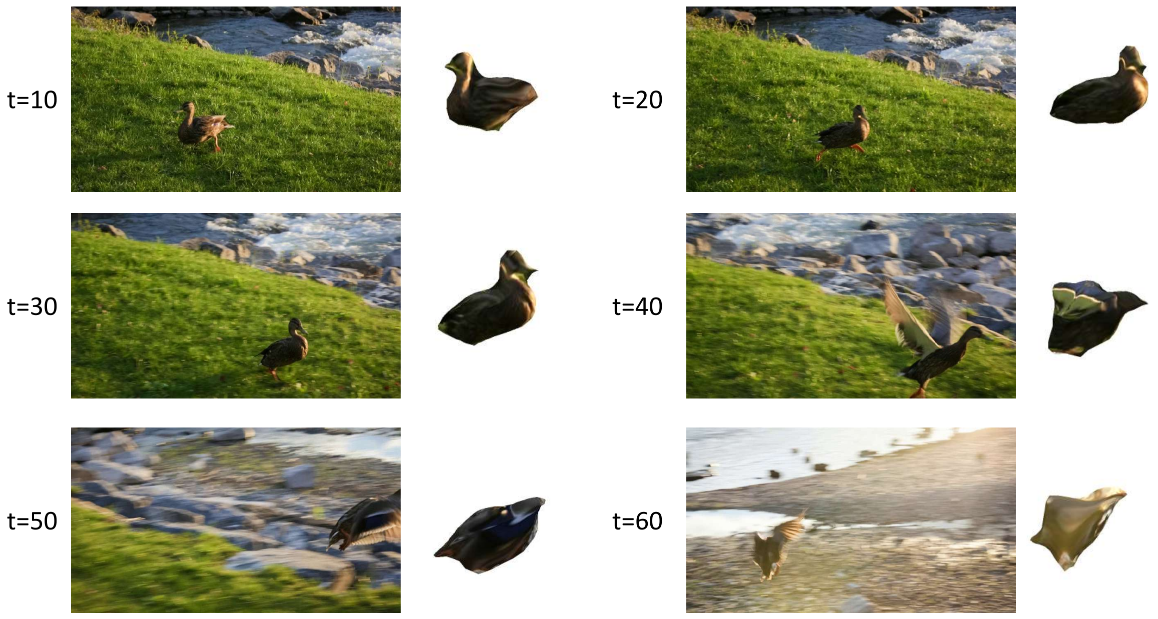


Figure 4.16: Failure cases. Our model fails when there is occlusion (e.g., $t = 50$, t stands for frame number) or large lighting changes (e.g., $t = 60$). Please refer to the accompanying video for more details of this test video.

4.8 Conclusions

We propose a method to reconstruct temporally consistent 3D meshes of deformable objects from videos captured in the wild. We learn a category-specific 3D mesh reconstruction model that jointly predicts the shape, texture, and camera pose from single-view images, which is capable of capturing asymmetric non-rigid motion deformation of objects. We then adapt this model to any unlabeled video by exploiting self-supervised signals in videos, including those of shape, texture, and part consistency. Experimental results demonstrate the superiority of the proposed method compared to state-of-the-art works, both qualitatively and quantitatively.

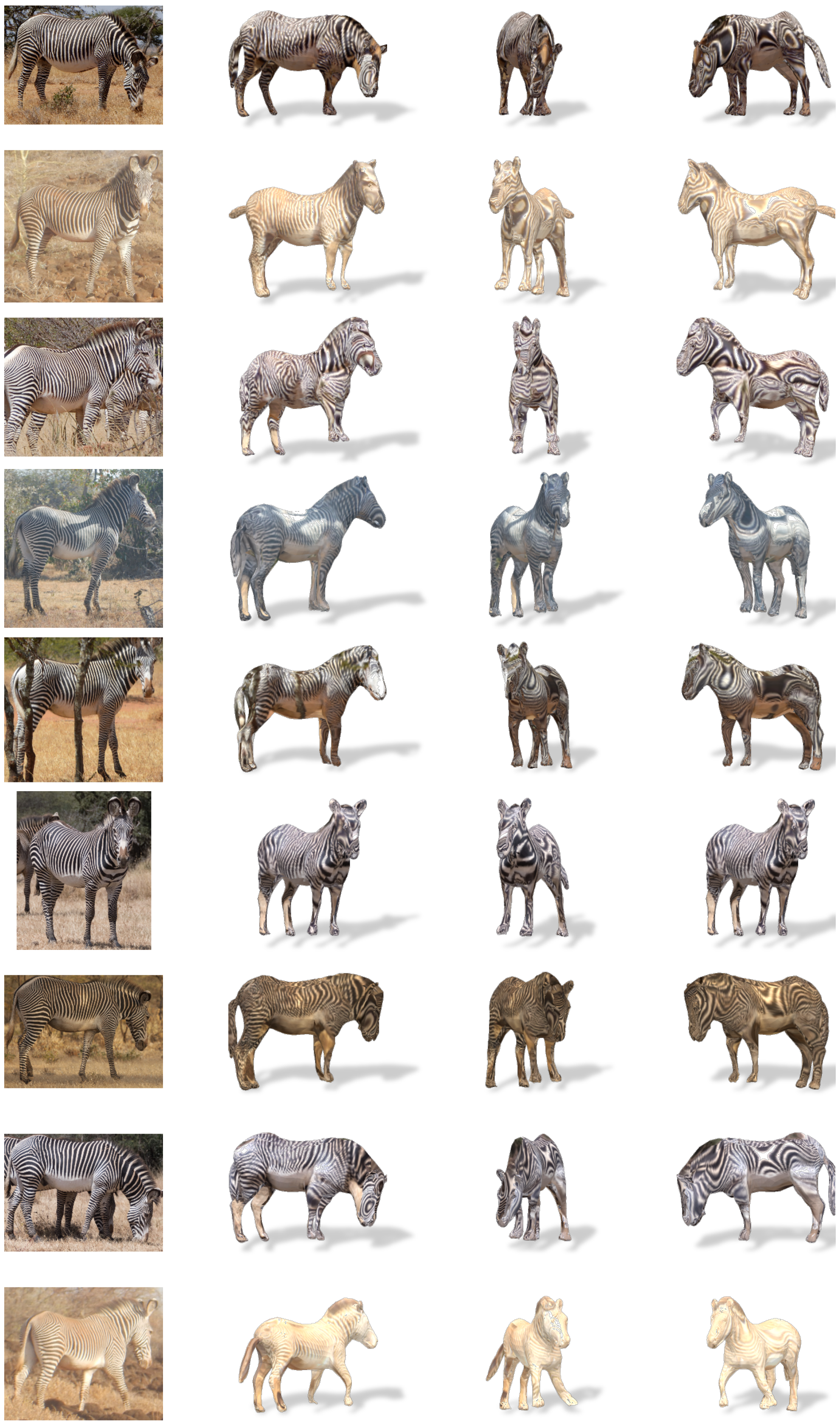


Figure 4.17: Visualization of reconstructed zebras.

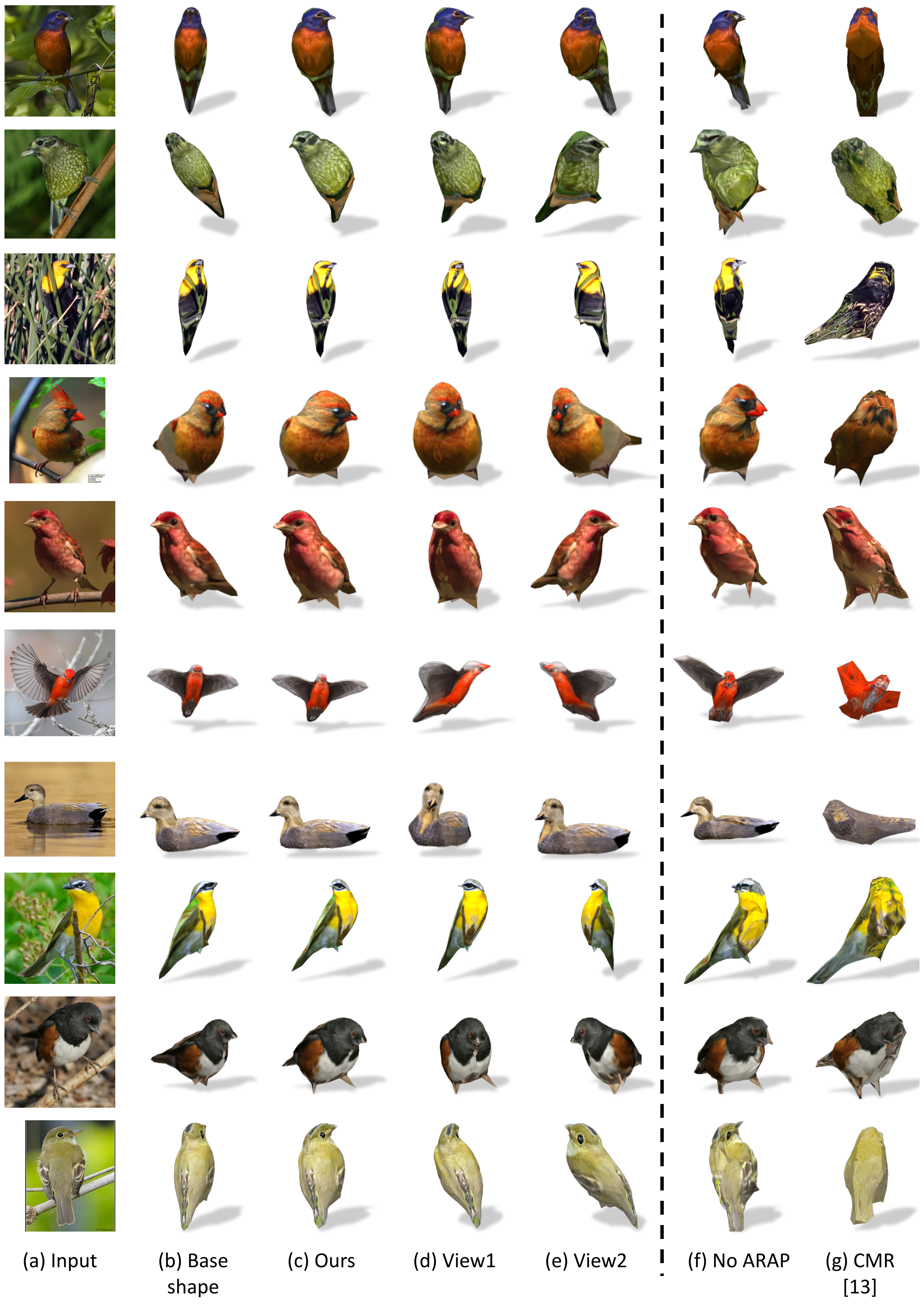


Figure 4.18: More qualitative reconstruction results on CUB birds [122].

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation, I first introduce a method that learns temporal correspondence from videos and then apply both semantic and temporal correspondence to facilitate 3D shape reconstruction for images and videos. Specifically, I demonstrate:

- A model that learns correspondences across video frames in a self-supervised manner by jointly tackling region-level and pixel-level correspondence learning and allowing them to facilitate each other through a shared inter-frame affinity matrix. Experimental results demonstrate the effectiveness of the proposed approach versus the state-of-the-art self-supervised video correspondence learning methods, as well as supervised models such as the ResNet-18 trained on ImageNet with classification labels.

- An algorithm that reconstructs 3D shape, texture and camera pose from single-view images, with only a category-specific collection of images and silhouettes as supervision. The self-supervised framework enforces semantic consistency between the reconstructed meshes and images and largely reduces ambiguities in the joint prediction of 3D shape and camera pose from 2D observations. It also creates a category-level template and a canonical semantic UV map, which capture the most representative shape characteristics and semantic parts of objects in each category, respectively. Experimental results demonstrate the efficacy of the proposed method in comparison to the state-of-the-art supervised category-specific reconstruction methods.

- A method that reconstructs temporally consistent 3D meshes of deformable objects from videos captured in the wild. I first learn a category-specific 3D mesh reconstruction model that jointly predicts the shape, texture, and camera pose from single-view images, which is capable of capturing asymmetric non-rigid motion deformation of objects. I then adapt this model to any unlabeled video by exploiting self-supervised signals in videos, including those of shape, texture, and part consistency. Experimental results demonstrate the superiority of the proposed method compared to state-of-the-art work, both qualitatively and quantitatively.

5.2 Future Work

The three work discussed in this dissertation open the door to unsupervised temporal correspondence learning and 3D mesh reconstruction from images and videos. They can be improved from several aspects:

- One promising direction is to learn temporal correspondence using transformers [20]. Vision transformer has spurred great interest due to its powerful capacity in modeling long-range spatial and temporal relationship in both images and videos. The UVC model discussed in Chapter 2 can only model the correspondences well between a limited number of frames, i.e., around 50 frames. Thus, it could potentially benefit from the powerful capacity of transformers to capture long-range correspondences in video frames. Such a design could also improve the work described in Chapter 4, where the sliding window schema can be removed if a model that captures long-range correspondence is available.

- For the work in Chapter 3 and Chapter 4, we represent the 3D shapes as meshes. Though well studied, the representation capacity of meshes is limited. For instance, they cannot precisely capture details in objects or model objects with a genus greater than zero. Recently, implicit functions [90, 85] have become popular due to their flexibility and high representation capacity. Thus, one future work is to employ implicit functions for unsupervised 3D shape reconstruction from videos and images. This could generalize the proposed models to more complex categories, e.g., chairs, bookshelves, etc.

Bibliography

[1] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008.

[2] Anurag* Arnab, Carl* Doersch, and Andrew Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *CVPR*, June 2019.

[3] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. *arXiv preprint arXiv:1606.09549*, 2016.

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

[5] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 2010.

[6] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[7] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[9] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[12] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, 2019.

[13] Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *CVPR*, 2021.

[14] Jongwon Choi, Hyung Jin Chang, Sangdoo Yun, Tobias Fischer, Yiannis Demiris, and Jin Young Choi. Attentional correlation filter network for adaptive visual tracking. In *CVPR*, 2017.

[15] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016.

[16] Edo Collins, Radhakrishna Achanta, and Sabine Susstrunk. Deep feature factorization for concept discovery. In *ECCV*, 2018.

[17] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[19] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. In *NeurIPS*, 2019.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[22] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017.

[23] Pedro Felzenszwalb, Ross Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2009.

[24] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *ECCV*, 2018.

[25] David F. Fouhey, Weicheng Kuo, Alexei A. Efros, and Jitendra Malik. From lifestyle vlogs to everyday interactions. In *CVPR*, 2018.

[26] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[27] Rohit Girdhar, David Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

[28] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *ICCV*, 2019.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[30] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, 2018.

[31] Pei Guo and Ryan Farrell. Aligned to the object, not to the image: A unified pose-aligned representation for fine-grained recognition. In *WACV*, 2019.

[32] JunYoung Gwak, Christopher B Choy, Manmohan Chandraker, Animesh Garg, and Silvio Savarese. Weakly supervised 3d reconstruction with adversarial constraint. In *3DV*, 2017.

[33] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *CVPR*, 2020.

[34] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *3DV*, 2017.

[35] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

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

[37] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *BMVC*, 2018.

[38] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *IJCV*, 2019.

[39] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 2014.

[40] Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. Improving landmark localization with semi-supervised learning. In *CVPR*, 2018.

[41] John F Hughes, Andries Van Dam, James D Foley, Morgan McGuire, Steven K Feiner, and David F Sklar. *Computer graphics: principles and practice*. Pearson Education, 2014.

[42] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019.

[43] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[44] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*, 2018.

[45] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[46] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015.

[47] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *ICCV*, 2013.

[48] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *TPAMI*, 2011.

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

[50] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *CVPR*, 2019.

[51] Hiroharu Kato and Tatsuya Harada. Learning view priors for single-view 3d reconstruction. In *CVPR*, 2019.

[52] Hiroharu Kato and Tatsuya Harada. Self-supervised learning of 3d objects from natural images. *arXiv preprint arXiv:1911.08850*, 2019.

[53] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[54] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[55] Anna Khoreva, Anna Rohrbach, and Bernt Schiele. Video object segmentation with language referring expressions. In *ACCV*, 2018.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICML*, 2015.

[57] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. In *ICCV*, 2019.

[58] Shu Kong and Charless Fowlkes. Multigrid predictive filter flow for unsupervised learning on videos. *arXiv preprint arXiv:1904.01693*, 2019.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

[60] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020.

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

[62] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[63] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. *arXiv preprint arXiv:1905.00875*, 2019.

[64] Junghyup Lee, Dohyung Kim, Jean Ponce, and Bumsub Ham. Sfnet: Learning object-aware semantic correspondence. In *CVPR*, 2019.

[65] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[66] Qizhu Li, Anurag Arnab, and Philip HS Torr. Holistic, instance-level human parsing. *arXiv preprint arXiv:1709.03612*, 2017.

[67] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. *NeurIPS*, 2020.

[68] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *ECCV*, 2020.

[69] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019.

[70] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV*, 2014.

[71] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *CVPR*, 2019.

[72] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[73] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *TPAMI*, 2011.

[74] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019.

[75] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *NeurIPS*, 2019.

[76] Shilong Liu, Lei Zhang, Xiao Yang, Hang Su, and Jun Zhu. Unsupervised part segmentation through disentangling appearance and shape. In *CVPR*, 2021.

[77] Sifei Liu, Guangyu Zhong, Shalini De Mello, Jinwei Gu, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Switchable temporal propagation network. In *ECCV*, 2018.

[78] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *SIGGRAPH*, 2015.

[79] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*. Vancouver, British Columbia, 1981.

[80] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *SIGGRAPH*, 2020.

[81] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *CVPR*, 2015.

[82] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, 2015.

[83] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[84] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

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

[86] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *ICCV*, 2019.

[87] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *TPAMI*, 1993.

[88] Daniil Osokin. Real-time 2D multi-person pose estimation on CPU: Lightweight openpose. *arXiv preprint arXiv:1811.12004*, 2018.

[89] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *ICCV*, 2019.

[90] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[91] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*. 2019.

[92] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *CVPR*, 2019.

[93] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37(6), November 2018.

[94] Bojan Pepik, Peter Gehler, Michael Stark, and Bernt Schiele. 3d 2 pm–3d deformable part models. In *ECCV*, 2012.

[95] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

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

[97] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[98] Junliang Xing Mengdan Zhang Weiming Hu Qiang Wang, Jin Gao. Dcfnet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*, 2017.

[99] Ke Gong Liang Lin Qixian Zhou, Xiaodan Liang. Adaptive temporal encoding network for video instance-level human parsing. In *Proc. of ACM International Conference on Multimedia (ACM MM)*, 2018.

[100] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *NeurIPS*, 2016.

[101] Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General automatic human shape and motion capture using volumetric contour cues. In *ECCV*, 2016.

[102] Kathleen Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, Scott Fleming, Tina Brill, David Hoeferlin, and Dennis Burnsides. Civilian american and european surface anthropom- etry resource (caesar) final report. *Tech. Rep. AFRL-HE- WP-TR-2002-0169, US Air Force Research Laboratory*, 2002.

[103] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *CVPR*, 2018.

[104] Michael Rubinstein, Ce Liu, and William T Freeman. Towards longer long-range motion trajectories. In *BMVC*, 2012.

[105] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[106] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, 2007.

[107] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[108] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization. *arXiv preprint arXiv:1909.13231*, 2019.

[109] Attila Szabó and Paolo Favaro. Unsupervised 3d shape learning from image collections in the wild. *arXiv preprint arXiv:1811.10519*, 2018.

[110] Ran Tao, Efstratios Gavves, and Arnold W M Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[111] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *ICCV*, 2017.

[112] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[113] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[114] Luan Tran and Xiaoming Liu. On learning 3d face morphable model from in-the-wild images. *TPAMI*, 2019.

[115] Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *CVPR*, 2018.

[116] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017.

[117] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *NeurIPS*, 2017.

[118] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[119] Timo von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018.

[120] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018.

[121] Kentaro Wada. labelme: Image Polygonal Annotation with Python. `https://github.com/wkentaro/labelme`, 2016.

[122] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

[123] Bastian Wandt, Hanno Ackermann, and Bodo Rosenhahn. 3d reconstruction of human motion from monocular image sequences. *TPAMI*, 2016.

[124] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NeurIPS*, 2013.

[125] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[126] Ning Wang, Yibing Song, Chao Ma, Wengang Zhou, Wei Liu, and Houqiang Li. Unsupervised deep tracking. In *CVPR*, 2019.

[127] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu. Dcfnet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*, 2017.

[128] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional siamese network for high performance online visual tracking. In *CVPR*, 2018.

[129] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. *CVPR*, 2019.

[130] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017.

[131] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019.

[132] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *ICCV*, 2019.

[133] Olivia Wiles and Andrew Zisserman. Silnet: Single-and multi-view reconstruction by learning from silhouettes. *arXiv preprint arXiv:1711.07888*, 2017.

[134] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016.

[135] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Photo-geometric autoencoding to learn 3d objects from unlabelled images. *arXiv preprint arXiv:1906.01568*, 2019.

[136] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020.

[137] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 2015.

[138] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[139] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *ECCV*, 2016.

[140] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014.

[141] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

[142] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NeurIPS*, 2016.

[143] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Efficient mean-shift tracking via a new similarity measure. In *CVPR*, 2005.

[144] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, 2018.

[145] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *TPAMI*, 2013.

[146] Chun-Han Yao, Wei-Chih Hung, Varun Jampani, and Ming-Hsuan Yang. Discovering 3d parts from image collections. In *ICCV*, 2021.

[147] Jason Y Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. Predicting 3d human dynamics from video. In *ICCV*, 2019.

[148] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[149] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018.

[150] Qixian Zhou, Xiaodan Liang, Ke Gong, and Liang Lin. Adaptive temporal encoding network for video instance-level human parsing. *arXiv preprint arXiv:1808.00661*, 2018.

[151] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *CVPR*, 2016.

[152] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua Tenenbaum, and William Freeman. Visual object networks: Image generation with disentangled 3D representations. In *NeurIPS*, 2018.

[153] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *ICCV*, 2017.

[154] Rui Zhu, Chaoyang Wang, Chen-Hsuan Lin, Ziyan Wang, and Simon Lucey. Object-centric photometric bundle adjustment with deep shape prior. In *WACV*, 2018.

[155] Yingying Zhu, Dong Huang, Fernando De La Torre, and Simon Lucey. Complex non-rigid motion 3d reconstruction by union of subspaces. In *CVPR*, 2014.

[156] Silvia Zuffi, Angjoo Kanazawa, Tanja Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images in the wild. In *ICCV*, 2019.

[157] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *ICCV*, 2019.

[158] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *CVPR*, 2017.