Visual Learning Beyond Direct Supervision

by

Tinghui Zhou

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

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexei A. Efros, Chair
Professor Jitendra Malik
Professor Bruno Olshausen

Summer 2018

Visual Learning Beyond Direct Supervision

Abstract

Visual Learning Beyond Direct Supervision

by

Tinghui Zhou

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

Deep learning has made great progress in solving many computer vision tasks for which labeled data is plentiful. But progress has been limited for tasks where labels are difficult or impossible to obtain. In this thesis, we propose alternative methods of supervised learning that do not require direct labels. Intuitively, although we do not know what the labels are, we might know various properties they should satisfy. The key idea is to formulate these properties as objectives for supervising the target task. We show that this kind of "meta-supervision" on *how* the output behaves, rather than *what* it is, turns out to be surprisingly effective in learning a variety of vision tasks.

The thesis is organized as follows. Part I proposes to use the concept of cycle-consistency as supervision for learning dense semantic correspondence. Part II proposes to use the task of view synthesis as supervision for learning different representations of scene geometry. Part III proposes to use adversarial supervision for learning gradual image transformations. Finally, we discuss the general concept of meta-supervision and how it can be applied to tasks beyond those presented in this thesis.

To my parents for their love and support

Contents

List of Figures

List of Tables

Acknowledgments

PhD is a long journey, so long that one cannot endure it alone. Fortunately, I was accompanied by an incredible group of people who helped me get through the finish line.

I would like to thank my advisor, Alexei (Alyosha) Efros, for his invaluable guidance, inspiration and support throughout the years. I first met him at CMU when I was a Masters student looking for a research advisor. Back then I was most fascinated by the mathematical complexity of machine learning problems, while Alyosha strived for the exact opposite – solving problems in the simplest possible way (with nearest neighbors being his favorite algorithm). I was very skeptical at first, but soon became a believer in simplicity as well. From him I not only received the best possible research guidance, but also learned to appreciate the much larger life beyond research. I am also thankful for his unequivocal support during the beginning of my PhD when my research was constantly hitting roadblocks. Despite mixed feelings about every submission being a last-minute rush and my snack box being looted on a daily basis, having Alyosha as my advisor was one of the best decisions of my life, and I will always be grateful for his mentorship.

When asked what it was like to be advised by Alyosha, Derek Hoiem had a great answer: "Alyosha is a poet. We [his students] must know how to translate his poetry into prose." However, this is not a trivial process. Fortunately, I had help from many of our talented postdocs and senior collaborators, including Yong Jae Lee, Stella Yu, Qixing Huang, Mathieu Aubry, Philipp Krähenbühl, and Phillip Isola. I would like to thank Yong Jae and Stella for guiding me through my first paper during PhD, and helping me transition from CMU to Berkeley. Thanks to Qixing and Mathieu for their passion and insight during our collaboration. I am thankful to Philipp Krähenbühl for all the insightful and candid conversations about research, programming and life in general. I learned a great deal from his unique perspectives. Thanks to Phillip Isola for showing me how to persist in pursuing grand research goals. One piece of advice about going on trips with Phil: be physically and mentally prepared for the unconventional level of roughness.

I am grateful to all my peers of the Efros group, including Jun-Yan Zhu, Shiry Ginosar, Richard Zhang, Carl Doersch, Taesung Park, Deepak Pathak, Allan Jabri, and Kate Rakelly. I don't think one could find a more friendly and supportive group anywhere else. Thanks to Jun-Yan (and his cat, Aquarius) for being my first and only roommate during my years at Berkeley. Jun-Yan seems to possess some special aura that makes living with him always eventful and full of surprises. I also learned a lot from witnessing his entire path of progression into the top-notch researcher as he is today. Special thanks to Shiry Ginosar for being the real "adult" in our group who never failed to provide the rational voice when group meetings and other events started derailing. I also had the fortune to work with her on multiple projects, and was constantly impressed by her unique insights and the ability to tease out the nonsense in seemingly complex research ideas. I thank Richard Zhang for all the fun discussion on sports, video games, American and Chinese culture, muscle gaining strategies and many others, which served a helpful reminder of life beyond the seemingly endless coding.

It persistently amazed me how much talent was packed into the small office space of the 7th floor of Sutardja Dai Hall (and later Cory 307 as well). I am thankful for being able to share the journey with the amazing peers and colleagues in BAIR, including Shubham Tulsiani, Pulkit Agrawal, Saurabh Gupta, Abhishek Kar, Bharath Hariharan, Georgia Gkioxari, Katerina Fragkiadaki, Andrew Owens, David Fouhey, Angjoo Kanazawa, Joao Carreira, Panna Felsen, Eric Kuo, Ke Li, Christian Häne, Amir Zamir, Ashish Kumar, Zhe Cao, Zhuang Liu, Shizhan Zhu, Yangqing Jia, Ross Girshick, Jeff Donahue, Evan Shelhamer, Jonathan Long, Eric Tzeng, Lisa Anne Hendricks, Judy Hoffman, Dinesh Jayaraman, Yang Gao, Samaneh Azadi, Ronghang Hu, Huazhe Xu, Dequan Wang, Fisher Yu, Yi Wu, Hyun Oh Song, Jacob Huh, Caroline Chan, Andrew Liu, Hemang Jangle, Angela Lin, Rocky Duan, and Peter Chen. I had the fortune to publish two papers with Shubham. It was such a pleasure to collaborate with him, and I would nominate him for the best co-author award if there was one. We spent a wonderful summer internship in New York, and discovered the joy of Broadway musicals together[1]. Thanks to Pulkit Agrawal for being a great friend with whom I can bounce all the crazy ideas. I wish him success in preaching Pulkitology as a professor, and hopefully some day as an entrepreneur too. Special thanks to David Fouhey and Angjoo Kanazawa for making our bay full of joy, energy and arguably the most interesting bay in Cory 307. I have also had the privilege of interacting with many faculty members at Berkeley, including my thesis committee members Jitendra Malik and Bruno Olshausen, as well as Ren Ng, Trevor Darrell, Sergey Levine, and Pieter Abbeel.

Thanks to many other Berkeley friends for sharing the journey with me, including Weilun Sun, Xuaner Zhang, Chang Lan, Biye Jiang, Ling-Qi Yan, Hezheng Yin, Chi Jin, Qifan Pu, Xiang Gao, Xin Wang, Yang You, Yuansi Chen, Yubei Chen and Xi Zhang. Special thanks to Angie Abbatecola for numerous career-saving tips on treading through Berkeley bureaucracies.

I also enjoyed my two-time internships at Google. Thanks to David Lowe and Matthew Brown for hosting me in the Seattle office, and giving me all the help with the internship project, especially during the tumultuous period when I was stuck in China due to visa issues. I spent an amazing summer at New York hosted by Noah Snavely. Noah's vast knowledge on both computer vision and graphics coupled with the nicest personality makes working with him both enlightening and fun. I am fortunate to have worked with him on several projects, and truly grateful for all the help and advice I received. Thanks to Richard Tucker for the life tips in New York and the tremendous help with the internship project. I would also like to thank all the Googlers and fellow interns for making both internships an incredible experience, including John Flynn, Graham Fyffe, Aseem Agarwala, Yiming Liu, Li Zhang, Eric Penner, Ziwei Liu, Raymond Yeh, Yipu Zhao, Yongbin Sun, Michael Figurnov, Jingbo Shang, Shanshan Wu, Shihui Li and many others.

Last but not least, this thesis would not have been possible without the love and support from my parents who have given me all the freedom to pursue my dream. I dedicate this thesis to them.

[1] Ok ok, it was mostly me nudging him for company...

Chapter 1

Introduction

Computer vision has made great progress in a variety of domains, including image classification [116, 180, 73], object detection [58, 164, 74], semantic segmentation [136, 20], human pose estimation [15, 65] and many others. Such progress is largely driven by the rapid development of supervised learning using deep neural networks. The number of trainable parameters in these networks could range from millions to billions, which require large amount of labeled examples for training. Datasets like ImageNet [171] and COCO [128] have been excellent data source so far for many recognition tasks. However, we should also notice that there exists a long list of vision tasks for which it is very difficult or even fundamentally infeasible to obtain labeled data in large scale – amodal scene completion, scene flow estimation, dense correspondence, intrinsic image decomposition, just to name a few. Therefore, a natural question arises: is it possible to overcome label scarcity while still leveraging the computational power of deep learning?

One plausible solution is computer simulation, where we use computer graphics to render a synthetic environment to have full control of the data generation process [14, 18, 37, 167]. While the quality and usability of these environments have been improving over the years, models trained on such data are still not directly applicable to the real world due to the significant domain gap between the rendered and the real world visual data. The other potential solution is transfer learning, where the network weights are initialized by training on some pretext task like ImageNet classification, and then later fine-tuned on the target task. This strategy is effective in reducing the number of training labels for the target task, but currently still requires a nontrivial amount of labeled data.

In this thesis, we investigate how to use alternative supervisory signals for learning visual tasks without requiring any direct labels. Intuitively, although we do not know what the ground-truth is, we might know how the various properties it should satisfy. The key idea is to formulate these properties as objectives for learning the target task. As demonstrated in this thesis, this kind of "meta-supervision" on *how* the output behaves, rather than *what* it is, turns out to be surprisingly effective in learning a variety of visual tasks.

Part I: Learning dense semantic correspondence We first study the concept of cycle-consistency and how it could be utilized to obtain globally-consistent semantic correspondence within image collections. Then we show how to use cycle-consistency as supervisory signal for learning pairwise dense correspondence. For this task, although it is infeasible to collect large-scale ground-truth in the real image domain, we know the ground-truth should be consistent across instances of the same category even for synthetic ones. We use consistency as supervision for training the deep network without access to ground-truth correspondences in the real domain.

Part II: Learning scene geometry Then we present a framework for learning scene geometry with view synthesis as supervision without ground-truth geometric labels. We formulate the learning objective around the observation that if the scene geometry is predicted correctly from the input image(s), it should consistently explain the nearby frames through the task of novel view synthesis. In particular, we show that one could learn monocular depth and camera motion estimation from unstructured video sequences, and layered scene representation from posed images.

Part III: Learning gradual image transformation Finally, we propose to use adversarial networks to provide supervisory signals for learning gradual image transformations. By utilizing adversarial training coupled with a differential loss (that provides the direction of transformation) and a content loss (that preserves desired input semantics), we are able to train the network to predict gradual transformations without directly labeled data.

Finally, we discuss the general concept of “meta-supervision”: supervision on not what the data is but how it should behave, and how it could be applied to a variety of domains beyond what is presented in this thesis.

Part I

Learning dense semantic correspondence

Chapter 2

Dense Semantic Correspondence Through Joint Alignment

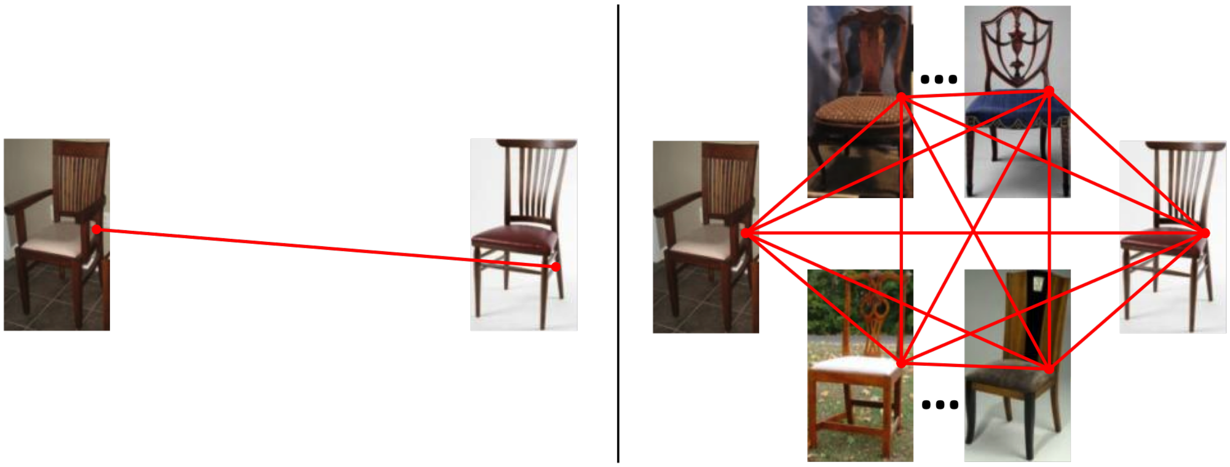
Correspondence (also known as alignment or registration) is the task of establishing connections between similar points/regions across different images, either sparsely (e.g. SIFT [139] keypoint matching), or densely at every pixel (e.g. optical flow). Correspondence can be defined either locally, as a pairwise connection between two images, or globally, as a joint label assignment across an image collection. In this chapter, we introduce FlowWeb, a correspondence-centric representation of image sets, and an algorithm for joint image alignment by maximizing the cycle consistency of the FlowWeb representation.

FlowWeb is a fully-connected correspondence flow graph with each node representing an image, and each edge representing the correspondence flow field between a pair of images, i.e. a vector field indicating how each pixel in one image can find a corresponding pixel in the other image. Correspondence flow is related to optical flow but allows for correspondences between visually dissimilar regions if there is evidence they correspond transitively on the graph. Our algorithm starts by initializing all edges of this complete graph with an off-the-shelf, pairwise flow method. We then iteratively update the graph to force it to be more self- consistent. Once the algorithm converges, dense, globally-consistent correspondences can be read off the graph. Our results suggest that FlowWeb improves alignment accuracy over previous pairwise as well as joint alignment methods [1]. The concept and effectiveness of cycle-consistency further inspired the work in the next chapter.

2.1 Introduction

Consider a pair of chairs depicted on Fig. 2.1(a). While the chairs might looks similar, locally their features (like the seat corner above) are very different in appearance, so classic image alignment approaches like SIFT Flow [129] have trouble finding correct correspon-

[1] This work was originally published as *FlowWeb: Joint Image Set Alignment by Weaving Consistent, Pixel-wise Correspondences.* In CVPR, 2015 [231].



(a) Example pixel correspondence using SIFT Flow

(b) Example pixel correspondence using our FlowWeb

Figure 2.1: Finding pixel-wise correspondences between images is difficult even if they depict similar objects: (a) a typical correspondence error using a state-of-the-art pairwise flow estimation algorithm. (b) We propose computing correspondences jointly across an image collection in a globally-consistent way.

dences. The reason we, human observers, have little trouble spotting visual correspondences between the features of these two chairs is likely because we have been exposed to many hundreds of chairs already, and are able to draw upon this knowledge to bridge the appearance gap. In light of this observation, we propose to "level the playing field" by starting with a *set of images* and computing correspondences *jointly* over this set in a globally-consistent way, as shown in Fig. 2.1(b).

One can appreciate the power of joint correspondence by considering faces, a domain where correspondences are readily available, either via human annotation, or via domain-specific detectors. Large-scale face datasets, meticulously annotated with globally-consistent keypoint labels ("right mouth corner", "left ear lower tip", etc) were the catalyst for a plethora of methods in vision and graphics for the representation, analysis, 3D modeling, synthesis, morphing, browsing, etc. of human faces [28, 145, 105, 12, 106]. Of course, faces are a special object class in many ways: they can generally be represented by a linear subspace, are relatively easy to detect in the wild and relatively easy to annotate (i.e. have well-defined keypoints). Nonetheless, we believe that some of the same benefits of having large, jointly registered image collections should generalize beyond faces and apply more broadly to a range of visual entities, provided we have access to reliable correspondences. Indeed, the recent work of Vicente et al. [196] on reconstructing PASCAL VOC classes using hand-annotated key-points is an exciting step in this direction. But what about cases when manual keypoint annotation is difficult or infeasible?

Our goal is to establish *globally-consistent* pixel-wise correspondences between all images within a given image collection, without any supervision. Just as the face modeling approaches start with a collection of detected faces in very coarse correspondence (on the level of a bounding box), we propose to start with a collection of coarsely similar images, which could be obtained as a result of an object detector [58], a mid-level discriminative visual element detector [35], or directly from a dataset with labeled bounding boxes.

The key insight is to focus on the correspondence flow fields between the images instead of working with image pixels directly. We achieve this by representing the image collection as a *FlowWeb* – a fully-connected graph with images as nodes and pixel flow fields between pairs of images as edges. We show that, starting with a simple initialization, we are able to force the FlowWeb to become consistent by iteratively updating the flow fields until convergence.

2.2 Background

Pairwise Image Flow The idea of generalizing optical flow to pairs of images that are only semantically related was first proposed in SIFT Flow [129], which adopted the computational framework of optical flow, but with local appearance matching being done on SIFT descriptors instead of raw pixels to add local appearance invariance. Deformable Spatial Pyramid (DSP) Matching [110], a recent follow-up to SIFT Flow, greatly improves the speed of the algorithm, also modestly improving the accuracy. Other works in this space include [10], which generalizes PatchMatch [8] to use feature descriptors instead of pixel patches, and more recently, finding pairwise correspondences using convolutional features [135].

Image graphs for pattern discovery The vast literature on object discovery and co-segmentation treats the image set as an unordered bag. Recent work exploits the connectivity within an image collection by defining a graph over images (e.g. [76, 138, 223, 68]) or objects (e.g. [142, 106, 25]). More relevant to us, [124, 43, 170] perform joint object discovery and segmentation on a noisy image set, resulting in often excellent region-wise correspondences. However, their main aim is to find and segment a consistent object, whereas we aim to find dense pixel-wise correspondences in an image set.

Graph consistency The idea of utilizing consistency constraints within a global graph structure has been applied to a variety of vision and graphics problems, including co-segmentation [199, 200], structure from motion [223, 208], and shape matching [85]. Most related to ours is [85], which formulates the constraint of cycle consistency as positive semi-definiteness of a matrix encoding a collection of pairwise correspondence maps on shapes, and solves for consistent maps via low-rank matrix recovery with convex relaxation. We also employ a cycle consistency constraint, but optimize it completely differently. Our problem complexity is also considerably larger: the number of pixels per image is typically two orders of magnitude greater than the number of sample points per shape.

Joint pixel-wise alignment of image sets Average images have long been used informally to visualize joint (mis)alignment of image sets (e.g. [190]). However, it was the seminal work of Congealing [122, 84] that established unsupervised joint alignment as a serious research problem. Congealing uses sequential optimization to gradually lower the entropy of the intensity distribution of the entire image set by continuously warping each image via a parametric transformation (e.g. affine). Congealing demonstrates impressive results on the digit dataset and some others, but does not perform as well on more difficult data.

RASL [157] also focuses on modeling a common image intensity structure of the image set; in their case, as a low-rank linear subspace plus sparse distractors specific to each image. Again, parametric transformations are used to align the images to the common subspace. The main difficulty with subspace methods is that they assume that the majority of images are already in good correspondence, else the subspace would end up encoding multiple shifted copies of the data. Collection Flow [104] also uses a low-rank subspace to model the common appearance of the collection, but with a clever twist by using non-parametric transformations (i.e. optical flow) that align between each image and its low-rank projection at each iteration (their application domain is faces, where the coarse alignment is good enough for subspace projections to work well). Mobahi et al. [149] propose a generative image representation that models each image as the composition of three functions: color, appearance, and shape. The appearance and shape functions are assumed to be constructed from a small set of basis functions (i.e., restricted to low-dimensional subspaces) in order to control the composition capacity. The model is used to establish dense correspondences between instances of the same object category.

All the subspace-based methods above share the same basic idea – compute some global representation of the image data, and then try to warp every image to make it more consistent with that representation (one can think of this as a star graph centered at the global representation connecting each image in the set). This works well if the distances between the images and the global representation can be trusted. But what if the image data lives on an articulation manifold [147], where only local distances are reliable? [172] takes this view, modeling the image collection not by some global representation, but using a locally-connected graph. This method shows very good results for aligning images of the same physical scene under low-dimensional transformations (global rotation, stretching, etc). However, it is not directly applicable for collections of multiple instances of the same object category. Concurrently with our work, Carreira et al. [16] models the image collection with a 'virtual view network', and resolves the difficulty of cross-view image alignment by finding the shortest geodesic path along the network. However, constructing the network requires either human annotations (e.g. keypoints) or pre-trained, category-specific pose predictors, whereas our method is fully unsupervised and does not require any training.

Like Collection Flow [104], our method uses compositions of flow fields to model connections between images. But instead of using a global, centered representation of the data like [209, 104, 149], our representation is defined on pairwise connections in the graph, like [172]. However, we differ from [172] in a number of important ways: 1) [172] represents the image set by a nearest neighbor graph, trusting the optical flow algorithm to be reliable when the flow field magnitude is small. We take a different perspective, and rely instead on the "wisdom of crowd", trusting the flow consistency among triplets of images in a fully connected image network. With the complementary information among images, not only can we ”fill in the blanks” arising from occlusion and outliers, but also find reliable correspondences between images that do not look alike; 2) [172] explicitly projects the manifold into a lower-dimensional space (3-4D), whereas we keep our correspondence flow graph in high dimension and let it become more self-consistent on its own, controlling its own intrinsic dimensionality.

2.3 Joint Image Alignment using FlowWeb

Given a collection of images $\{I_1, \ldots, I_N\}$ of the same visual concept, we would like to find dense pixel-wise correspondences that are consistent throughout the entire image collection. Our basic idea is that global correspondences emerge from consistent local correspondences in a bootstrap fashion. The quality of pixel-wise matching between two images I_i, I_j can be validated with multiple additional images. For each third image I_k, pixels $p \in I_i$ and $q \in$ image I_j are matched transitively if there is $r \in I_k$, where (p, I_i) matches (r, I_k), and (r, I_k) matches (q, I_j). That is, even when p, q do not have sufficient feature similarity directly, there may be sufficient indirect evidence from their similarity to other images supporting their match.

FlowWeb Representation Given a collection of N images, we build a complete graph of N nodes, where a node denotes an image, and the edge between two nodes (i, j) is associated with flow field T_{ij} between images (I_i, I_j) (see Fig. 2.2). For M pixels per image, T_{ij} is an $M \times 2$ matrix, each row containing the displacement vector between two matching pixels p and q in images I_i and I_j respectively:

$$T_{ij}^{pq} = x_q - x_p, \quad (p, I_i) \text{ matches } (q, I_j)\ , \tag{2.1}$$

where x_p denotes the spatial coordinates of pixel p.

2.3.1 Cycle consistency

Global correspondences in the image collection require the pairwise flow fields to be consistent among different paths connecting two nodes in the graph. Cycle consistency criterion can be expressed as the net displacement along a cycle in the FlowWeb being zero, e.g. for two-image cycle,

$$\begin{aligned} T_{ij}^{pq} + T_{ji}^{qr} &= (x_q - x_p) + (x_r - x_q) \\ &= x_r - x_p = 0, \text{iff } r = p. \end{aligned}$$

Let $T_{ik} \circ T_{kj}$ denote such flow composition from I_i through I_k to I_j. We define:

$$\begin{aligned} &\text{2-cycle consistency:} && T_{ij} \circ T_{ji} = 0 \\ &\text{3-cycle consistency:} && T_{ik} \circ T_{kj} \circ T_{ji} = 0. \end{aligned}$$

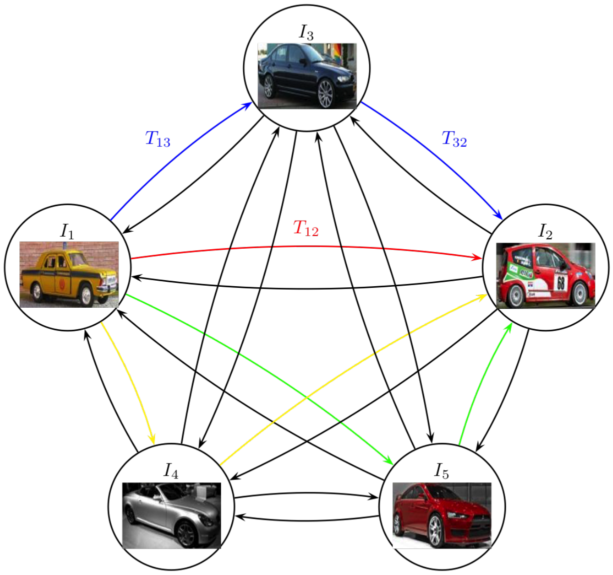


Figure 2.2: An example of our FlowWeb representation, where a node denotes an image, and each edge represents the flow field between two images.

While the number of cycles with arbitrary length is exponential in the number of nodes in the graph, [153] shows that considering only 2-cycles and 3-cycles are often sufficient for complete graphs. The concept of cycle consistency has also been explored in joint shape matching [85, 153], co-segmentation [199, 200] as well as SfM [223, 208].

We measure the quality of a matching flow by counting how many consistent 3-cycles go through it in the FlowWeb. If three images form a consistent cycle at a flow T_{ij}^{pq}, it means this flow is validated by a third image I_k, such that

$$T_{ij}^{pq} = T_{ik}^{pr} + T_{kj}^{rq}. \tag{2.2}$$

Let $\triangle_{ij}^{pq}$ denote the set of image nodes that complete a consistent cycle with flow T_{ij}^{pq}. We define the *single flow cycle consistency* (SFCC) score as the cardinality of $\triangle$:

$$\mathcal{C}(T_{ij}^{pq}) = |\triangle_{ij}^{pq}|_{card} = \sum_{k=1, k\notin\{i,j\}}^{N} [T_{ij}^{pq} = T_{ik}^{pr} + T_{kj}^{rq}], \tag{2.3}$$

where $[\cdot]$ is the binary indicator function.

We generalize the SFCC concept to the whole flow set $\mathbf{T} = \{T_{ij}\}$, and define *all flow cycle consistency* (AFCC) that counts the number of consistent 3-cycles in $\mathbf{T}$:

$$\mathcal{C}(\mathbf{T}) = \frac{1}{3} \sum_{i,j=1, i\neq j}^{N} \sum_{p\in I_i} \mathcal{C}(T_{ij}^{pq}) \ . \tag{2.4}$$

The factor of 1/3 corrects for the over-counting when summing over SFCC's for the three edges of the same cycle.

2.3.2 Objective

Our objective has two terms: FlowWeb cycle consistency $\mathcal{C}(\cdot)$, and regularization $\mathcal{R}(\cdot)$ that measures the difference between the current $\mathbf{T} = \{T_{ij}\}$ and the initial flow set $\mathbf{T}_0 = \{S_{ij}\}$ provided by a pairwise flow method (e.g. [110, 129]):

$$\max_{\mathbf{T}} \ \mathcal{C}(\mathbf{T}) - \lambda \mathcal{R}(\mathbf{T}, \mathbf{T}_0) \ , \tag{2.5}$$

$$\mathcal{R}(\mathbf{T}, \mathbf{T}_0) = \sum_{i,j=1, i \neq j}^{N} \sum_{p \in I_i} \| T_{ij}^{pq} - S_{ij}^{ps} \| \ , \tag{2.6}$$

where $\lambda > 0$ can be chosen based on the initialization quality, s denotes p's initial correspondence in image j, and $\| \cdot \|$ is the Euclidean norm.

2.3.3 Optimization

For clarity of exposition, we ignore the regularization term $\mathcal{R}(\cdot)$ for now and focus on optimizing the cycle consistency term alone. Our iterative optimization procedure builds on the following intuition: even when pixels p and q do not have sufficient feature similarity to be matched directly, they should still be matched if there is *sufficient indirect evidence* from 1) their similarity to other images supporting the match (inter-image) and/or 2) proximity to neighboring pixels that have a good match (intra-image). Both are provided by the cycle consistency measure, and exploited alternately at each iteration.

Inter-image phase The first phase of our iterative optimization involves the computation of a *priority* score for each flow in the current flow set. The update priority is high for flows that satisfy two criteria: 1) have low cycle consistency and 2) the consistency of an alternative solution is high. In our case, the alternative solutions to T_{ij}^{pq} are provided by one-hop transitive flows, i.e. $\{T_{ik}^{pr} + T_{kj}^{rt}, \forall k\}$[2]. Essentially, we would like the priority to measure the overall consistency gap between the current solution and some transitive solution. However, exact evaluation of the consistency gap is too expensive, as the change of one flow could potentially affect the consistency of all other flows that involve it in the SFCC computation.

Instead, we compute a lower bound based on the following observation: *if pixels $< p, r, t >$ are cycle-consistent, and there exists another pixel u such that both $< p, u, r >$ and $< r, u, t >$ are cycle-consistent, then $< p, u, t >$ are also cycle-consistent.* In other words, if we consider the two flows T^{pr} and T^{rt} that comprise a transitive flow between p and t, and denote the set of nodes each is consistent with by $\triangle^{pr}$ and $\triangle^{rt}$ respectively, then the transitive flow $T^{pt} = T^{pr} + T^{rt}$ is guaranteed to be consistent with $\triangle^{pr} \cap \triangle^{rt}$, and $|\triangle^{pr} \cap \triangle^{rt}|_{card}$ is the SFCC lower bound for T^{pt}, while holding all other flows fixed. In light of this observation,

[2] Note that we use q to denote p's direct correspondence in image j, and t to denote the transitive correspondence.

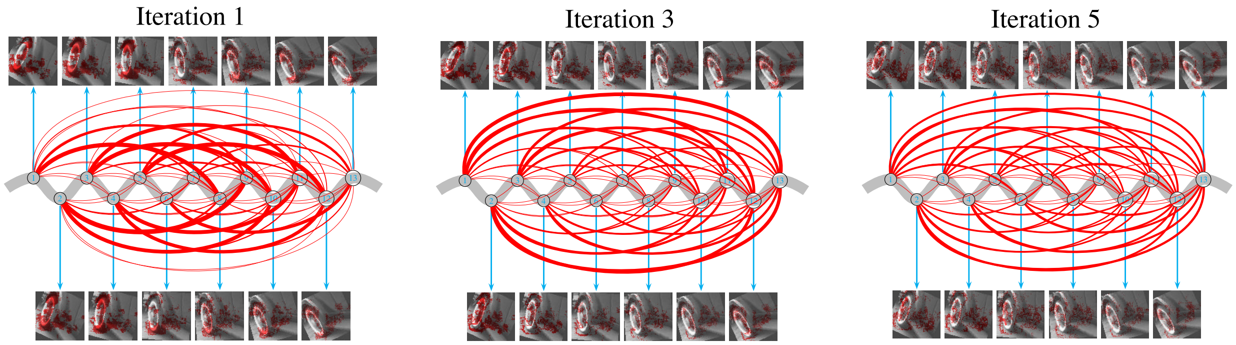


Figure 2.3: The flow update priority pattern over iterations. Shown here is an image ensemble made of 13 wheel images related by in-plane rotation, i.e. they lie on a 1D manifold (light gray curve) with increasing differences from left to right. The priority score is defined for each flow and it is large if there exists a transitive alternative that achieves better cycle consistency. Each image is shown with a red mask, indicating the sum of priority for all the flows associated with each pixel. The connections between each pair of images show the overall priority summed over all flows between them (thicker means higher). As shown, there are more mid-range connections (high update priority between not so similar images) initially, more long-range connections (high update priority between more distinct images) subsequently, and more even connections throughout the ensemble finally. There are far fewer short-range connections throughout iterations, since nearby images tend to have good correspondences and are cycle-consistent already. These flows thus have low priority.

for each pair of images i and j, we compute the update priority of a flow T_{ij}^{pq} by

$$\mathcal{P}(i,j,p) = \max_k |\triangle_{ik}^{pr} \cap \triangle_{kj}^{rt}|_{card} - |\triangle_{ij}^{pq}|_{card} \ , \tag{2.7}$$

where the first term of the RHS computes the consistency lower bound for each transitive flow and takes the maximum. Intuitively, $\mathcal{P}(i,j,p)$ is the lower bound of cycle consistency improvement if T_{ij}^{pq} is replaced by the transitive flow through image $\hat{k}$, where $\hat{k} = \arg\max_k |\triangle_{ik}^{pr} \cap \triangle_{kj}^{rt}|_{card}$. See Figure 2.3 for an illustration of the update priority pattern on a set of synthetic examples.

Intra-image phase While the previous phase essentially identifies and updates inconsistent flows to consistent ones through propagation, it is nonetheless unable to deal with cases in which the correct correspondence does not exist in the initial flow set, or simply has low cycle consistency because most of its transitive counterparts are noisy. Consider a set of front-view car images. The hood is typically texture-less while occupying a large image area, and pairwise matching based on low-level features such as SIFT would be highly noisy. As a result, it is likely that all flows emanating from such regions are incorrect and not consistent for propagation with the priority-based update.

The second phase of our optimization addresses this issue by exploiting *consistency-weighted spatial smoothing*, which identifies highly-consistent flows within a pairwise flow field, and utilizes them as *soft* anchor points to guide inconsistent flows to likely better solutions. For the example of front-view cars, one could potentially use flows from headlights or window corners that tend to be more cycle-consistent to guide flows from the hood. Specifically, for each flow field corresponding to a pair of images, we first identify flows that are of relatively low cycle consistency, and then apply a consistency-weighted Gaussian filter to each of them by

$$T_{ij}^{pq} = \frac{1}{Z} \sum_{p' \in I_i} T_{ij}^{p'q'} g_{\sigma_s}(\|x_{p'} - x_p\|) h_{\sigma_c}(\mathcal{C}(T_{ij}^{p'q'}) - \mathcal{C}(T_{ij}^{pq})) \tag{2.8}$$

where

$$Z = \sum_{p' \in I_i} g_{\sigma_s}(\|x_{p'} - x_p\|) h_{\sigma_c}(\mathcal{C}(T_{ij}^{p'q'}) - \mathcal{C}(T_{ij}^{pq})) \ . \tag{2.9}$$

$g_{\sigma_s}(\cdot)$ is a zero-mean Gaussian with σ_s controlling the spatial extent of the filter, and

$$h_{\sigma_c}(x) = \begin{cases} \exp(x/\sigma_c) & \text{if } x \geq 0 \\ 0 & \text{Otherwise} \end{cases} \tag{2.10}$$

determines how much an adjacent flow is weighted according to the gap in cycle consistency. Having $g(\cdot)$ and $h(\cdot)$ together ensures that each filtered flow is only influenced by flows that are both spatially near *and* more cycle-consistent.

Our iterative update pipeline is summarized below:

1. Compute the $SFCC$ score for each T_{ij}^{pq} using Eq. 2.3.
2. For each T_{ij}^{pq}, compute its update priority by Eq. 2.7, and record the node $\hat{k}$ that achieves the maximum.
3. Sort flows according to $\mathcal{P}(i, j, p)$, and update top $\beta\%$ flows by their transitive alternatives through image $\hat{k}$.
4. For each image pair i and j, apply Eq. 2.8 for consistency-weighted filtering.
5. Iterate 1–4 until the improvement of $\mathcal{C}(\mathbf{T})$ is below some threshold.

Regularization Optimizing the regularization term $\mathcal{R}(\cdot)$ can be easily incorporated into both update phases above. For the inter-image phase, the update priority becomes

$$\begin{aligned} \mathcal{P}(i, j, p) = \max_k |\triangle_{ik}^{pr} \cap \triangle_{kj}^{rt}|_{card} - |\triangle_{ij}^{pq}|_{card} - \\ \lambda(\|T_{ik}^{pr} + T_{kj}^{rt} - S_{ij}^{ps}\| - \|T_{ij}^{pq} - S_{ij}^{ps}\|). \end{aligned} \tag{2.11}$$

Similarly for the intra-image phase, we replace $h_{\sigma_c}(\mathcal{C}(T_{ij}^{p'q'}) - \mathcal{C}(T_{ij}^{pq}))$ with $h_{\sigma_c}(\mathcal{C}(T_{ij}^{p'q'}) - \mathcal{C}(T_{ij}^{pq}) - \lambda(\|T_{ij}^{p'q'} - S_{ij}^{ps}\| - \|T_{ij}^{pq} - S_{ij}^{ps}\|))$.

Implementation details: For better robustness to noisy initial pairwise matching, we use a relaxed threshold for determining cycle completeness in Eq. 2.3. In particular, we replace

$[T_{ik}^{pr} + T_{kj}^{rq} = T_{ij}^{pq}]$ with $[\|T_{ik}^{pr} + T_{kj}^{rq} - T_{ij}^{pq}\| \leq \epsilon]$, where $\epsilon = 0.05 \cdot \max(h, w)$ (h and w are image height and width). $\beta = 20, \sigma_c = 0.05, \sigma_s = \epsilon$, and $\lambda = 0.01$ for all our experiments. The code will be available on our website.

2.4 Experiments

We compare our alignment performance with Congealing [122] (using SIFT), Collection Flow [104], DSP [110], and RASL [157]. All the baseline algorithms perform joint alignment across the whole image collection, except DSP, which is the state-of-the-art pairwise image matching algorithm and also used by us to initialize $\mathbf{T}_0$. We use publicly available code for all baselines except Collection Flow, for which we implement our own version in Matlab. All baselines are run with default parameters.

The image sets we use are sampled from the PASCAL-Part dataset [24]. To parse the images of each category into sets that are meaningful to align (a counter example would be aligning front-view cars to side-view cars), we run K-means clustering ($K = 10$) on the provided part visibility labels and coarse viewpoint annotations from the original VOC 2010 dataset, and select three representative clusters with largest sizes to evaluate for each category. A cluster is pruned if it has less than 10 images since joint alignment has little effect with few samples. The total number of image sets remaining is 47. In the interest of time, we limit the largest size of each set to 100. Images within each set are further resized to the average aspect ratio and maximum dimension of 150.

2.4.1 Part segment matching

We first evaluate alignment quality using human-annotated part segments. For quantitative evaluation, we use weighted intersection over union (IOU) with weights determined by the pixel area of each part, and report the mean performance over all sets for each category in Table 2.1. For categories without part annotations (boat, chair, table, and sofa) we simply use silhouette annotations for evaluation. We outperform all baselines on almost all categories.

We also visualize the part matching results in Fig. 2.4. Overall, our method is able to produce substantially more accurate correspondences than the baselines. The fact that many of the mistakes made by the initial DSP matching are corrected in our final output highlights the effectiveness of our joint alignment procedure.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
Congealing	0.26	0.40	**0.24**	0.48	0.68	0.46	0.39	0.19	0.49	0.30	0.42	**0.15**	0.26	0.32	0.18	0.38	0.35	0.71	0.45	0.58	0.38
RASL	0.26	0.40	0.22	0.49	0.70	0.46	0.42	0.19	0.51	0.30	**0.43**	**0.15**	0.25	0.33	0.18	0.38	0.34	0.72	0.47	0.64	0.39
CollectionFlow	0.29	0.40	0.22	0.49	0.69	0.46	0.41	**0.20**	0.51	0.28	0.35	**0.15**	0.25	0.28	0.18	0.36	0.34	0.66	0.44	0.59	0.38
DSP	0.25	0.46	0.21	0.48	0.63	0.50	0.45	0.19	0.48	0.30	0.37	0.14	0.26	0.35	0.13	0.40	0.37	0.66	0.48	0.62	0.39
Ours	**0.33**	**0.53**	**0.24**	**0.51**	**0.72**	**0.54**	**0.51**	**0.20**	**0.52**	**0.32**	0.41	**0.15**	**0.29**	**0.45**	**0.19**	**0.41**	**0.39**	**0.73**	**0.51**	**0.68**	**0.43**

Table 2.1: Weighted intersection over union (IOU) for part segment matching on 20 PASCAL VOC categories. Higher is better.

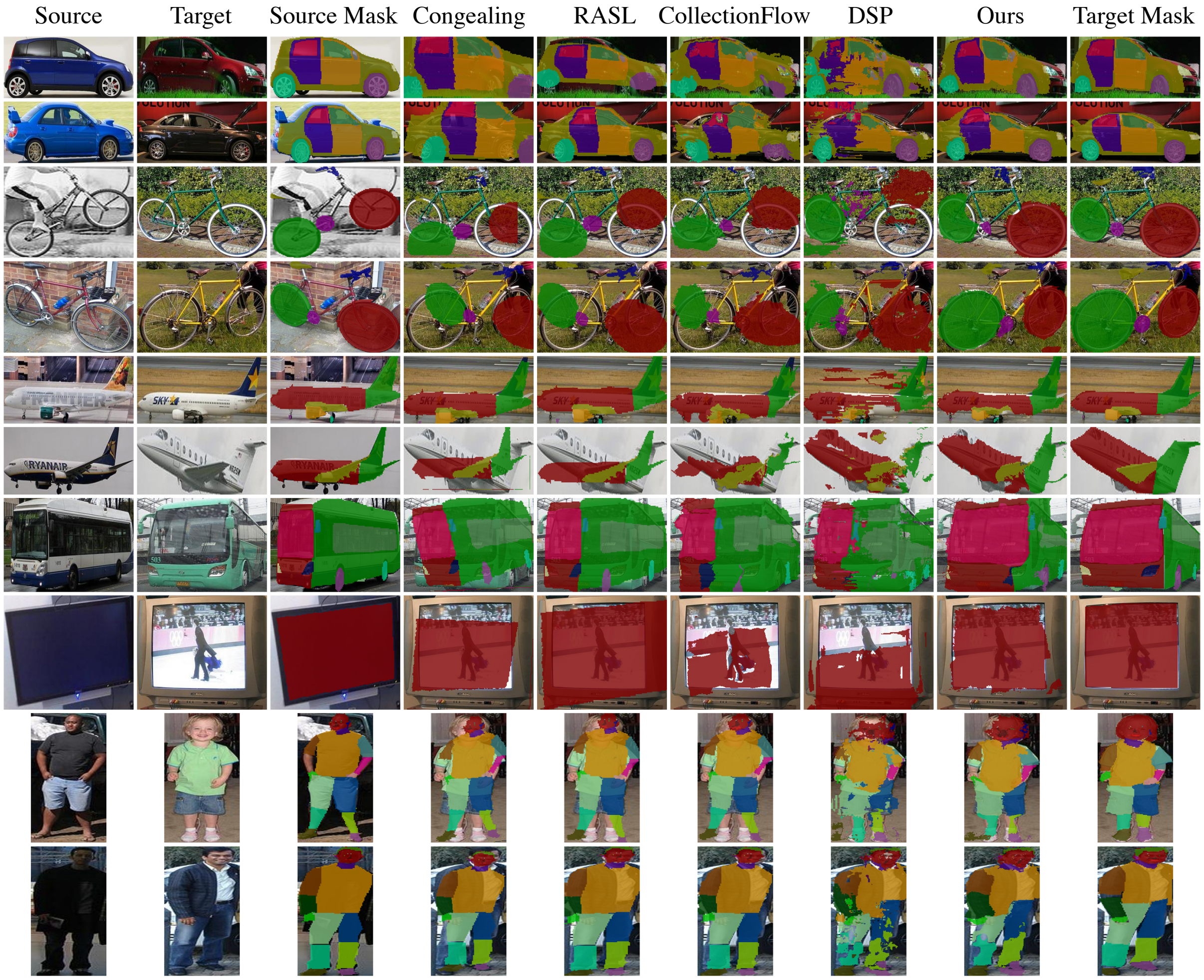


Figure 2.4: Correspondence visualization for different methods with color-coded part segments. Columns 1–2: source and target images. Column 3: annotated part segments for the source image. Column 4–8: predicted part correspondences on the target image using different methods. Column 9: annotated part segments for the target image (i.e. ground-truth). Overall, our correspondence output improves significantly over the initial DSP matching, and align part segments in greater precision than all baselines. (Best viewed in pdf.)

2.4.2 Keypoint matching

We next compare alignment accuracy using keypoint annotations for the 12 rigid PASCAL categories provided by [212]. We use the same set of images sampled in the previous experiment. The matching accuracy is assessed by the standard PCK measure [217], which defines a keypoint matching to be correct if the prediction falls within $\alpha \cdot \max(h, w)$ pixels of the ground-truth (h and w are image height and width respectively). For each category, we report the mean PCK over all sampled sets with different methods in Table 2.2. Again,

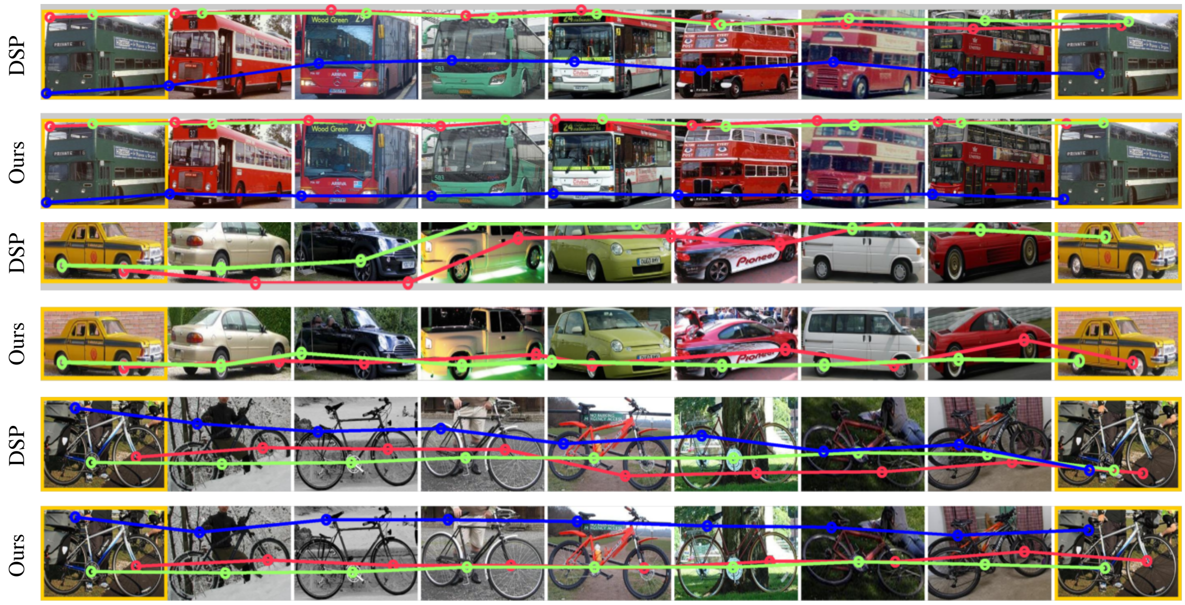


Figure 2.5: Comparison of keypoint correspondence tracks along a cycle in the graph (the first and the last image is the same for all examples) between DSP (initialization to our method) and ours. The keypoint correspondences become much more accurate and cycle-consistent after our joint alignment procedure.

our method substantially outperforms all baselines.

Fig. 2.5 compares the keypoint correspondence tracks between DSP (pairwise matching used for our initialization) and ours. DSP tracks tend to drift more as the path becomes longer, while our tracks are relatively stable and cycle-consistent along the graph (note that the first and the last image is the same for all examples).

	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
Congealing	0.12	0.23	0.03	0.22	0.19	0.14	0.06	**0.04**	0.12	0.07	0.08	0.06	0.11
RASL	0.18	0.17	0.04	0.33	0.31	0.17	0.09	**0.04**	0.12	0.10	0.11	0.23	0.16
CollectionFlow	0.16	0.17	0.04	0.31	0.25	0.16	0.09	0.02	0.08	0.07	0.06	0.09	0.12
DSP	0.17	0.30	**0.05**	0.19	0.33	0.34	0.09	0.03	0.17	0.12	0.12	0.18	0.17
Ours	**0.29**	**0.41**	**0.05**	**0.34**	**0.54**	**0.50**	**0.14**	**0.04**	**0.21**	**0.16**	**0.15**	**0.33**	**0.26**

Table 2.2: Keypoint matching accuracy (PCK) on 12 rigid PASCAL VOC categories ($\alpha = 0.05$). Higher is better.

2.4.3 Effect of image collection size

We hypothesize that the more images in the set, the better correspondences our method would produce as the cycle consistency measure becomes more robust. To verify this, we

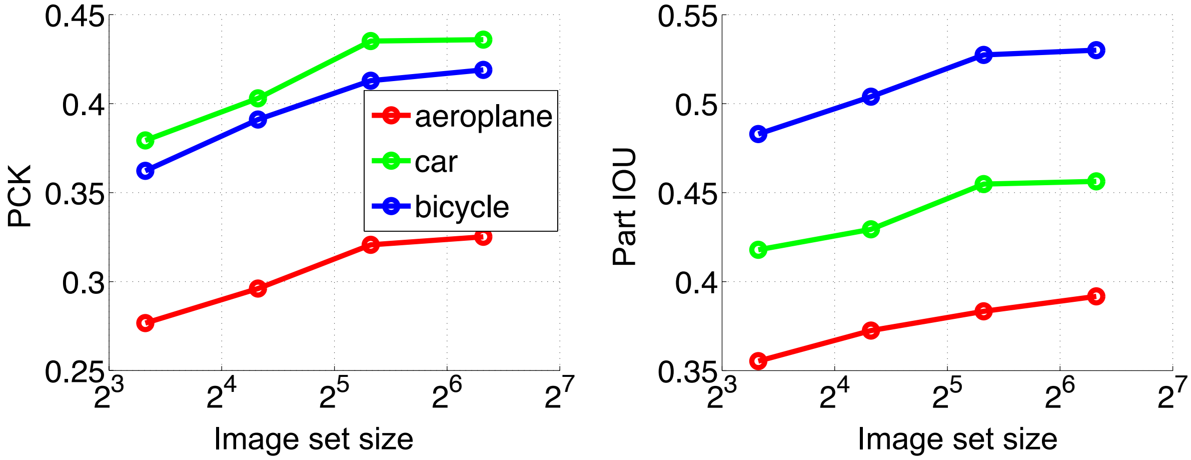


Figure 2.6: Alignment accuracy as a function of image set size using our method. The test set remains fixed as more images are included for joint alignment. Left: PCK. Right: Part segment IOU. Overall, more images leads to more accurate correspondences.

plot alignment accuracy as a function of image set size. Specifically, for car, aeroplane, and bicycle categories, we randomly sample 10 images as the test set for evaluation, and progressively add more images to construct the alignment set together with the 10 test images. As shown in Fig. 2.6, both keypoint and part-based matching accuracies indeed improve as more images become available.

2.4.4 Comparison with Mobahi et al. [149]

To compare with Mobahi et al. [149], we use their Mushroom dataset [149], comprised of 120 mushroom images and ground-truth foreground region and boundary masks for evaluation. After joint alignment, for each image pair, we compute both region and boundary matching scores as defined in [149]. The region score measures the fraction of foreground pixels in the warped source image that coincide with the foreground pixels in the target image (perfect alignment would result in a region score of 1; so higher is better). The boundary matching score measures the boundary displacement error (in pixels) between the warped source image and the target image (perfect alignment would result in a boundary score of 0; so lower is better). We average these scores computed for every pair of images in the dataset.

We obtain 0.84 and 6.44 for region and boundary alignment, respectively, compared to Mobahi et al.'s 0.73 and 5.69. Upon closer examination of why we perform worse in the boundary measure, we find our alignment to be more deformable than [149]. This can lead to highly accurate results (top four rows in Fig. 2.7) but also to very poor results if the deformation of the object is completely wrong (bottom row in Fig. 2.7). Such behavior could greatly affect boundary matching score as it is very sensitive to outliers.

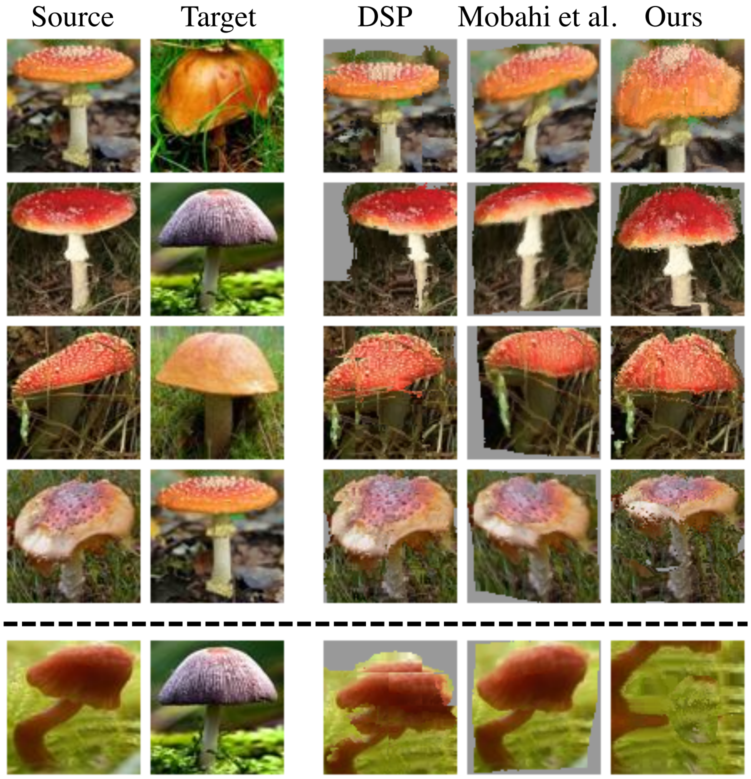


Figure 2.7: Comparison with the compositional model of [149]. Rows 1–4/5 are success/failure examples of our method.

2.4.5 Annotation-free Active Appearance Models

Training Active Appearance Models (AAM) [28] typically requires extensive human labeling of landmark keypoints. We show that it is possible to bypass the keypoint annotation step by using the cycle-consistency measure to identify keypoint surrogates. In particular, we can sum over the $SFCC$ score for all the flows coming out of a pixel p in image i by $\sum_{j=1}^{N}\sum_{k=1,k\notin\{i,j\}}^{N}[T_{ij}^{pq} = T_{ik}^{pr} + T_{kj}^{rq}]$, and use it to guide keypoint selection. Here is a simple pipeline: 1) Compute per-pixel consistency score using the above equation; 2) Pick a seed alignment image with the highest overall consistency; 3) Run max pooling to select a sparse set of candidate keypoints; 4) Do thresholding to select a final high-quality set of keypoints; 5) Obtain the keypoint correspondences for the rest of the image set according to the flows from the seed image to the target. Once the keypoints are established, standard AAM can be applied (we used the package from [192]). Fig. 2.8 shows sample results on cars.

2.4.6 Runtime complexity

For 50 images of size 150×150, our algorithm takes about 10 iterations to converge, each iteration taking about 10 minutes on a 3GHz, 16GB machine using a Matlab implementation.

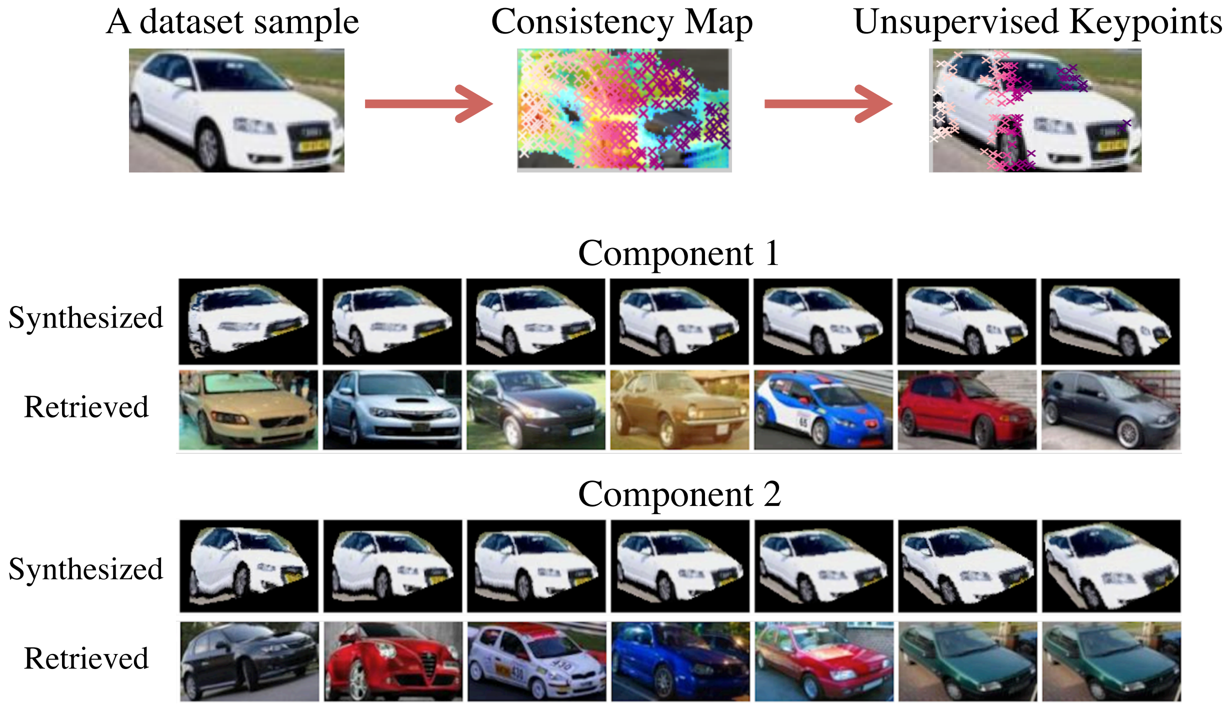


Figure 2.8: Visualization of unsupervised keypoint selection using cycle-consistency and its application to AAM (see Sec. 2.4.5 for more details). By varying the coefficients for AAM shape components, one can synthesize new instances that pertain to the variations within the image collection.

For 100 images, each iteration takes about an hour. There are two major computational bottlenecks: 1) The computation of priority is $\mathcal{O}(MN^4)$; 2) Consistency-weighted filtering is $\mathcal{O}(N^2M^2)$. One way to speed up the alignment process is to first break down the fully-connected graph into sub-clusters (to reduce N) and optimize the flows within each cluster, and then bring them together by connecting the closest matches between clusters. Our preliminary experiments show that the overall alignment accuracy won't be compromised much with such approximation as long as the size of each cluster is still considerably large. We plan to explore more options for efficiency improvement in the future.

2.5 Discussion

Now that object detection and retrieval are finally starting to work, it's possible to go from a very large, unorganized image collection to a relatively small set of coarsely-aligned images. But going from coarse to fine-grained pixel-wise correspondence is still very much an open problem, which this chapter is aiming to tackle. A successful solution could benefit many vision and graphics tasks. While achieving state-of-the-art performance, FlowWeb is overly dependent on the initialization quality, and scales poorly with the size of the image collection. In the next chapter, we address these issues by presenting a framework for learning dense correspondence networks with cycle consistency as the supervisory signal.

Chapter 3

Learning Dense Correspondence via 3D-guided Cycle Consistency

In the last chapter, we have shown that maximizing cycle consistency within an image collection is effective in obtaining high-quality dense correspondence. However, one issue with such collection-based methods is that they require a large set of images during runtime, which is often impractical. The natural question is: can we keep the same benefits of matching through a large collection of related images without storing them "explicitly"?

Recently, deep learning approaches have shown impressive results for problems where human-labeled ground truth is plentiful. However, for dense semantic correspondence it is infeasible to collect large-scale human labels since each pair of images has hundreds and thousands of pixel correspondences. Our key insight is that although we do not know what the ground-truth is, we know it should be **consistent** across instances of that category. We exploit this consistency as a supervisory signal to train a convolutional neural network to predict cross-instance correspondences between pairs of images depicting objects of the same category. For each pair of training images we find an appropriate 3D CAD model and render two synthetic views to link in with the pair, establishing a correspondence flow 4-cycle. We use ground-truth synthetic-to-synthetic correspondences, provided by the rendering engine, to train a deep network to predict synthetic-to-real, real-to-real and real-to-synthetic correspondences that are cycle-consistent with the ground-truth. At test time, no CAD models are required. We demonstrate that our end-to-end trained network supervised by cycle-consistency outperforms state-of-the-art pairwise matching methods in correspondence-related tasks [1].

[1] This work was originally published as *Learning dense correspondence via 3d-guided cycle consistency.* In CVPR, 2016 [232].

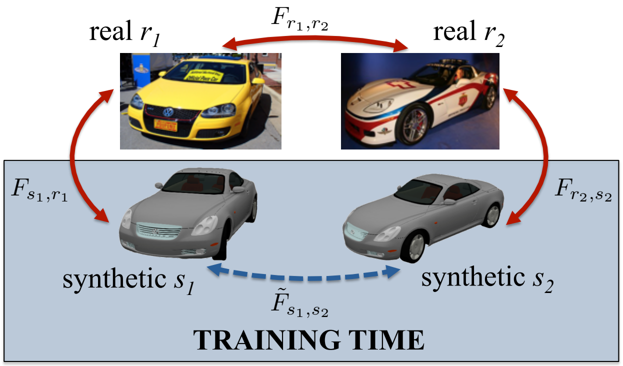


Figure 3.1: Estimating a dense correspondence flow field F_{r_1,r_2} between two images r_1 and r_2 — essentially, where do pixels of r_1 need to go to bring them into correspondence with r_2 — is very difficult. There is a large viewpoint change and the physical differences between the cars are substantial. We propose to *learn* to do this task by training a ConvNet using the concept of cycle consistency in lieu of ground truth. At training time, we find an appropriate 3D CAD model and establish a correspondence 4-cycle, training to minimize the discrepancy between $\tilde{F}_{s_1,s_2}$ and $F_{s_1,r_1} \circ F_{r_1,r_2} \circ F_{r_2,s_2}$, where $\tilde{F}_{s_1,s_2}$ is known by construction. At test time, no CAD models are used.

3.1 Motivation and background

In the past couple of years, deep learning has swept though computer vision like wildfire. One needs only to buy a GPU, arm oneself with enough training data, and turn the crank to see head-spinning improvements on most computer vision benchmarks. So it is all the more curious to consider tasks for which deep learning has *not* made much inroad, typically due to the lack of easily obtainable training data. One such task is *dense visual correspondence* – the problem of estimating a pixel-wise correspondence field between images depicting visually similar objects or scenes. Not only is this a key ingredient for optical flow and stereo matching, but many other computer vision tasks, including recognition, segmentation, depth estimation, etc. could be posed as finding correspondences in a large visual database followed by label transfer.

In cases where the images depict the same physical object/scene across varying viewpoints, such as in stereo matching, there is exciting new work that aims to use the commonality of the scene structure as supervision to learn deep features for correspondence [4, 47, 95, 69, 225]. But for computing correspondence *across different object/scene instances*, no learning method to date has managed to seriously challenge SIFT flow [129], the dominant approach for this task.

How can we get supervision for dense correspondence between images depicting different object instances, such as images r_1 and r_2 in Figure 3.1? Our strategy is to learn the things we don't know by linking them up to the things we do know. In particular, at training time, we use a large dataset of 3D CAD models [177] to find one that could link the two images, as shown in Figure 3.1. Here the dense correspondence between the two views of the same 3D model s_1 and s_2 can serve as our ground truth supervision (as we know precisely where each shape point goes when rendered in a different viewpoint), but the challenge is to use this information to train a network that can produce correspondence between two real images at test time.

A naive strategy is to train a network to estimate correspondence between the rendered views of the same 3D model, and then hope that the network could generalize to real images as well. Unfortunately, this does not work in practice (see Table 3.1), likely due to 1) the large visual difference between synthetic and real images and 2) the lack of cross-instance ground truth correspondence for training. Instead, we utilize the concept of *cycle consistency* of correspondence flows [87, 231, 236] – the notion that the composition of flow fields for any circular path through the image set should have a zero combined flow. Here, cycle consistency serves as a way to link the correspondence between real images and the rendered views into a single 4-cycle chain. We can then train our correspondence network using cycle consistency as the supervisory signal. The idea is to take advantage of the known synthetic-to-synthetic correspondence as ground-truth anchors that allow cycle consistency to propagate the correct correspondence information from synthetic to real images, without diverging or falling into a trivial solution. Here we could interpret the cycle consistency as a kind of "meta-supervision" that operates not on the data directly, but rather on how the data should behave. As we show later, such 3D-guided consistency supervision allows the network to learn cross-instance correspondence that potentially overcomes some of the major difficulties (e.g. significant viewpoint and appearance variations) of previous pairwise matching methods like SIFT flow [129]. Our approach could also be thought of as an extension and a reformulation of FlowWeb (described in the previous chapter) as a learning problem, where the image collection is stored implicitly in the network representation.

Recently, several works have applied convolutional neural networks to learn same-instance dense correspondence. FlowNet [45] learns an optical flow CNN with a synthetic Flying Chairs dataset that generalizes well to existing benchmark datasets, yet still falls a bit short of state-of-the-art optical flow methods like DeepFlow [205] and EpicFlow [165]. Several recent works have also used supervision from reconstructed 3D scene and stereo pairs [69, 225, 4]. However all these approaches are inherently limited to matching images of the same physical object/scene. Long *et al.* [135] use deep features learned from large-scale object classification tasks to perform intra-class image alignment, but found it to perform similarly to SIFT flow.

Our work is also partially motivated by recent progress in image-shape alignment that allows establishing correspondence between images through intermediate 3D shapes. Aubry *et al.* [6] learns discriminative patches for matching 2D images to their corresponding 3D CAD models, while Peng *et al.* [156] utilizes CAD models to train object detectors with few shots of labeled real images. In cases where depth data is available, deep learning methods have recently been applied to 3D object recognition and alignment between CAD models and RGB-D images [66, 181, 211]. Other works [86, 184] leverage image and shape collections for joint pose estimation and refining image-shape alignment, which are further applied to single-view object reconstruction and depth estimation. Although our approach requires 3D CAD models for constructing the training set, the image-shape alignment is jointly learned with the image-image alignment, and no CAD models are required at test time.

3.2 Approach

Our goal is to predict a dense flow (or correspondence) field $F_{a,b} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ between pairs of images a and b. The flow field $F_{a,b}(p) = (p_x - q_x, p_y - q_y)$ computes the relative offset from each point p in image a to a corresponding point q in image b. Given that pairwise correspondence might not always be well-defined (e.g. a side-view car and a frontal-view car do not have many visible parts in common), we additionally compute a matchability map $M_{a,b} : \mathbb{R}^2 \rightarrow [0, 1]$ predicting if a correspondence exists $M_{a,b}(p) = 1$ or not $M_{a,b}(p) = 0$.

We learn both the flow field and the matchability prediction with a convolutional neural network. Both functions are differentiable with respect to the network parameters, which could be directly learned if we had dense annotations for $F_{a,b}$ and $M_{a,b}$ on a large set of real image pairs. However, in practice it is infeasible to obtain those annotations at scale as they are either too time-consuming or ambiguous to annotate.

We instead choose a different route, and learn both functions by placing the supervision on the desired properties of the ground-truth, i.e. while we do not know what the ground-truth is, we know how it should behave. In this chapter, we use *cycle consistency* with 3D CAD models as the desired property that will be our supervisory signal. Specifically, for each pair of real training images r_1 and r_2, we find a 3D CAD model of the same category, and render two synthetic views s_1 and s_2 in similar viewpoint as r_1 and r_2, respectively (see 3.3.1 for more details). For each training quartet $< s_1, s_2, r_1, r_2 >$ we learn to predict flows from s_1 to r_1 (F_{s_1,r_1}) to r_2 (F_{r_1,r_2}) to s_2 (F_{r_2,s_2}) that are cycle-consistent with the ground-truth flow from s_1 to s_2 ($\tilde{F}_{s_1,s_2}$) provided by the rendering engine (similarly for the matchability prediction). By constructing consistency supervision through 3D CAD models, we aim to learn 2D image correspondences that potentially captures the 3D semantic appearance of the query objects. Furthermore, making $\tilde{F}_{s_1,s_2}$ be ground-truth by construction prevents the cycle-consistency optimization from producing trivial solutions, such as identity flows.

Sections 3.2.1 and 3.2.2 formally define our training objective for learning correspondence F and matchability M, respectively. Section 3.2.3 demonstrates how to obtain continuous approximation of discrete maps that allows end-to-end training. Section 3.2.4 describes our network architecture.

3.2.1 Learning dense correspondence

Given a set of training quartets $\{< s_1, s_2, r_1, r_2 >\}$, we train the CNN to minimize the following objective:

$$\sum_{<s_1,s_2,r_1,r_2>} \mathcal{L}_{flow}\left(\tilde{F}_{s_1,s_2},\ F_{s_1,r_1} \circ F_{r_1,r_2} \circ F_{r_2,s_2}\right), \tag{3.1}$$

where $\tilde{F}_{s_1,s_2}$ refers to the ground-truth flow between two synthetic views, F_{s_1,r_1}, F_{r_1,r_2} and F_{r_2,s_2} are predictions made by the CNN along the transitive path. The transitive flow composition $\bar{F}_{a,c} = F_{a,b} \circ F_{b,c}$ is defined as

$$\bar{F}_{a,c}(p) = F_{a,b}(p) + F_{b,c}(p + F_{a,b}(p)) \ , \tag{3.2}$$

which is differentiable as long as $F_{a,b}$ and $F_{b,c}$ are differentiable. $\mathcal{L}_{flow}(\tilde{F}_{s_1,s_2}, \bar{F}_{s_1,s_2})$ denotes the truncated Euclidean loss defined as

$$\mathcal{L}_{flow}(\tilde{F}_{s_1,s_2}, \bar{F}_{s_1,s_2}) = \sum_{p|\tilde{M}_{s_1,s_2}(p)=1} \min(\|\tilde{F}_{s_1,s_2}(p) - \bar{F}_{s_1,s_2}(p)\|^2, T^2) \ , \tag{3.3}$$

where $\tilde{M}_{s_1,s_2}(p)$ is the ground-truth matchability map provided by the rendering engine ($\tilde{M}_{s_1,s_2}(p) = 0$ when p is either a background pixel or not visible in s_2), and $T = 15$ (pixels) for all our experiments. In practice, we found the truncated loss to be more robust to spurious outliers for training, especially during the early stage when the network output tends to be highly noisy.

3.2.2 Learning dense matchability

Our training objective for matchability prediction also utilizes the cycle consistency signal:

$$\sum_{<s_1,s_2,r_1,r_2>} \mathcal{L}_{mat}\left(\tilde{M}_{s_1,s_2},\ M_{s_1,r_1} \circ M_{r_1,r_2} \circ M_{r_2,s_2}\right) , \tag{3.4}$$

where $\tilde{M}_{s_1,s_2}$ refers to the ground-truth matchability map between the two synthetic views, M_{s_1,r_1}, M_{r_1,r_2} and M_{r_2,s_2} are CNN predictions along the transitive path, and $\mathcal{L}_{mat}$ denotes per-pixel cross-entropy loss. The matchability map composition is defined as

$$\bar{M}_{a,c}(p) = M_{a,b}(p) M_{b,c}(p + F_{a,b}(p)) \ , \tag{3.5}$$

where the composition depends on both the matchability as well as the flow field.

Due to the multiplicative nature in matchability composition (as opposed to additive in flow composition), we found that training with objective 3.4 directly results in the network exploiting the clean background in synthetic images, which helps predict a perfect segmentation of the synthetic object in M_{s_1,r_1}. Once M_{s_1,r_1} predicts zero values for background points, the network has no incentive to correctly predict the matchability for background points in M_{r_1,r_2}, as the multiplicative composition has zero values regardless of the transitive predictions along M_{r_1,r_2} and M_{r_2,s_2}. To address this, we fix $M_{s_1,r_1} = \mathbf{1}$ and $M_{r_2,s_2} = \mathbf{1}$, and only train the CNN to infer M_{r_1,r_2}. This assumes that every pixel in $s_1(s_2)$ is matchable in $r_1(r_2)$, and allows the matchability learning to happen between real images. Note that this is still different from directly using $\tilde{M}_{s_1,s_2}$ as supervision for M_{r_1,r_2} as the matchability composition depends on the predicted flow field along the transitive path.

The matchability objective 3.4 is jointly optimized with the flow objective 3.1 during training, and our final objective can be written as $\sum_{<s_1,s_2,r_1,r_2>} \mathcal{L}_{flow} + \lambda \mathcal{L}_{mat}$ with $\lambda = 100$.

3.2.3 Continuous approximation of discrete maps

An implicit assumption made in our derivation of the transitive composition (Eq. 3.2 and 3.5) is that F and M are differentiable functions over continuous input, while images inherently consist of discrete pixel grids. To allow end-to-end training with stochastic gradient descent (SGD), we obtain continuous approximation of the full flow field and the matchability map with bilinear interpolation over the CNN predictions on discrete pixel locations. Specifically, for each discrete pixel location $\hat{p} \in \{1, \dots, W\} \times \{1, \dots, H\}$, the network predicts a flow vector $F_{a,b}(\hat{p})$ as well as a matchability score $M_{a,b}(\hat{p})$, and the approximation over all continuous points $p \in [1, W] \times [1, H]$ is obtained by:

$$\begin{aligned} F_{a,b}(p) &= \sum_{\hat{p} \in \mathcal{N}_p} (1 - |p_x - \hat{p}_x|)(1 - |p_y - \hat{p}_y|) F_{a,b}(\hat{p}) \\ M_{a,b}(p) &= \sum_{\hat{p} \in \mathcal{N}_p} (1 - |p_x - \hat{p}_x|)(1 - |p_y - \hat{p}_y|) M_{a,b}(\hat{p}) \ , \end{aligned}$$

where $\mathcal{N}_p$ denotes the four-neighbor pixels (top-left, top-right, bottom-left, bottom-right) of point p, or just p if it is one of the discrete pixels. This is equivalent to the differentiable image sampling with a bilinear kernel proposed in [92].

3.2.4 Network architecture

Our network architecture (see 3.2) follows the encoder-decoder design principle with three major components: 1) **feature encoder** of 8 convolution layers that extracts relevant features from both input images with shared network weights; 2) **flow decoder** of 9 fractionally-strided/up-sampling convolution (uconv) layers that assembles features from both input images, and outputs a dense flow field; 3) **matchability decoder** of 9 uconv layers that assembles features from both input images, and outputs a probability map indicating whether each pixel in the source image has a correspondence in the target.

All conv/uconv layers are followed by rectified linear units (ReLUs) except for the last uconv layer of either decoder, and the filter size is fixed to 3×3 throughout the whole network. No pooling layer is used, and the stride is 2 when increasing/decreasing the spatial dimension of the feature maps. The output of the matchability decoder is further passed to a sigmoid layer for normalization.

During training, we apply the same network to three different input pairs along the cycle ($s_1 \to r_1, r_1, \to r_2$, and $r_2 \to s_2$), and composite the output to optimize the consistency objectives 3.1 and 3.4.

3.3 Experimental Evaluation

In this section, we describe the details of our network training procedure, and evaluate the performance of our network on correspondence and matchability tasks.

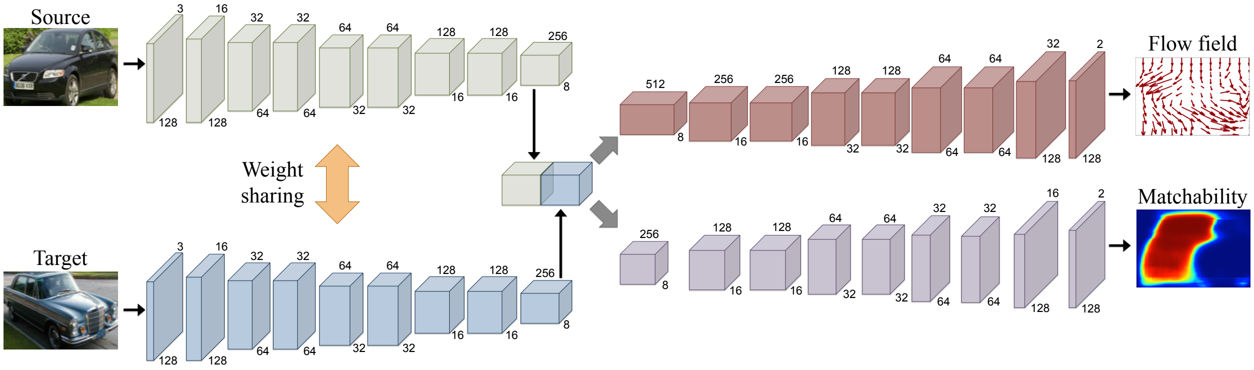


Figure 3.2: Overview of our network architecture, which consists of three major components: 1) **feature encoder** on both input images, 2) **flow decoder** predicting the dense flow field from the source to the target image and 3) **matchability decoder** that outputs a probability map indicating whether each pixel in the source image has a correspondence in the target. See Section 3.2.4 for more details.

3.3.1 Training set construction

The 3D CAD models we used for constructing training quartets come from the ShapeNet database [177], while the real images are from the PASCAL3D+ dataset [212]. For each object instance (cropped from the bounding box and rescaled to 128×128) in the train split of PASCAL3D+, we render all 3D models under the same camera viewpoint (provided by PASCAL3D+), and only use $K = 20$ nearest models as matches to the object instance based on the HOG [30] Euclidean distance. We then construct training quartets each consisting of two real images (r_1 and r_2) matched to the same 3D model and their corresponding rendered views (s_1 and s_2). On average, the number of valid training quartets for each category is about $80,000$.

3.3.2 Network training

We train the network in a category-agnostic manner (i.e. a single network for all categories). We first initialize the network (feature encoder + flow decoder pathway) to mimic SIFT flow by randomly sampling image pairs from the training quartets and training the network to minimize the Euclidean loss between the network prediction and the SIFT flow output on the sampled pair[2]. Then we fine-tune the whole network end-to-end to minimize the consistency loss defined in Eq. 3.1 and 3.4. We use the ADAM solver [114] with $\beta_1 = 0.9, \beta_2 = 0.999$, initial learning rate of 0.001, step size of $50,000$, step multiplier of 0.5 for $200,000$ iterations. We train with mini-batches of 40 image pairs during initialization and 10 quartets during fine-tuning.

[2] We also experimented with other initialization strategies (e.g. predicting ground-truth flows between synthetic images), and found that initializing with SIFT flow output works the best.

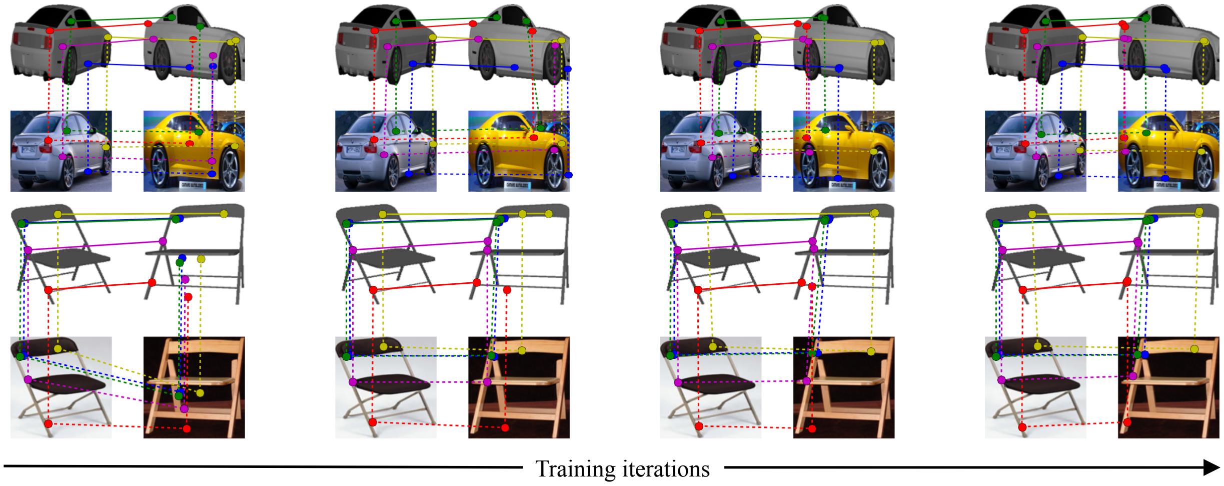


Figure 3.3: Visualizing the effects of consistency training on the network output. The randomly sampled ground-truth correspondences between synthetic images are marked in solid lines, and the correspondence predictions along the cycle (synthetic to real, real to real and real to synthetic) made by our network are marked in dashed lines. One can see that the transitive composition of our network output becomes more and more consistent with the ground-truth as training progresses, while individual correspondences along each edge of the cycle also tend to become more semantically plausible.

	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	**mean**
SIFT flow [129]	9.8	**23.3**	8.9	28.3	28.6	22.4	10.8	**13.2**	**17.9**	14.2	14.4	42.9	19.6
Long *et al.* [135]	10.4	22.8	7.6	30.8	28.4	21.1	10.2	12.7	13.5	12.9	12.6	38.5	18.5
CNN$_{I2S}$	9.1	14.7	5.2	25.9	25.4	23.7	11.9	11.3	13.4	16.8	11.3	45.2	17.8
CNN$_{init}$	8.6	20.3	8.5	29.4	24.3	20.1	9.9	11.6	15.4	11.6	12.5	40.2	17.7
CNN$_{init}$+ Synthetic ft.	10.2	22.2	8.7	30.4	24.5	21.3	10.2	12.1	15.7	12.0	12.8	40.5	18.4
CNN$_{init}$+ Consistency ft.	**11.3**	22.3	**10.1**	**40.3**	**40.3**	**33.3**	**15.0**	**13.2**	17.2	**17.4**	**16.7**	**51.1**	**24.0**

Table 3.1: Keypoint transfer accuracy measured in PCK ($\alpha = 0.1$) on the PASCAL3D+ categories. Overall, our final network (last row) outperforms all baselines (except on "bicycle" and "motorbike"). Notice the performance gap between our initialization (CNN$_{init}$) and the final network, which highlights the improvement made by cycle-consistency training.

We visualize the effect of our cycle-consistency training in Figure 3.3, where we sample some random points in the synthetic image s_1, and plot their predicted correspondences along the cycle $s_1 \rightarrow r_1 \rightarrow r_2 \rightarrow s_2$ to compare with the ground-truth in s_2. One can see that the transitive trajectories become more and more cycle-consistent with more iterations of training, while individual correspondences along each edge of the cycle also tend to become more semantically plausible.

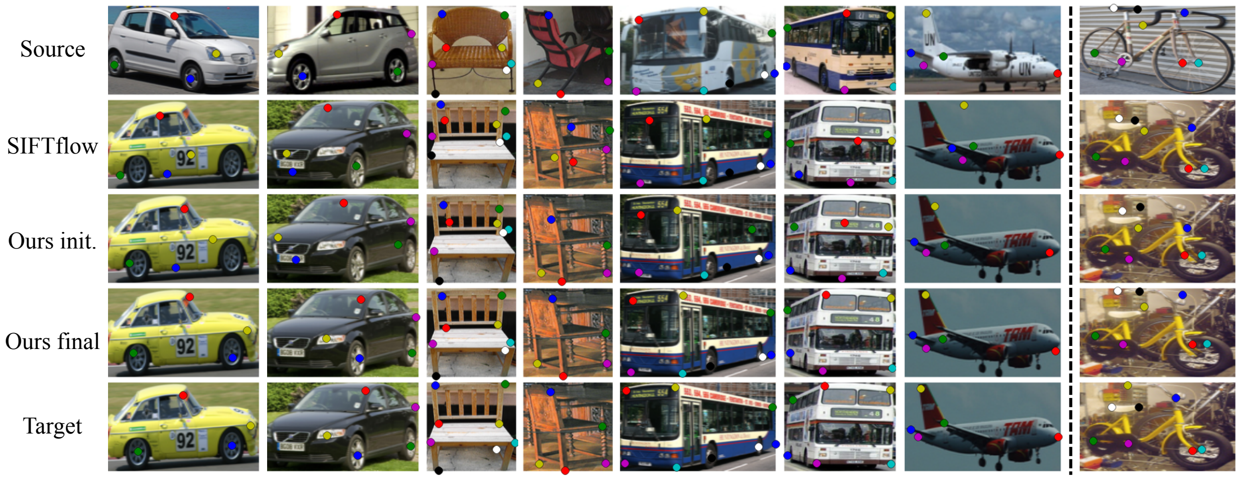


Figure 3.4: Comparison of keypoint transfer performance for different methods on example test image pairs. Overall, our consistency-supervised network (second-to-last row) is able to produce more accurate keypoint transfer results than the baselines. The last column shows a case when SIFT flow performs better than ours.

3.3.3 Keypoint transfer

We evaluate the quality of our correspondence output using the keypoint transfer task on the 12 categories from PASCAL3D+ [212]. For each category, we exhaustively sample all image pairs from the val split (not seen during training), and determine if a keypoint in the source image is transferred correctly by measuring the Euclidean distance between our correspondence prediction and the annotated ground-truth (if exists) in the target image. A correct transfer means the prediction falls within $\alpha \cdot \max(H, W)$ pixels of the ground-truth with H and W being the image height and width, respectively (both are 128 pixels in our case). We compute the percentage of correct keypoint transfer (PCK) over all image pairs as the metric, and provide performance comparison for the following methods in Table 3.1:

- SIFT flow [129] – A classic method for dense correspondence using SIFT feature descriptors and hand-designed smoothness and large-displacement priors. We also ran preliminary evaluation on a more recent follow-up based on deformable spatial pyramids [110], and found it to perform similarly to SIFT flow.

- Long *et al.* [135] – Similar MRF energy minimization framework as SIFT flow but with deep features learned from the ImageNet classification task.

- CNN$_{I2S}$ – Our network trained on real image pairs with correspondence inferred by compositing the output of an off-the-shelf image-to-shape alignment algorithm [86] and the ground-truth synthetic correspondence (i.e. obtaining direct supervision for F_{r_1,r_2} through $F_{r_1,s_1} \circ \tilde{F}_{s_1,s_2} \circ F_{s_2,r_2}$, where F_{r_1,s_1} and F_{s_2,r_2} are inferred from [86]).

- CNN$_{init}$ – Our network trained to mimic SIFT flow.

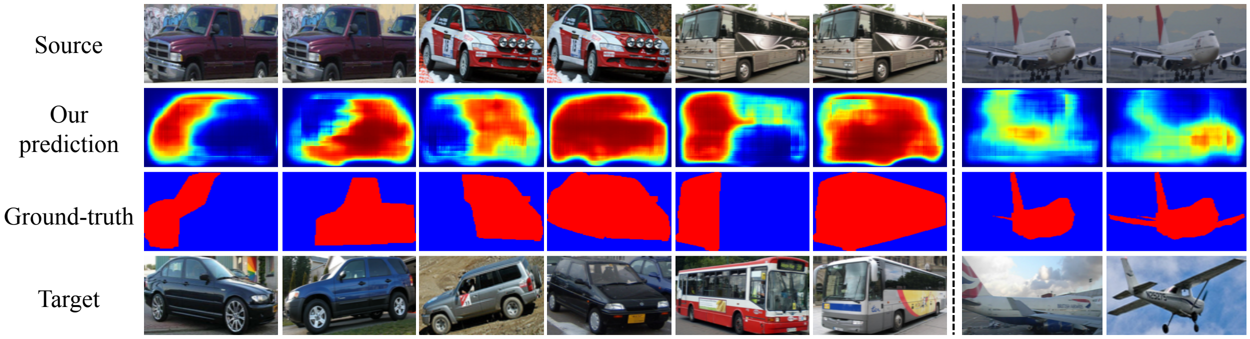


Figure 3.5: Sample visualization of our matchability prediction. Notice how the prediction varies for the same source image when changing only the target image. The last two columns demonstrate a typical failure mode of our network having trouble localizing the fine boundaries of the matchable regions.

- CNN$_{init}$+ Synthetic ft. – fine-tuning on synthetic image pairs with ground-truth correspondence after initialization with SIFT flow.
- CNN$_{init}$+ Consistency ft. – fine-tuning with our objectives 3.1 and 3.4 after initialization with SIFT flow.

Overall, our consistency-supervised network significantly outperforms all other methods (except on “bicycle” and “motorbike” where SIFT flow has a slight advantage). Notice the significant improvement over the initial network after consistency fine-tuning. The performance gap between the last two rows of Table 3.1 suggests that consistency supervision is much more effective in adapting to the real image domain than direct supervision from synthetic ground-truth.

Figure 3.4 compares sample keypoint transfer results using different methods. In general, our final prediction tends to match the ground-truth much better than the other baselines, and could sometimes overcome substantial viewpoint and appearance variation where previous methods, like SIFT flow, are notoriously error-prone.

3.3.4 Matchability prediction

We evaluate matchability prediction using the PASCAL-Part dataset [24], which provides human-annotated part segment labeling[3]. For each test image pair, a pixel in the source image is deemed matchable if there exists another pixel in the target image that shares the same part label, and all background pixels are unmatchable. We measure the performance by computing the percentage of pixels being classified correctly. For our method, we classify a pixel as matchable if its probability is > 0.5 according to the network prediction. To

[3] For categories without part labels, including boat, chair, table and sofa, we use the foreground segmentation mask instead.

	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	**mean**
SIFT flow [129]	66.2	**62.7**	49.5	50.5	52.0	64.5	50.7	50.5	**80.6**	49.6	58.5	50.2	57.1
Ours	**75.8**	61.0	**66.7**	**67.1**	**67.3**	**72.0**	**66.1**	**68.4**	68.0	**71.2**	**64.4**	**65.1**	**67.8**

Table 3.2: Performance comparison of matchability prediction between SIFT flow and our method (higher is better). See Section 3.3.4 for more details on the experiment setup.

obtain matchability prediction for SIFT flow, we compute the L_1 norm of the SIFT feature matching error for each source pixel after the alignment, and a pixel is predicted to be matchable if the error is below a certain threshold (we did grid search on the training set to determine the threshold, and found $1,000$ to perform the best). Table 3.2 compares the classification accuracy between our method and SIFT flow prediction (chance performance is 50%). Our method significantly outperforms SIFT flow on all categories except "bicycle" and "motorbike" (67.8% vs. 57.1% mean accuracy).

We visualize some examples of our matchability prediction in Figure 3.5. Notice how the prediction varies when the target image changes with the source image being the same.

3.3.5 Shape-to-image segmentation transfer

Although so far we are mostly interested in finding correspondence between real images, a nice byproduct of our consistency training is that the network also implicitly learns cross-domain, shape-to-image correspondence, which allows us to transfer per-pixel labels (e.g. surface normals, segmentation masks, etc.) from shapes to real images. As a proof of concept, we ran a toy experiment on the task of segmentation transfer. Specifically, we construct a shape database of about 200 shapes per category, with each shape being rendered in 8 canonical viewpoints. Given a query real image, we apply our network to predict the correspondence between the query and each rendered view of the same category, and warp the query image according to the predicted flow field. Then we compare the HOG Euclidean distance between the warped query and the rendered views, and retrieve the rendered view with minimum error whose correspondence to the query image on the foreground region is used for segmentation transfer. Figure 3.6 shows sample segmentation using different methods. We can see that our learned flows tend to produce more accurate segmentation transfer than SIFT flow using the same pipeline. In some cases our output can even segment challenging parts such as the bars and wheels of the chairs.

3.4 Discussion

In this chapter, we described a framework for using cycle-consistency as a supervisory signal to learn dense cross-instance correspondences. Not only did we find that this kind of supervision is surprisingly effective, but also that the idea of learning with cycle-consistency could potentially be fairly general. One could apply the same idea to construct other training

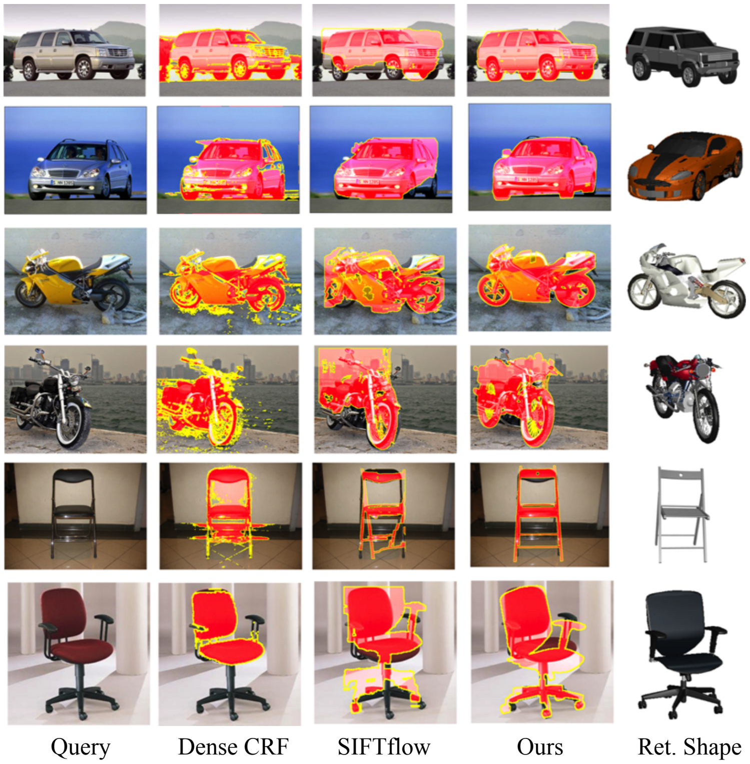


Figure 3.6: Visual comparison among different segmentation methods. From left to right: input query image, segmentation by [115], segmentation transferred using SIFT flow, segmentation transferred using our flow and the retrieved shape whose segmentation is used for transferring. See Section 3.3.5 for more details.

scenarios, as long as the ground-truth of one or more edges along the cycle is known. Since the publication of this work, cycle consistency has been successfully applied to other domains too such as unpaired image-to-image translation [238, 112] and single-view depth estimation [59]. We hope that this work will inspire more efforts to tackle tasks with little or no direct labels by exploiting cycle consistency or other types of indirect or "meta"-supervision.

Part II

Learning scene geometry

Chapter 4

View Synthesis by Appearance Flow

This chapter addresses the problem of *novel view synthesis*: given an input image, synthesizing new images of the same object or scene observed from arbitrary viewpoints. We approach this as a learning task but, critically, instead of learning to synthesize pixels from scratch, we learn to *copy* them from the input image. Our approach exploits the observation that the visual appearance of different views of the same instance is highly correlated, and such correlation could be explicitly learned by training a convolutional neural network (CNN) to predict *appearance flows* – 2-D coordinate vectors specifying which pixels in the input view could be used to reconstruct the target view. Furthermore, the proposed framework easily generalizes to multiple input views by learning how to optimally combine single-view predictions. We show that for both objects and scenes, our approach is able to synthesize novel views of higher perceptual quality than previous CNN-based techniques at the time of publication [1].

Notice that the appearance flows naturally emerge as the result of learning view synthesis as we did not need to provide any ground-truth supervision for the flows. The emergence of explicit pixel associations further inspired the work in the next two chapters, where we utilize this observation to learn scene geometry without labeled data.

4.1 Introduction

When we *look at* a 2D image, numerous psychophysics experiments tell us that what we are *seeing* is not the 2D image but the 3D object that it represents. For example, one classic experiment demonstrates that people excel at "mental rotation" [178] – predicting what a given object would look like after a known 3D rotation is applied. In this paper, we study the computational equivalent of mental rotation called *novel view synthesis*. Given one or more input images of an object or a scene plus the desired viewpoint transformation, the goal is to synthesize a new image capturing this novel view.

[1] This work was originally published as *View Synthesis by Appearance Flow.* In ECCV, 2016 [235].

Besides purely academic interest (how well can this be done?), novel view synthesis has a plethora of practical applications, mostly in computer graphics and virtual reality. For example, it could enable photo editing programs like Photoshop to manipulate objects in 3D instead of 2D. Or it could help create full virtual reality environments based on historic images or video footage.

The ways that novel view synthesis has been approached in the past fall into two broad categories: geometry-based approaches and learning-based approaches. Geometric approaches try to first estimate (or fake) the approximate underlying 3D structure of the object, and then apply some transformation to the pixels in the input image to produce the output [83, 154, 226, 81, 230, 23, 109]. Besides the requirement of somehow estimating the 3D structure, which is a difficult task by itself, the other major downside of these methods is that they produce holes in places where the source image does not have the appropriate visual content (e.g. the back side of an object). In such cases, various types of texture hole-filling are sometimes used but they are not always effective.

Learning-based approaches, on the other hand, argue that novel view synthesis is fundamentally a learning problem, because otherwise it is woefully underconstrained. Given a side of a car, there is no way to ever guess what the front of this car looks like, unless the system has observed other fronts of cars so it can make an educated guess. Such methods typically try, at training time, to build a parametric model of the object class, and then use it at test time, together with the input image, to generate a novel view. Unfortunately, parametric image generation is an open research topic, and currently the results of such methods are often too blurry).

In this chapter, we propose to combine the benefits of both types of approaches, while also avoiding their pitfalls. Like geometric methods, we propose to use the pixels of the input image as much as possible, instead of trying to synthesize new ones from scratch. At the same time, we will use a learning-based approach to implicitly capture the approximate geometry of the object, avoiding the explicit estimation of the 3D structure. Our model also learns the appearance correlation between different parts of the object that enables synthesizing the backside of the object.

Conceptually, our approach is quite simple: we train a deep generative convolutional encoder-decoder model, similar to [188], but instead of generating RGB values for each pixel in the target view, we generate an *appearance flow* vector indicating the corresponding pixel in the input view to steal from. This way, the model does not need to learn how to generate pixels from scratch – just where to copy from the input view. In addition to making the learning problem more tractable, it also provides a natural way of preserving the identity and structure of the input instance – a task typically difficult for conventional learning approaches. We demonstrate the applicability of our approach by synthesizing views corresponding to rotation of objects and ego-motion in scenes. We further extend our framework to leverage multiple input views and empirically show the quantitative as well as perceptual improvements obtained with our approach.

4.2 Background

Feature learning by disentangling pose and identity. Synthesizing novel views of objects can be thought of as decoupling pose and identity and has long been studied as part of feature learning and view-invariant recognition. Hinton *et al.* [79] learned a hierarchy of "capsules", computational units that locally transform their input, for generating small rotations to an input stereo pair, and argued for the use of similar units for recognition. More recently, Jaderberg *et al.* [93] demonstrated the use of computational layers that perform global spatial transformation over their input features as useful modules for recognition tasks. Jayaraman *et al.* [94] studied the task of synthesizing features transformed by ego-motion and demonstrated its utility as an auxiliary task for learning semantically useful feature space. Cheung *et al.* [26] proposed an auto-encoder with decoupled semantic units representing pose, identity *etc.* and latent units representing other factors of variation and showed that their approach was capable of generating novel views of faces. Kulkarni *et al.* [117] introduced a similarly motivated variational approach for decoupling and manipulating the factors of variation for images of faces. While the feature-learning approaches convincingly demonstrated the ability to disentangle factors of variation, the view manipulations demonstrated were typically restricted to small rotations or categories with limited shape variance like digits and faces.

CNNs for view synthesis. A recent interest in learning to synthesize views for more challenging objects under diverse view variations has been driven by the ability of Convolutional Neural Networks (CNNs) [51, 123] to function as image decoders. Dosovitiskiy *et al.* [2] learned a CNN capable of functioning as a renderer: given an input graphics code containing identity, pose, lighting *etc.* their model could render the corresponding image of a chair. Yang *et al.* [216] and Tatarchenko *et al.* [188] built on this work using the insight that the graphics code, instead of being presented explicitly, can be implicitly captured by an example source image along with the desired transformation. Yang *et al.* [216] learned a decoder to obtain implicit pose and identity units from the input source image, applied the desired transformation to the pose units, and used a decoder CNN to render the desired view. Concurrently, Tatarchenko *et al.* [188] followed a similar approach without the explicit decoupling of identity and pose to obtain similar results. A common module in these approaches is the use of a decoder CNN to generate the pixels corresponding to the transformed view from an implicit/explicit graphics code. Our work demonstrates that predicting appearance flows instead of pixels leads to significant improvements.

Geometric view synthesis. An alternative paradigm for synthesizing novel views of an object is to explicitly model the underlying 3D geometry. In cases when more than one input view is available, modern multi-view stereo algorithms (see Furukawa and Hernandez [52] for an excellent tutorial) have demonstrated results of impressive visual quality. However, these methods fundamentally rely on finding visual correspondences – pixels that is in common across the views – so they break down when there are only a couple of views from very different viewpoints. In cases when only a single view is available, user interaction had typically been needed to help define a coarse geometry for the object or scene [83, 154, 226, 230, 23]. More recently, large Internet collections of stock 3D shape models have been leveraged to get 3D geometry for a wide range of common objects. For example, Kholgade *et al.* [109] obtained realistic renderings of novel views of an object by transferring texture from the corresponding 3D model, though they required manual annotation of the exact 3D model and its placement in the image. Rematas *et al.* [163] employed a similar technique after automatically inferring the closest 3D model from a shape collection as well as explicitly obtaining pose via a learnt system to situate the 3d model in the image. Their approach, however, is restricted to rendering the closest model in the shape collection instead of the original object. Su *et al.* [183] overcome this restriction by interpolating between several similar models from the shape collections, though they only demonstrate their technique for generating HOG [31] features for novel views. Unlike the CNN based learning approaches, these geometry-based methods require access to a shape collection during inference and are limited by the intermediate bottlenecks of inferring pose and retrieving similar models.

Image-based Rendering. The idea of directly re-using the pixels from available images to generate new views has been popular in computer graphics. Debevec *et al.* [32] used the underlying geometry to composite multiple views for rendering novel views. Light-field/lumigraph [126, 64] rendering presented an alternate setup where a structured, dense set of views is available. Buehler *et al.* [13] presented a unifying framework for these image-based rendering techniques. The recent DeepStereo work by Flynn *et al.* [48] is a learning-based extension that performs compositing through learned geometric reasoning using a CNN, and can generate intermediate views of a scene by interpolating from a set of surrounding views. While these methods yield high-quality novel views, they do so by composting the corresponding input image rays for each output pixel and can therefore only generate already seen content, (e.g. they cannot create the rear-view of a car from available frontal and side-view images).

Texture Synthesis and Epitomes. Reusing pixels of the input image to synthesize new visual context is also at the heart of non-parametric texture synthesis approaches. In texture synthesis [39, 9], the synthesized image is pieced together by combining samples of the input texture image in a visually consistent way, whereas for texture transfer [78, 38], an additional constraint aims to make the overall result also mimic a secondary "source" image. A related line of work uses *epitomes* [98] as a generative model for a set of images. The key idea is to use a condensed image as a palette for sampling patches to generate new images. In a similar spirit, our approach can be thought of as generating novel views of an object using the original image as an epitome.

4.3 Approach

Our approach to novel view synthesis is based on the observation that the appearance (texture, shape, color, etc.) of different views of the same object/scene is highly correlated, and in many cases even a single input view contains rich amount of information for inferring various novel views. For instance, given the side view of a car, one could extract appearance properties such as the 3D shape, body color, window layout and wheel types of the query instance that are sufficient for reconstructing many other views.

In this work, we *explicitly* infer the appearance correlation between different views of a given object/scene by training a convolutional neural network that takes 1) an input view and 2) a desired viewpoint transformation, and predicts a dense *appearance flow field* (AFF) that specifies how to reconstruct the target view using pixels from the input view. Specifically, for each pixel i in the target view, the appearance flow vector $f^{(i)} \in \mathbb{R}^2$ specifies the coordinate at the input view where the pixel value is sampled to reconstruct pixel i. The notion of appearance flow field is closely related to the nearest neighbor field (NNF) in PatchMatch [9], except that NNF is explicitly defined on a distance function between two patches, while our appearance flow field is the output of a CNN after end-to-end training for cross-view reconstruction.

The benefits of predicting the appearance flow field over raw pixels of the target view are three-fold: 1) It alleviates the perceptual blurriness in images generated by CNN trained with L_p loss. By constraining the CNN to only utilize pixels available in the input view, we are able to avoid the undesirable local minimum attained by predicting the mean (when $p = 2$) colors around texture/edge boundaries that lead to blurriness in the resulting image (e.g. see Section 4.4 for empirical comparison). 2) The color identity of the instance is preserved by construction since the synthesized view is reconstructed using only pixels from the same instance; 3) The appearance flow field enables intuitive interpretation of the network output since we can visualize exactly how the target view is constructed with the input pixels (e.g. see Figure 4.5).

We first describe our training objective and the network architecture for the setting of a single input view in Section 4.3.1, and then present a simple extension in Section 4.3.2 that allows the network to learn how to combine individual predictions when multiple input views are available.

4.3.1 Learning view synthesis via appearance flow

Recall that our goal is to train a CNN that, given an input view I_s and a relative viewpoint transformation T, synthesizes the target view I_t by sampling pixels from I_s according to the predicted appearance flow field. This can be formalized as minimizing the following objective:

$$\text{minimize} \sum_{<I_s, I_t, T> \in \mathcal{D}} \|I_t - g(I_s, T)\|_p, \quad \text{subject to } \; g^{(i)}(I_s, T) \in \{I_s\}, \forall i \;, \tag{4.1}$$

where $\mathcal{D}$ is the set of training tuples, $g(\cdot)$ refers to the CNN whose weights we wish to optimize, $\|\cdot\|_p$ denotes the L_p norm[2], and i indexes over pixels of the synthesized view. Internally, the CNN computes a dense flow field f, where each element $f^{(i)} = (x^{(i)}, y^{(i)})$ specifies the pixel sampling location (in the coordinate frame of the input view) for constructing the output $g^{(i)}(I_s, T)$. To allow end-to-end training via stochastic gradient descent

[2] We use $p = 1$ in all our experiments, but similar results can be obtained with L_2 norm as well.

when $f^{(i)}$ falls into a sub-pixel coordinate, we rewrite the constraint of Eq. 4.1 in the form of bilinear interpolation:

$$g^{(i)}(I_s, T) = \sum_{q \in \{\text{neighbors of } (x^{(i)}, y^{(i)})\}} I_s^{(q)}(1 - |x^{(i)} - x^{(q)}|)(1 - |y^{(i)} - y^{(q)}|) \ , \qquad (4.2)$$

where q denotes the 4-pixel neighbors (top-left, top-right, bottom-left, bottom-right) of $(x^{(i)}, y^{(i)})$. This is also known as differentiable image sampling with a bilinear kernel, and its (sub)-gradient with respect to the CNN parameters could be efficiently computed [93].

Network architecture Our view synthesis network (Figure 4.1) follows a similar high-level design as [216] and [188] with three major components:

1. Input view encoder – extracts relevant features (e.g. color, pose, texture, shape, etc.) of the query instance (6 conv + 2 fc layers).
2. Viewpoint transformation encoder – maps the specified relative viewpoint to a higher-dimensional hidden representation (2 fc layers).
3. Synthesis decoder – assembles features from the two feature encoders, and outputs the appearance flow field that reconstructs the target view with pixels from the input view (2 fc + 6 uconv layers).

All the convolution, fully-connected and fractionally-strided/up-sampling convolution (uconv) layers are followed by rectified linear units except for the last flow decoder layer.

Foreground prediction For synthesizing object views, we also train another network that predicts the foreground segmentation mask of the target view. The architecture is the same as the synthesis network in Figure 4.1, except that in this case the last layer predicts a per-pixel binary classification mask (0 is background and 1 is foreground), and the network is trained with cross-entropy loss. At test time, we further apply the predicted foreground mask to the synthesized view.

4.3.2 Learning to leverage multiple input views

A single view of the object sometimes might not contain sufficient information for inferring an arbitrary target view. For instance, it would be very challenging to infer the texture details of the wheel spoke given only the frontal view of a car, and similarly, the side view of a car contains little to none information about the appearance of the head lights. Thus, it would be ideal to develop a mechanism that could leverage the individual strength of different input views to synthesize target views that might not be feasible with any input view alone.

To achieve this, we modify our view synthesis network to also output a *soft* confidence mask C_j that indicates per-pixel prediction quality using input view s_j, which could be implemented by adding an extra output channel to the last decoder layer. The confidence masks for all input views are further normalized to sum to one at each pixel location: $\bar{C}_j^{(i)} =$

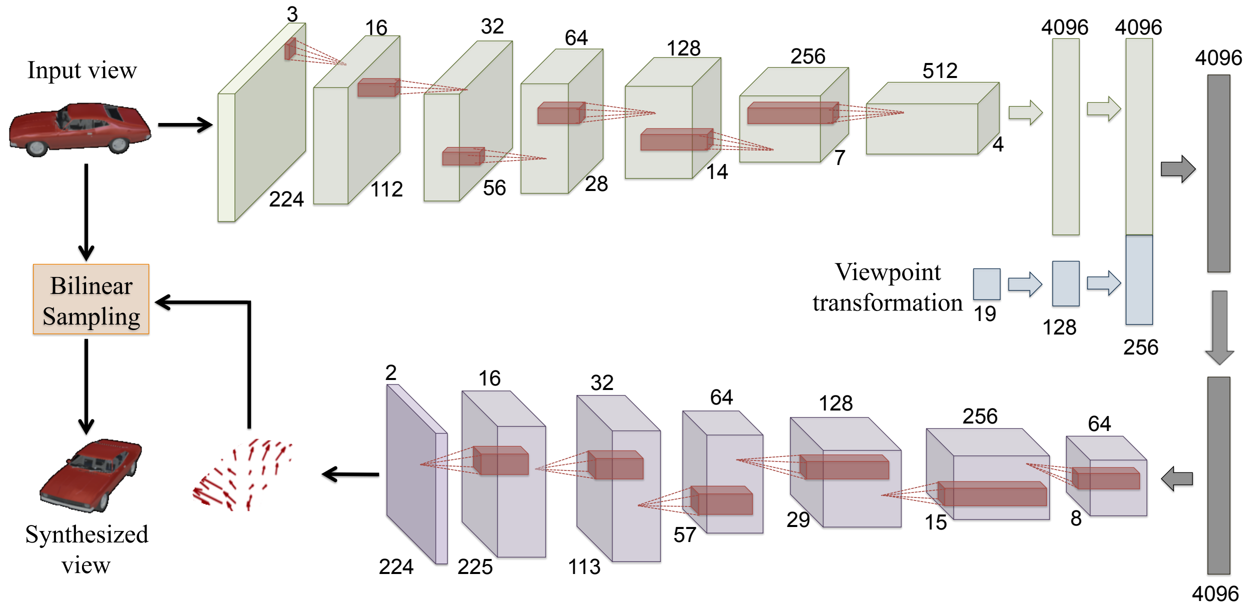


Figure 4.1: Overview of our single-view network architecture. We follow an encoder-decoder framework where the input view and the desired viewpoint transformation are first encoded via several convolution and fully-connected layers, and then a decoder consisting of two fully-connected and six up-sampling convolution layers outputs an appearance flow field, which in conjunction with the input view yields the synthesized view through a bilinear sampling layer. All the layer weights are learned end-to-end through back-propagation.

$C_j^{(i)} / \sum_{k=1}^{N} C_k^{(i)}$, where N denotes the number of input views. Intuitively, $\bar{C}_j^{(i)}$ is an estimator of *relative* prediction quality using input view j at pixel i, and by using $\bar{C}_j$ as a hypothesis selection mask, the final joint prediction is simply a weighted combination of hypotheses predicted by different input views: $\sum_{j=1}^{N} \bar{C}_j * g(I_{s_j}, r_j)$. Figure 4.2 illustrates the architecture of our multi-view network that is also end-to-end learnable.

Comparison with DeepStereo [48] While the general idea of learning hypothesis selection for view synthesis has been recently explored in [48], there are a few key differences between our framework and [48]: 1) We do not require projecting the input image stack onto a planesweep volume that prohibits their method from synthesizing pixels that are invisible in the input views (i.e. view extrapolation); 2) Unlike [48], who have a fixed number of input views, our multi-view network is more flexible at both training and test time as it could take in an *arbitrary* number of input views for joint prediction, which is particularly beneficial when the number of input views varies at test time.

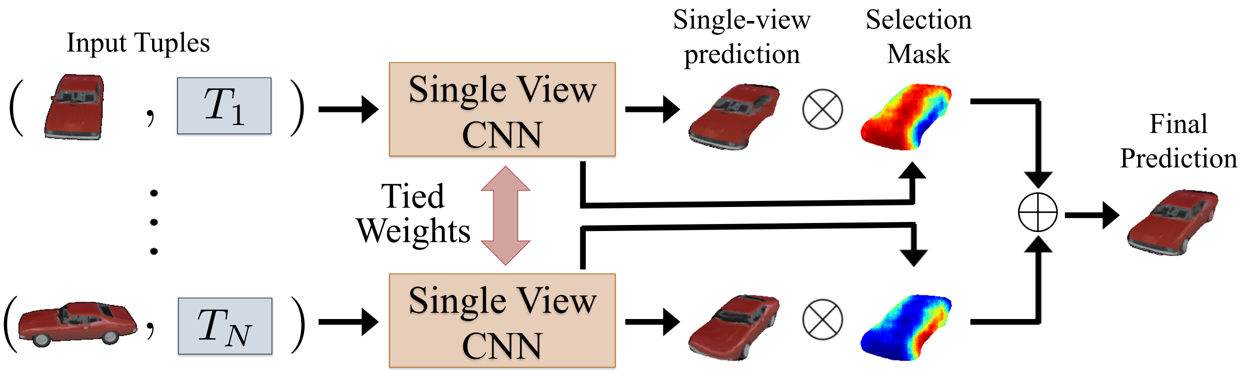


Figure 4.2: Overview of our multi-view network architecture (⊗: per-pixel product, ⊕: per-pixel normalized sum). For each input view, we use a single-view CNN (same as Figure 4.1 but with an extra output channel) with shared weights to independently predict the target view as well as a per-pixel selection/confidence mask. The final target view prediction is obtained by linearly combining the predictions from each view weighted by the selection masks.

4.4 Experiments

To evaluate the performance of our view synthesis approach, we conduct experiments on ShapeNet objects, including *car* and *chair*. Our main baseline is the recent work of Tatarchenko et al [188] that synthesizes novel views by training a CNN to directly generate pixels. For fair comparison, we use the same number of network layers for their method and ours, and for experiments on multiple input views we extend their method to output hypothesis selection masks as described in Section 4.3.2.

Network training details We train the networks using a modified version of Caffe [96] to support the bilinear sampling layer. We use the ADAM solver [114] with $\beta_1 = 0.9, \beta_2 = 0.999$, initial learning rate of 0.0001, step size of $50,000$ and a step multiplier $\gamma = 0.5$.

4.4.1 Novel view synthesis for objects

Data setup We train and evaluate our view synthesis CNN for objects using the ShapeNet database [17]. In particular, we split the available shapes ($7,497$ cars and 700 chairs[3]) of each category into 80% for training and 20% for testing. For each shape, we render a total of 504 viewing angles (azimuth ranges from 0 to 355 degrees, and elevation ranges from 0 to 30 degrees, both at steps of 5 degrees) with fixed camera distance. For simplicity, we limit the viewpoint transformation for CNN to a discrete set of 19 azimuth variations ranging

[3] The original ShapeNet core release contains a total of $6,778$ chair models. However, a majority of the models are of low visual quality (e.g. texture-less), and we only keep a subset of 700 high-quality ones for our experiments.

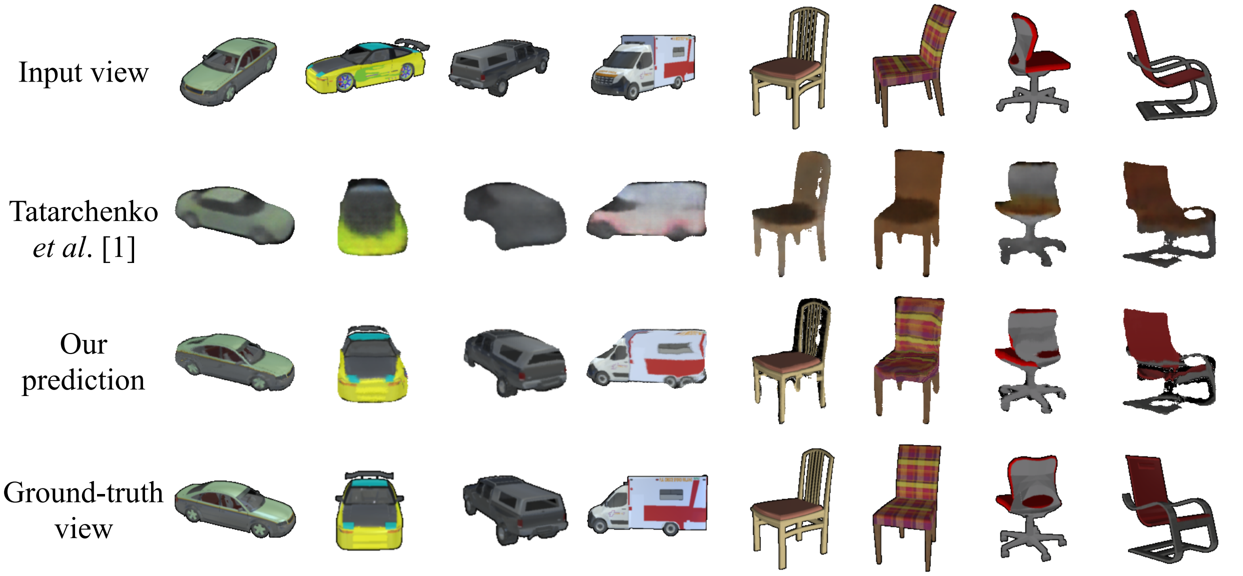


Figure 4.3: Comparison of our single-view synthesis results with the baseline method [188] on cars (left) and chairs (right). Our prediction tends to be consistently better at preserving high-frequency details (e.g. texture and edge boundaries) than the baseline.

from −180 to +180 degrees at steps of 20 degrees, and encode the transformation as a 19-D one-hot vector.

At each training iteration, we randomly sample a batch of $< I_s, I_t, T >$ tuples from the training split for the single-view setting, and $< I_{s_1}, I_{s_2}, I_t, T_1, T_2 >$ tuples for the multi-view setting, where T_i denotes the relative viewpoint transformation between I_{s_i} and I_t, and T_i is randomly sampled from the set of valid transformations. For each category, we construct a test set of $20,000$ tuples by following the same sampling procedure above, except that the shapes are now sampled from the test split.

Appearance flows versus direct pixel generation Our first experiment compares the view synthesis performance of our appearance flow approach with the direct pixel generation method by [188] under the single input view setting.

Figure 4.3 compares the view synthesis results using different methods on examples from the test set of two categories (*car* and *chair*). Overall, our prediction tends to be much sharper and matches the ground-truth better than the baseline. In particular, our synthesized views using appearance flows are able to maintain detailed textures and edge boundaries that are lost in direct pixel generation despite both networks are trained with the same loss function.

For quantitative evaluation we measure the mean pixel L_1 error between the predicted views and the ground-truth on the foreground regions. As shown in Table **??**, our method outperforms the baseline in both categories (*car* and *chair*). We further analyze the error statistics by computing the pairwise cross-view confusion matrix for both methods, which

Input	Method	Car	Chair
Single-view	Tatarchenko *et al.* [188]	0.404	0.345
	Ours	**0.368**	**0.323**
Multi-view	Tatarchenko *et al.* [188]	0.385	0.334
	Ours	**0.285**	**0.248**

Table 4.1: Mean pixel L_1 error between the ground-truth and predictions by different methods. Lower is better.

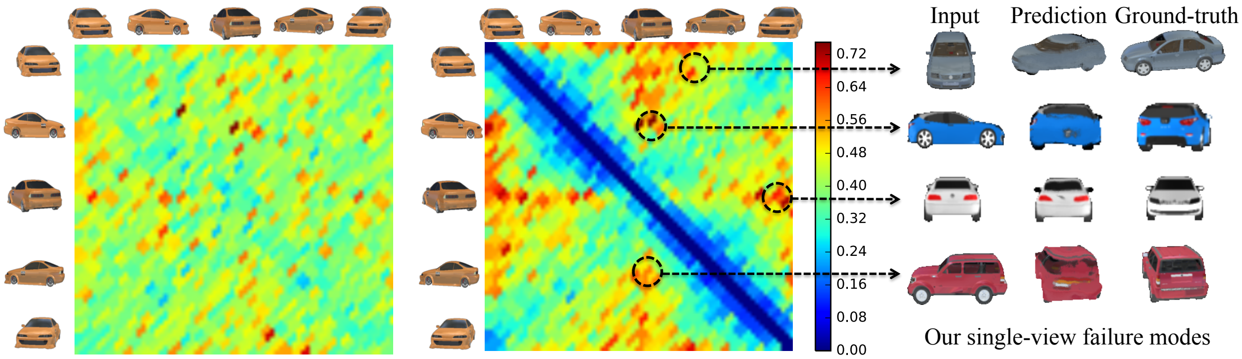


Figure 4.4: Visualization of error statistics for generating novel views from a single input view on the *car* category. The heatmaps (blue–low, red–high) depict the mean pixel error for obtaining the target view (columns) from the input view (rows) for the baseline [188] (left) and our approach(middle). Some common failure modes of our method are visualized on the right.

measures how predictive/informative a given view is for synthesizing another view (see the visualization in Figure 4.4). The error statistics suggest that our method is especially strong in synthesizing views that share significant number of common pixels with the input view (within ±45 degrees azimuth variation from the input view – the diagonals in the plot) or along the corresponding symmetry planes (off-diagonals) that typically exhibit high appearance correlation with the input view (e.g. synthesizing the right view from the left view of a car), and slightly weaker than direct pixel generation in views that do not share much in common (e.g. from frontal to the side or rear views).

Interestingly though, when we conduct perceptual studies comparing the visual similarity between predicted views and the ground-truth, our method is far ahead of the baseline across the entire spectrum of the cross-view predictions. More specifically, we randomly sampled $1,000$ test tuples, and asked users on Amazon Mechanical Turk to select the prediction that looks more similar to the ground-truth. We average the responses over 5 unique turkers for each test tuple, and find that 95% of the time our prediction is chosen over the baseline for cars and 93% for chairs, suggesting that the L_1 metric might not fully reflect the strength

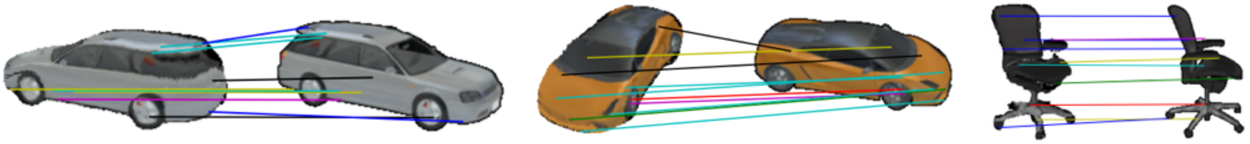


Figure 4.5: Sample appearance flow vectors predicted by our method. For randomly sampled points in the generated target image (left), the lines depict the corresponding appearance flow to the source image (right).

of our method.

One additional benefit of predicting appearance flows is that it allows intuitive visualization and understanding of exactly how the synthesized view is constructed. For instance, Figure 4.5 shows sample appearance flow vectors predicted by our method. It is interesting to note that the appearance flows do not necessarily correspond to anatomically/symmetrically corresponding parts. For example, while the top-right pixels of the first car in Figure 4.5 transfer appearance from their corresponding location in the source image, the pixels in the back wheel are generated using the front wheel of the source image.

Multi-view versus single-view In this experiment, we evaluate the synthesis performance of using multiple input views (two in this case). It turns out that having multiple input views is much more beneficial for our approach than for the baseline, as our synthesis error drops significantly compared to the single-view setting while less so for the baseline (see Table 4.1). This indicates that predicting appearance flows allows more effective utilization of different prediction hypotheses. Figure 4.6 shows sample visualization of how our multi-view synthesis network automatically combines high-quality predictions from individual input views to construct the final prediction.

Results on PASCAL VOC [42] images Although our synthesis network is trained on rendered synthetic images, it also exhibits potentials in generalizing to real images. In order to use our learnt models for synthesizing views for objects in PASCAL VOC, we require some pre-processing to ensure input statistics similar to the rendered training set. We therefore re-scale the input image to have similar number of foreground pixels as objects in the training set with the same aspect ratio. We visualize and compare a few example synthesis results on segmented PASCAL VOC images in Figure 4.7.

4.5 Discussion

We have presented a framework that re-parametrizes image synthesis as predicting the appearance flow field between the input image(s) and the output, and demonstrated its successful application to novel view synthesis. However, our method is by no means close to solving the problem in the general case. A number of major challenges are yet to be addressed:

- Our current method is incapable of hallucinating pixel values not present in the input

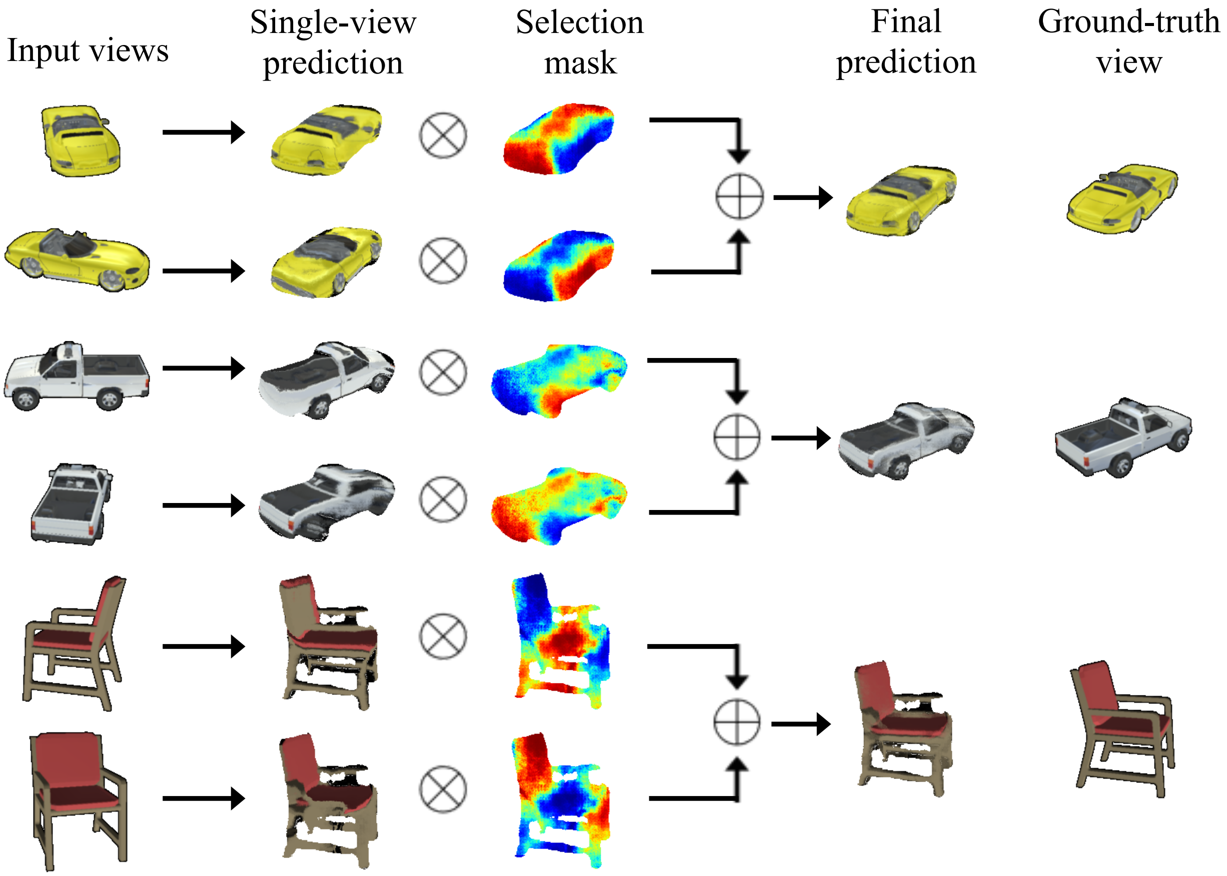


Figure 4.6: View synthesis examples using our multi-view network. Each input view makes independent prediction of a candidate target view as well as a selection/confidence mask (blue–low, red–high). The final prediction is obtained by linearly combining the single-view predictions with weights normalized across the selection masks. Typically, the final prediction is more similar to the ground-truth than any independent prediction.

view. While this is not as bad is it sounds (since the color palette of a typical image is quite rich), it would be beneficial to develop a mechanism that combines the hallucination capability of pixel generation CNN and the detail-preserving property of our flow-based synthesis.

- Empirically we observe that our network sometimes struggles in learning long-range appearance correlations, since the gradients derived from the flows are quite local. We conducted preliminary experiments with multi-scale reconstruction loss, and found it to alleviate the gradient locality to some extent.

- While the academic community around view synthesis is growing rapidly, we are still missing large-scale datasets of diverse real-world objects/scenes and a proper metric

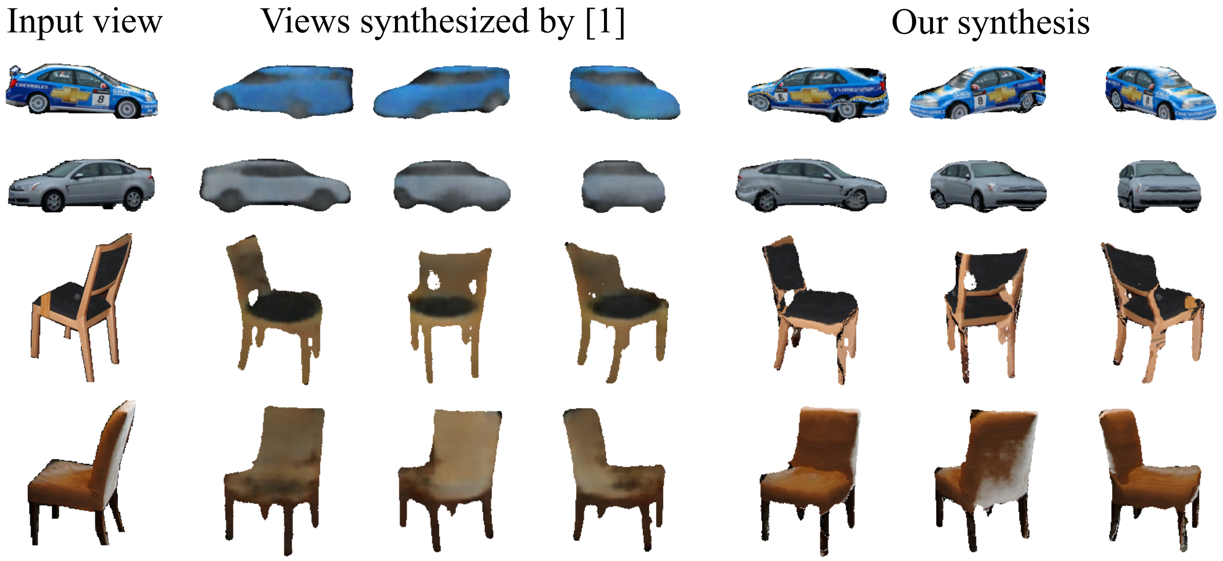


Figure 4.7: View synthesis results for segmented objects in the PASCAL VOC dataset. Our method generalizes better and yields more realistic results than the baseline [188].

(L_1 pixel error is certainly not ideal) for measuring research progress.

- All the existing learning-based view synthesis approaches assume knowing the category of the object. An interesting direction is to develop a method that is category-agnostic, and once learned, can be applied to any real-world image.

Finally, we believe that our technique of leveraging appearance flows is also applicable to tasks beyond novel view synthesis, including image inpainting, video frame prediction, modeling effect of actions, super-resolution, *etc.* Furthermore, notice that the explicit pixel associations provided by appearance flows naturally emerge without the need of any ground-truth labels. In the next two chapters, we show how to utilize this observation to learn scene geometry without labeled data.

Chapter 5

Learning Depth and Ego-Motion via View Synthesis

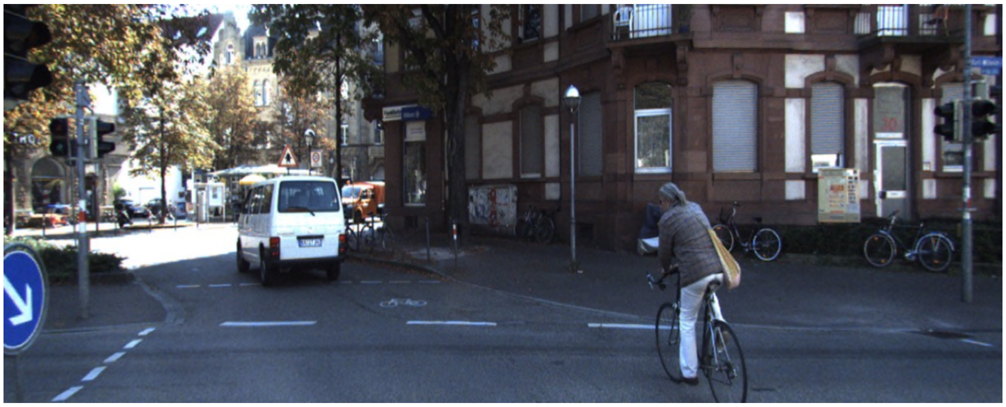


Figure 5.1: An example image from the KITTI dataset [57].

Humans are capable of perceiving rich 3D structure from a single 2D image. For instance, given the image in Figure 5.1, we could easily infer that the biker is closer to the camera than the minivan, and the minivan is closer than the tree in the background. We can also tell that the road is flat, and faces up towards the sky. We can even tell that the minivan has a cuboidal shape without looking at its hidden surfaces. This is remarkable because single-image 3D is an ambiguous task by itself. A 2D image could be the projection of an infinite number of different 3D entities. Therefore, we must rely on learning from our past visual experience to resolve the ambiguity.

This chapter presents a framework for learning monocular depth and camera motion estimation from unlabeled video sequences. By "unlabeled" we mean no ground-truth depth or pose labels are available for training. We formulate the learning objective around the observation that if both the depth and camera motion are predicted correctly, they should consistently explain the nearby frames through the task of novel view synthesis[1].

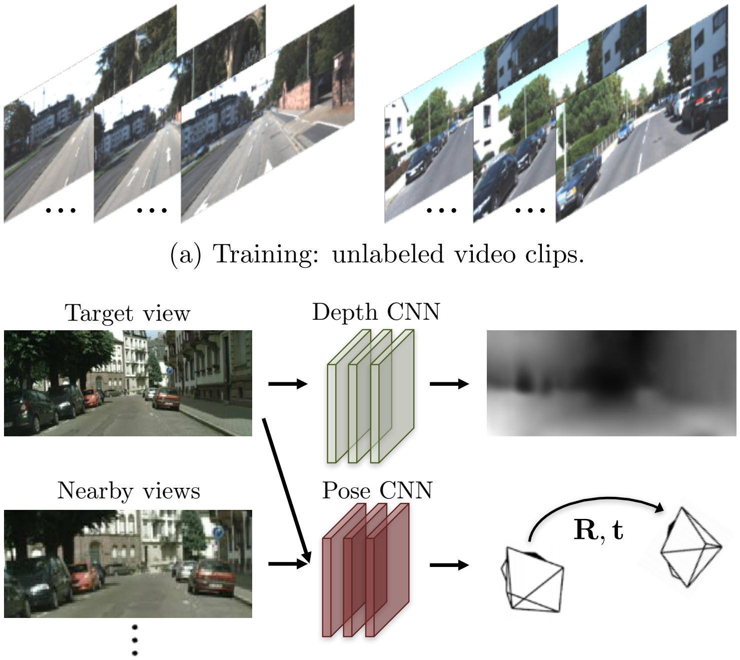
5.1 Introduction

Humans are remarkably capable of inferring ego-motion and the 3D structure of a scene even over short timescales. For instance, in navigating along a street, we can easily locate obstacles and react quickly to avoid them. Years of research in geometric computer vision has failed to recreate similar modeling capabilities for real-world scenes (e.g., where non-rigidity, occlusion and lack of texture are present). So why do humans excel at this task? One hypothesis is that we develop a rich, structural understanding of the world through our past visual experience that has largely consisted of moving around and observing vast numbers of scenes and developing *consistent* modeling of our observations. From millions of such observations, we have learned about the regularities of the world—roads are flat, buildings are straight, cars are supported by roads etc., and we can apply this knowledge when perceiving a new scene, even from a single monocular image.

In this chapter, we mimic this approach by training a model that observes sequences of images and aims to explain its observations by predicting likely camera motion and the scene structure (as shown in Fig. 5.2). We take an end-to-end approach in allowing the model to map directly from input pixels to an estimate of ego-motion (parameterized as 6-DoF transformation matrices) and the underlying scene structure (parameterized as per-pixel depth maps under a reference view). We are particularly inspired by prior work that has suggested view synthesis as a metric [185] and recent work that tackles the calibrated, multi-view 3D case in an end-to-end framework [47]. Our model is not supervised by ground-truth depth or camera motion, and can be trained simply using sequences of images with no manual labeling or even camera motion information.

Our approach builds upon the insight that a geometric view synthesis system only performs *consistently* well when its intermediate predictions of the scene geometry and the camera poses correspond to the physical ground-truth. While imperfect geometry and/or pose estimation can cheat with reasonable synthesized views for certain types of scenes (e.g., textureless), the same model would fail miserably when presented with another set of scenes with more diverse layout and appearance structures. Thus, our goal is to formulate the entire view synthesis pipeline as the inference procedure of a convolutional neural network, so that by training the network on large-scale video data for the 'meta'-task of view synthesis the network is forced to learn about intermediate tasks of depth and camera pose estimation in order to come up with a consistent explanation of the visual world.

[1] This work was originally published as *Unsupervised learning of depth and ego-motion from video.* In CVPR, 2017 [234].



(a) Training: unlabeled video clips.

(b) Testing: single-view depth and multi-view pose estimation.

Figure 5.2: The training data to our system consists solely of unlabeled image sequences capturing scene appearance from different viewpoints, where the poses of the images are not provided. Our training procedure produces two models that operate independently, one for single-view depth prediction, and one for multi-view camera pose estimation.

5.2 Background

Structure from motion The simultaneous estimation of structure and motion is a well studied problem with an established toolchain of techniques [53, 210, 152]. Whilst the traditional toolchain is effective and efficient in many cases, its reliance on accurate image correspondence can cause problems in areas of low texture, complex geometry/photometry, thin structures, and occlusions. To address these issues, several of the pipeline stages have been recently tackled using deep learning, e.g., feature matching [69], pose estimation [107], and stereo [47, 108, 224]. These learning-based techniques are attractive in that they are able to leverage external supervision during training, and potentially overcome the above issues when applied to test data.

Warping-based view synthesis One important application of geometric scene understanding is the task of novel view synthesis, where the goal is to synthesize the appearance of the scene seen from novel camera viewpoints. A classic paradigm for view synthesis is to first either estimate the underlying 3D geometry explicitly or establish pixel correspondence among input views, and then synthesize the novel views by compositing image patches from the input views (e.g., [22, 239, 175, 32, 46]). Recently, end-to-end learning has been applied to reconstruct novel views by transforming the input based on depth or flow, e.g., DeepStereo [47], Deep3D [213] and Appearance Flows [235]. In these methods, the underlying geometry is represented by quantized depth planes (DeepStereo), probabilistic disparity maps (Deep3D) and view-dependent flow fields (Appearance Flows), respectively. Unlike methods that directly map from input views to the target view (e.g., [187]), warping-based methods are forced to learn intermediate predictions of geometry and/or correspondence. In this work, we aim to distill such geometric reasoning capability from CNNs trained to perform warping-based view synthesis.

Learning single-view 3D from registered 2D views Our work is closely related to a line of recent research on learning single-view 3D inference from registered 2D observations. Garg *et al.* [55] propose to learn a single-view depth estimation CNN using projection errors to a calibrated stereo twin for supervision. Concurrently, Deep3D [213] predicts a second stereo viewpoint from an input image using stereoscopic film footage as training data. A similar approach was taken by Godard *et al.* [59], with the addition of a left-right consistency constraint, and a better architecture design that led to impressive performance. Like our approach, these techniques only learn from image observations of the world, unlike methods that require explicit depth for training, e.g., [80, 173, 40, 108, 118].

These techniques bear some resemblance to direct methods for structure and motion estimation [90], where the camera parameters and scene depth are adjusted to minimize a pixel-based error function. However, rather than directly minimizing the error to obtain the estimation, the CNN-based methods only take a gradient step for each batch of input instances, which allows the network to learn an implicit prior from a large corpus of related imagery. Several authors have explored building differentiable rendering operations into their models that are trained in this way, e.g., [70, 117, 137].

While most of the above techniques (including ours) are mainly focused on inferring depth maps as the scene geometry output, recent work (e.g., [54, 166, 191, 215]) has also shown success in learning 3D volumetric representations from 2D observations based on similar principles of projective geometry. Fouhey *et al.* [50] further show that it is even possible to learn 3D inference without 3D labels (or registered 2D views) by utilizing scene regularity.

Unsupervised/Self-supervised learning from video Another line of related work to ours is visual representation learning from video, where the general goal is to design pretext tasks for learning generic visual features from video data that can later be re-purposed for other vision tasks such as object detection and semantic segmentation. Such pretext tasks include ego-motion estimation [4, 95], tracking [202], temporal coherence [63], temporal order verification [148], and object motion mask prediction [155]. While we focus on inferring the explicit scene geometry and ego-motion in this work, intuitively, the internal representation learned by the deep network (especially the single-view depth CNN) should capture some level of semantics that could generalize to other tasks as well.

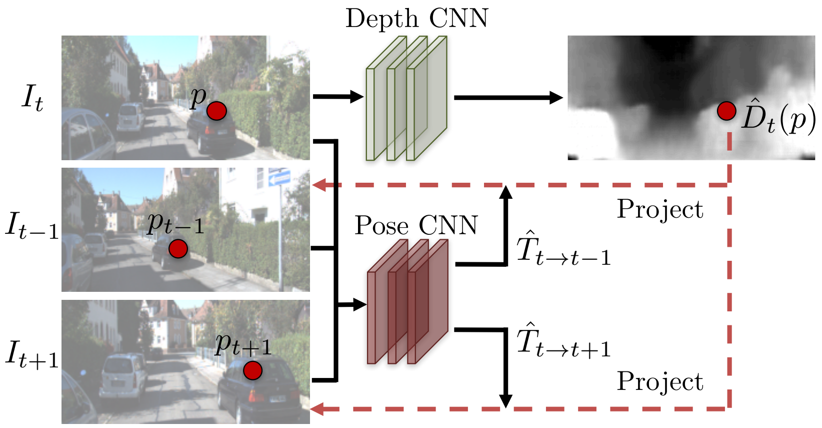


Figure 5.3: Overview of the supervision pipeline based on view synthesis. The depth network takes only the target view as input, and outputs a per-pixel depth map $\hat{D}_t$. The pose network takes both the target view (I_t) and the nearby/source views (e.g., I_{t-1} and I_{t+1}) as input, and outputs the relative camera poses ($\hat{T}_{t\to t-1}, \hat{T}_{t\to t+1}$). The outputs of both networks are then used to inverse warp the source views (see Sec. 5.3.2) to reconstruct the target view, and the photometric reconstruction loss is used for training the CNNs. By utilizing view synthesis as supervision, we are able to train the entire framework in an unsupervised manner from videos.

Concurrent to our work, Vijayanarasimhan *et al.* [197] independently propose a framework for joint training of depth, camera motion and scene motion from videos. While both methods are conceptually similar, ours is focused on the unsupervised aspect, whereas their framework adds the capability to incorporate supervision (e.g., depth, camera motion or scene motion). There are significant differences in how scene dynamics are modeled during training, in which they explicitly solve for object motion whereas our explainability mask discounts regions undergoing motion, occlusion and other factors.

5.3 Approach

Here we describe a framework for jointly training a single-view depth CNN and a camera pose estimation CNN from unlabeled video sequences. Despite being jointly trained, the depth model and the pose estimation model can be used independently during test-time inference. Training examples to our model consist of short image sequences of scenes captured by a moving camera. While our training procedure is robust to some degree of scene motion, we assume that the scenes we are interested in are mostly rigid, i.e., the scene appearance change across different frames is dominated by the camera motion.

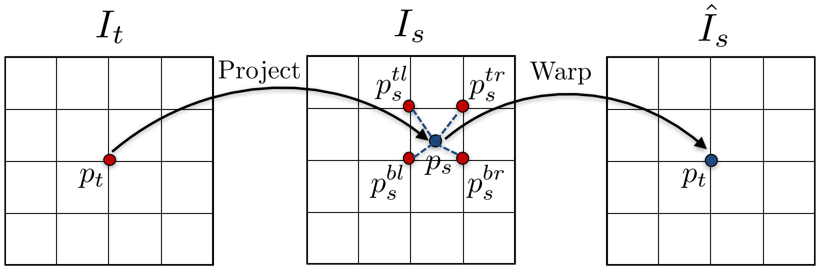


Figure 5.4: Illustration of the differentiable image warping process. For each point p_t in the target view, we first project it onto the source view based on the predicted depth and camera pose, and then use bilinear interpolation to obtain the value of the warped image $\hat{I}_s$ at location p_t.

5.3.1 View synthesis as supervision

The key supervision signal for our depth and pose prediction CNNs comes from the task of *novel view synthesis*: given one input view of a scene, synthesize a new image of the scene seen from a different camera pose. We can synthesize a target view given a per-pixel depth in that image, plus the pose and visibility in a nearby view. As we will show next, this synthesis process can be implemented in a fully differentiable manner with CNNs as the geometry and pose estimation modules. Visibility can be handled, along with non-rigidity and other non-modeled factors, using an "explanability" mask, which we discuss later (Sec. 5.3.3).

Let us denote $< I_1, \ldots, I_N >$ as a training image sequence with one of the frames I_t being the target view and the rest being the source views $I_s (1 \leq s \leq N, s \neq t)$. The view synthesis objective can be formulated as

$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)| \ , \tag{5.1}$$

where p indexes over pixel coordinates, and $\hat{I}_s$ is the source view I_s warped to the target coordinate frame based on a depth image-based rendering module [44] (described in Sec. 5.3.2), taking the predicted depth $\hat{D}_t$, the predicted 4×4 camera transformation matrix[2] $\hat{T}_{t \to s}$ and the source view I_s as input.

Note that the idea of view synthesis as supervision has also been recently explored for learning single-view depth estimation [55, 59] and multi-view stereo [47]. However, to the best of our knowledge, all previous work requires posed image sets during training (and testing too in the case of DeepStereo), while our framework can be applied to standard videos without pose information. Furthermore, it predicts the poses as part of the learning framework. See Figure 5.3 for an illustration of our learning pipeline for depth and pose estimation.

[2] In practice, the CNN estimates the Euler angles and the 3D translation vector, which are then converted to the transformation matrix.

5.3.2 Differentiable depth image-based rendering

As indicated in Eq. 5.1, a key component of our learning framework is a differentiable depth image-based renderer that reconstructs the target view I_t by sampling pixels from a source view I_s based on the predicted depth map $\hat{D}_t$ and the relative pose $\hat{T}_{t\to s}$.

Let p_t denote the homogeneous coordinates of a pixel in the target view, and K denote the camera intrinsics matrix. We can obtain p_t's projected coordinates onto the source view p_s by[3]

$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t \tag{5.2}$$

Notice that the projected coordinates p_s are continuous values. To obtain $I_s(p_s)$ for populating the value of $\hat{I}_s(p_t)$ (see Figure 5.4), we then use the differentiable bilinear sampling mechanism proposed in the *spatial transformer networks* [92] that linearly interpolates the values of the 4-pixel neighbors (top-left, top-right, bottom-left, and bottom-right) of p_s to approximate $I_s(p_s)$, i.e. $\hat{I}_s(p_t) = I_s(p_s) = \sum_{i\in\{t,b\},j\in\{l,r\}} w^{ij} I_s(p_s^{ij})$, where w^{ij} is linearly proportional to the spatial proximity between p_s and p_s^{ij} , and $\sum_{i,j} w^{ij} = 1$. A similar strategy is used in [235] for learning to directly warp between different views, while here the coordinates for pixel warping are obtained through projective geometry that enables the factorization of depth and camera pose.

5.3.3 Modeling the model limitation

Note that when applied to monocular videos the above view synthesis formulation implicitly assumes 1) the scene is static without moving objects; 2) there is no occlusion/disocclusion between the target view and the source views; 3) the surface is Lambertian so that the photo-consistency error is meaningful. If any of these assumptions are violated in a training sequence, the gradients could be corrupted and potentially inhibit training. To improve the robustness of our learning pipeline to these factors, we additionally train a *explainability prediction* network (jointly and simultaneously with the depth and pose networks) that outputs a per-pixel soft mask $\hat{E}_s$ for each target-source pair, indicating the network's belief in where direct view synthesis will be successfully modeled for each target pixel. Based on the predicted $\hat{E}_s$, the view synthesis objective is weighted correspondingly by

$$\mathcal{L}_{vs} = \sum_{<I_1,\ldots,I_N>\in\mathcal{S}} \sum_p \hat{E}_s(p)|I_t(p) - \hat{I}_s(p)| \ . \tag{5.3}$$

Since we do not have direct supervision for $\hat{E}_s$, training with the above loss would result in a trivial solution of the network always predicting $\hat{E}_s$ to be zero, which perfectly minimizes the loss. To resolve this, we add a regularization term $\mathcal{L}_{reg}(\hat{E}_s)$ that encourages nonzero predictions by minimizing the cross-entropy loss with constant label 1 at each pixel location. In other words, the network is encouraged to minimize the view synthesis objective, but allowed a certain amount of slack for discounting the factors not considered by the model.

[3] For notation simplicity, we omit showing the necessary conversion to homogeneous coordinates along the steps of matrix multiplication.

5.3.4 Overcoming the gradient locality

One remaining issue with the above learning pipeline is that the gradients are mainly derived from the pixel intensity difference between $I(p_t)$ and the four neighbors of $I(p_s)$, which would inhibit training if the correct p_s (projected using the ground-truth depth and pose) is located in a low-texture region or far from the current estimation. This is a well known issue in motion estimation [11]. Empirically, we found two strategies to be effective for overcoming this issue: 1) using a convolutional encoder-decoder architecture with a small bottleneck for the depth network that implicitly constrains the output to be globally smooth and facilitates gradients to propagate from meaningful regions to nearby regions; 2) explicit multi-scale and smoothness loss (e.g., as in [55, 59]) that allows gradients to be derived from larger spatial regions directly. We adopt the second strategy in this work as it is less sensitive to architectural choices. For smoothness, we minimize the L_1 norm of the second-order gradients for the predicted depth maps (similar to [197]).

Our final objective becomes

$$\mathcal{L}_{final} = \sum_l \mathcal{L}_{vs}^l + \lambda_s \mathcal{L}_{smooth}^l + \lambda_e \sum_s \mathcal{L}_{reg}(\hat{E}_s^l) \ , \tag{5.4}$$

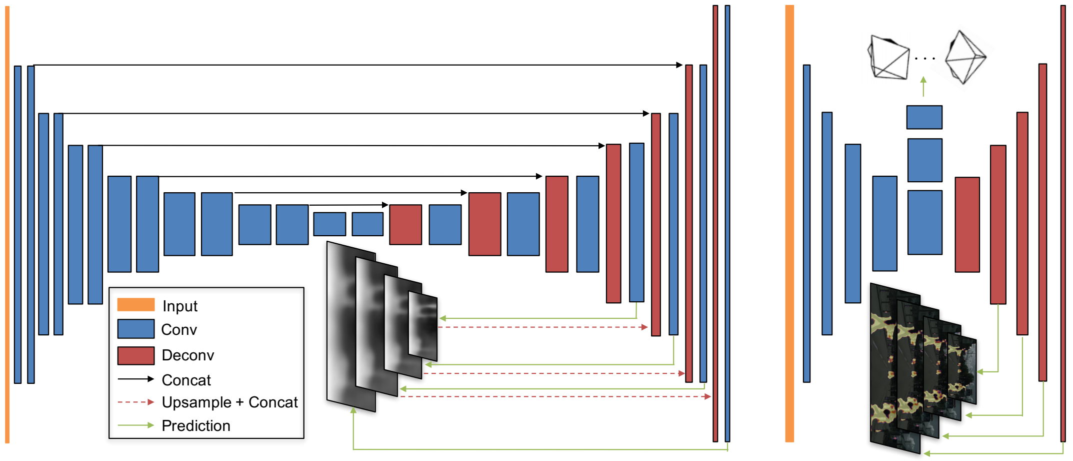
where l indexes over different image scales, s indexes over source images, and λ_s and λ_e are the weighting for the depth smoothness loss and the explainability regularization, respectively.

5.3.5 Network architecture

Single-view depth For single-view depth prediction, we adopt the DispNet architecture proposed in [146] that is mainly based on an encoder-decoder design with skip connections and multi-scale side predictions (see Figure 5.5). All conv layers are followed by ReLU activation except for the prediction layers, where we use $1/(\alpha * sigmoid(x) + \beta)$ with $\alpha = 10$ and $\beta = 0.1$ to constrain the predicted depth to be always positive within a reasonable range. We also experimented with using multiple views as input to the depth network, but did not find this to improve the results. This is in line with the observations in [194], where optical flow constraints need to be enforced to utilize multiple views effectively.

Pose The input to the pose estimation network is the target view concatenated with all the source views (along the color channels), and the outputs are the relative poses between the target view and each of the source views. The network consists of 7 stride-2 convolutions followed by a 1×1 convolution with $6 * (N - 1)$ output channels (corresponding to 3 Euler angles and 3-D translation for each source view). Finally, global average pooling is applied to aggregate predictions at all spatial locations. All conv layers are followed by ReLU except for the last layer where no nonlinear activation is applied.

Explainability mask The explainability prediction network shares the first five feature encoding layers with the pose network, followed by 5 deconvolution layers with multi-scale



(a) Single-view depth network

(b) Pose/explainability network

Figure 5.5: Network architecture for our depth/pose/explainability prediction modules. The width and height of each rectangular block indicates the output channels and the spatial dimension of the feature map at the corresponding layer respectively, and each reduction/increase in size indicates a change by the factor of 2. (a) For single-view depth, we adopt the DispNet [146] architecture with multi-scale side predictions. The kernel size is 3 for all the layers except for the first 4 conv layers with $7, 7, 5, 5$, respectively. The number of output channels for the first conv layer is 32. (b) The pose and explainabilty networks share the first few conv layers, and then branch out to predict 6-DoF relative pose and multi-scale explainability masks, respectively. The number of output channels for the first conv layer is 16, and the kernel size is 3 for all the layers except for the first two conv and the last two deconv/prediction layers where we use $7, 5, 5, 7$, respectively. See Section 5.3.5 for more details.

side predictions. All conv/deconv layers are followed by ReLU except for the prediction layers with no nonlinear activation. The number of output channels for each prediction layer is $2 * (N - 1)$, with every two channels normalized by *softmax* to obtain the explainability prediction for the corresponding source-target pair (the second channel after normalization is $\hat{E}_s$ and used in computing the loss in Eq. 5.3).

5.4 Experiments

Here we evaluate the performance of our system, and compare with prior approaches on single-view depth as well as ego-motion estimation. We mainly use the KITTI dataset [57] for benchmarking, but also use the Make3D dataset [173] for evaluating cross-dataset generalization ability.

Training Details We implemented the system using the publicly available TensorFlow [1] framework. For all the experiments, we set $\lambda_s = 0.5/l$ (l is the downscaling factor for the corresponding scale) and $\lambda_e = 0.2$. During training, we used batch normalization [89] for all the layers except for the output layers, and the Adam [114] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate of 0.0002 and mini-batch size of 4. The training typically converges after about $150K$ iterations. All the experiments are performed with image sequences captured with a monocular camera. We resize the images to 128×416 during training, but both the depth and pose networks can be run fully-convolutionally for images of arbitrary size at test time.

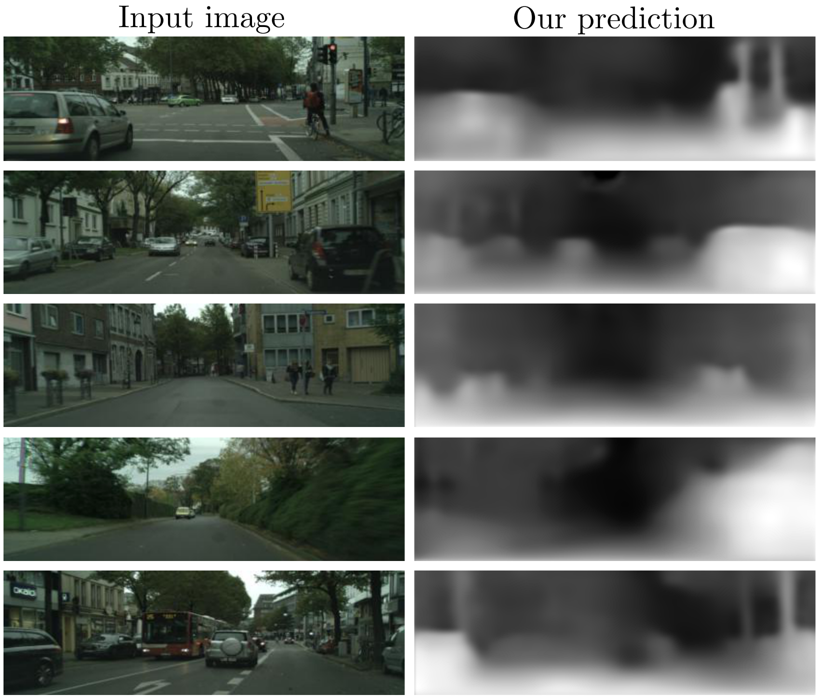


Figure 5.6: Our sample predictions on the Cityscapes dataset using the model trained on Cityscapes only.

5.4.1 Single-view depth estimation

We train our system on the split provided by [40], and exclude all the frames from the testing scenes as well as static sequences with mean optical flow magnitude less than 1 pixel for training. We fix the length of image sequences to be 3 frames, and treat the central frame as the target view and the ±1 frames as the source views. We use images captured by both color cameras, but treated them independently when forming training sequences. This results in a total of $44,540$ sequences, out of which we use $40,109$ for training and $4,431$ for validation.

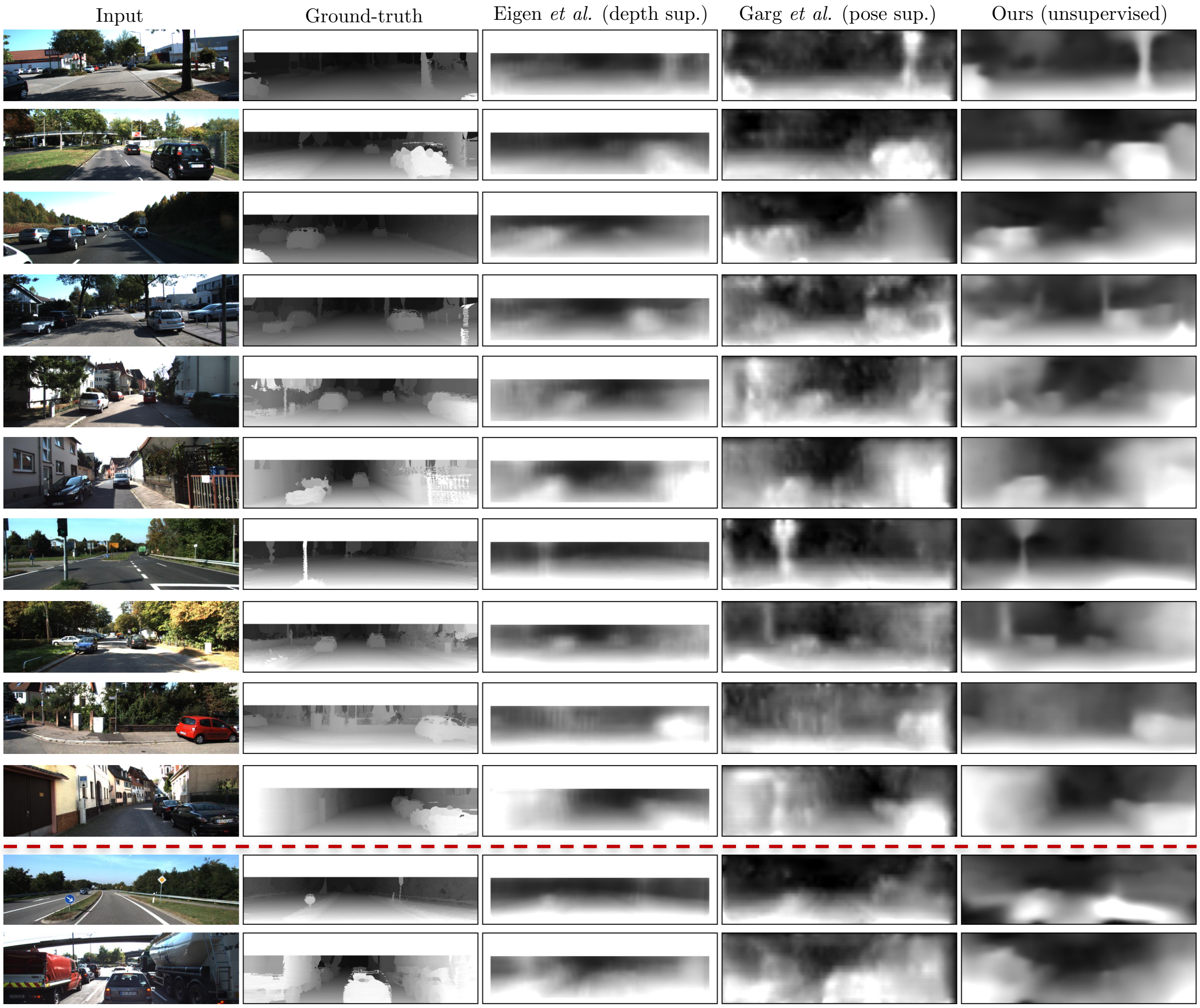


Figure 5.7: Comparison of single-view depth estimation between Eigen *et al.* [40] (with ground-truth depth supervision), Garg *et al.* [55] (with ground-truth pose supervision), and ours (unsupervised). The ground-truth depth map is interpolated from sparse measurements for visualization purpose. The last two rows show typical failure cases of our model, which sometimes struggles in vast open scenes and objects close to the front of the camera.

To the best of our knowledge, no previous systems exist that learn single-view depth estimation in an unsupervised manner from monocular videos. Nonetheless, here we provide comparison with prior methods with depth supervision [40] and recent methods that use calibrated stereo images (i.e. with pose supervision) for training [55, 59]. Since the depth predicted by our method is defined up to a scale factor, for evaluation we multiply the predicted depth maps by a scalar $\hat{s}$ that matches the median with the ground-truth, i.e. $\hat{s} = median(D_{gt})/median(D_{pred})$.

Similar to [59], we also experimented with first pre-training the system on the larger

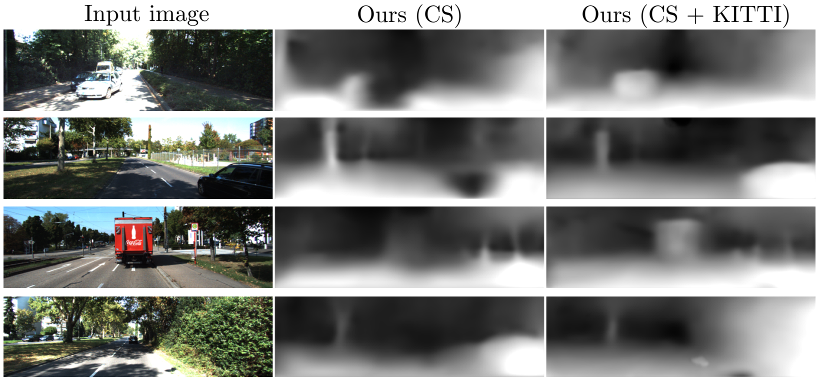


Figure 5.8: Comparison of single-view depth predictions on the KITTI dataset by our initial Cityscapes model and the final model (pre-trained on Cityscapes and then fine-tuned on KITTI). The Cityscapes model sometimes makes structural mistakes (e.g. holes on car body) likely due to the domain gap between the two datasets.

Cityscapes dataset [29] (sample predictions are shown in Figure 5.6), and then fine-tune on KITTI, which results in slight performance improvement.

Method	Dataset	Supervision		Error metric				Accuracy metric		
		Depth	Pose	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Train set mean	K	✓		0.403	5.530	8.709	0.403	0.593	0.776	0.878
Eigen *et al.* [40] Coarse	K	✓		0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen *et al.* [40] Fine	K	✓		0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu *et al.* [130]	K	✓		0.202	1.614	6.523	0.275	0.678	0.895	0.965
Godard *et al.* [59]	K		✓	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Godard *et al.* [59]	CS + K		✓	0.124	1.076	5.311	0.219	0.847	0.942	0.973
Ours (w/o explainability)	K			0.221	2.226	7.527	0.294	0.676	0.885	0.954
Ours	K			0.208	1.768	6.856	0.283	0.678	0.885	0.957
Ours	CS			0.267	2.686	7.580	0.334	0.577	0.840	0.937
Ours	CS + K			0.198	1.836	6.565	0.275	0.718	0.901	0.960
Garg *et al.* [**garg**] cap 50m	K		✓	0.169	1.080	5.104	0.273	0.740	0.904	0.962
Ours (w/o explainability) cap 50m	K			0.208	1.551	5.452	0.273	0.695	0.900	0.964
Ours cap 50m	K			0.201	1.391	5.181	0.264	0.696	0.900	0.966
Ours cap 50m	CS			0.260	2.232	6.148	0.321	0.590	0.852	0.945
Ours cap 50m	CS + K			0.190	1.436	4.975	0.258	0.735	0.915	0.968

Table 5.1: Single-view depth results on the KITTI dataset [57] using the split of Eigen *et al.* [40] (Baseline numbers taken from [59]). For training, K = KITTI, and CS = Cityscapes [29]. All methods we compare with use some form of supervision (either ground-truth depth or calibrated camera pose) during training. Note: results from Garg et al. [55] are capped at 50m depth, so we break these out separately in the lower part of the table.

KITTI Here we evaluate the single-view depth performance on the 697 images from the test split of [40]. As shown in Table 5.1, our unsupervised method performs comparably with several supervised methods (e.g. Eigen *et al.* [40] and Garg *et al.* [55]), but falls short of concurrent work by Godard *et al.* [59] that uses calibrated stereo images (i.e. with pose supervision) with left-right cycle consistency loss for training. For future work, it would be interesting to see if incorporating the similar cycle consistency loss into our framework could further improve the results. Figure 5.7 provides examples of visual comparison between our results and some supervised baselines over a variety of examples. One can see that although trained in an unsupervised manner, our results are comparable to that of the supervised baselines, and sometimes preserve the depth boundaries and thin structures such as trees and street lights better.

We show sample predictions made by our initial Cityscapes model and the final model (pre-trained on Cityscapes and then fine-tuned on KITTI) in Figure 5.8. Due to the domain gap between the two datasets, our Cityscapes model sometimes has difficulty in recovering the complete shape of the car/bushes, and mistakes them with distant objects.

We also performed an ablation study of the explainability modeling (see Table 5.1), which turns out only offering a modest performance boost. This is likely because 1) most of the KITTI scenes are static without significant scene motions, and 2) the occlusion/visibility effects only occur in small regions in sequences across a short time span (3-frames), which make the explainability modeling less essential to the success of training. Nonetheless, our explainability prediction network does seem to capture the factors like scene motion and visibility well (see Sec. 5.4.3), and could potentially be more important for other more challenging datasets.

Make3D To evaluate the generalization ability of our single-view depth model, we directly apply our model trained on Cityscapes + KITTI to the Make3D dataset unseen during training. While there still remains a significant performance gap between our method and others supervised using Make3D ground-truth depth (see Table 5.2), our predictions are able to capture the global scene layout reasonably well without any training on the Make3D images (see Figure 5.9).

Method	Supervision		Error metric			
	Depth	Pose	Abs Rel	Sq Rel	RMSE	RMSE log
Train set mean	✓		0.876	13.98	12.27	0.307
Karsch *et al.* [102]	✓		0.428	5.079	8.389	0.149
Liu *et al.* [131]	✓		0.475	6.562	10.05	0.165
Laina *et al.* [120]	✓		0.204	1.840	5.683	0.084
Godard *et al.* [59]		✓	0.544	10.94	11.76	0.193
Ours			0.383	5.321	10.47	0.478

Table 5.2: Results on the Make3D dataset [173]. Similar to ours, Godard *et al.* [59] do not utilize any of the Make3D data during training, and directly apply the model trained on KITTI+Cityscapes to the test set. Following the evaluation protocol of [59], the errors are only computed where depth is less than 70 meters in a central image crop.

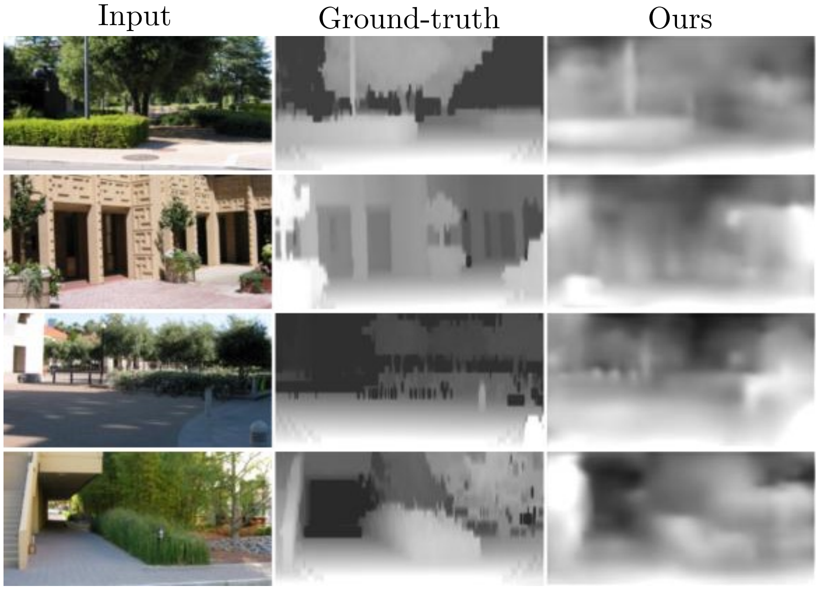


Figure 5.9: Our sample predictions on the Make3D dataset. Note that our model is trained on KITTI + Cityscapes only, and directly tested on Make3D.

5.4.2 Pose estimation

To evaluate the performance of our pose estimation network, we applied our system to the official KITTI odometry split (containing 11 driving sequences with ground truth odometry obtained through the IMU/GPS readings, which we use for evaluation purpose only), and used sequences 00-08 for training and 09-10 for testing. In this experiment, we fix the length of input image sequences to our system to 5 frames. We compare our ego-motion estimation with two variants of monocular ORB-SLAM [150] (a well-established SLAM system): 1) `ORB-SLAM (full)`, which recovers odometry using all frames of the driving sequence (i.e. allowing loop closure and re-localization), and 2) `ORB-SLAM (short)`, which runs on 5-frame snippets (same as our input setting). Another baseline we compare with is the dataset mean of car motion (using ground-truth odometry) for 5-frame snippets. To resolve scale ambiguity during evaluation, we first optimize the scaling factor for the predictions made by each method to best align with the ground truth, and then measure the Absolute Trajectory Error (ATE) [150] as the metric. ATE is computed on 5-frame snippets and averaged over the full sequence.[4] As shown in Table 5.3 and Fig. 5.10, our method outperforms both baselines (mean odometry and `ORB-SLAM (short)`) that share the same input setting as ours, but falls short of `ORB-SLAM (full)`, which leverages whole sequences (1591 for seq. 09 and 1201 for seq. 10) for loop closure and re-localization.

For better understanding of our pose estimation results, we show in Figure 5.10 the ATE curve with varying amount of side-rotation by the car between the beginning and the end

[4] For evaluating `ORB-SLAM (full)` we break down the trajectory of the full sequence into 5-frame snippets with the reference coordinate frame adjusted to the central frame of each snippet.

of a sequence. Figure 5.10 suggests that our method is significantly better than `ORB-SLAM (short)` when the side-rotation is small (i.e. car mostly driving forward), and comparable to `ORB-SLAM (full)` across the entire spectrum. The large performance gap between ours and `ORB-SLAM (short)` suggests that our learned ego-motion could potentially be used as an alternative to the local estimation modules in monocular SLAM systems.

Method	**Seq.** 09	**Seq.** 10
ORB-SLAM (full)	$\mathbf{0.014 \pm 0.008}$	$\mathbf{0.012 \pm 0.011}$
ORB-SLAM (short)	0.064 ± 0.141	0.064 ± 0.130
Mean Odom.	0.032 ± 0.026	0.028 ± 0.023
Ours	$\mathbf{0.021 \pm 0.017}$	$\mathbf{0.020 \pm 0.015}$

Table 5.3: Absolute Trajectory Error (ATE) on the KITTI odometry split averaged over all 5-frame snippets (lower is better). Our method outperforms baselines with the same input setting, but falls short of `ORB-SLAM (full)` that uses strictly more data.

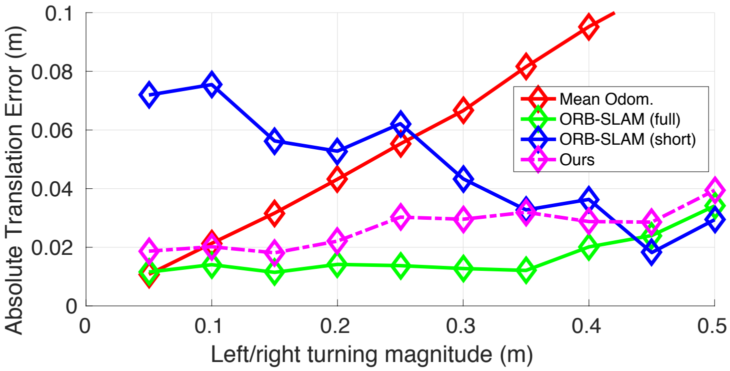


Figure 5.10: Absolute Trajectory Error (ATE) at different left/right turning magnitude (coordinate difference in the side-direction between the start and ending frame of a testing sequence). Our method performs significantly better than `ORB-SLAM (short)` when side rotation is small, and is comparable with `ORB-SLAM (full)` across the entire spectrum.

5.4.3 Visualizing the explainability prediction

We visualize example explainability masks predicted by our network in Figure 5.11. The first three rows suggest that the network has learned to identify dynamic objects in the scene as unexplainable by our model, and similarly, rows 4–5 are examples of objects that disappear from the frame in subsequent views. The last two rows demonstrate the potential downside of explainability-weighted loss: the depth CNN has low confidence in predicting thin structures well, and tends to mask them as unexplainable.

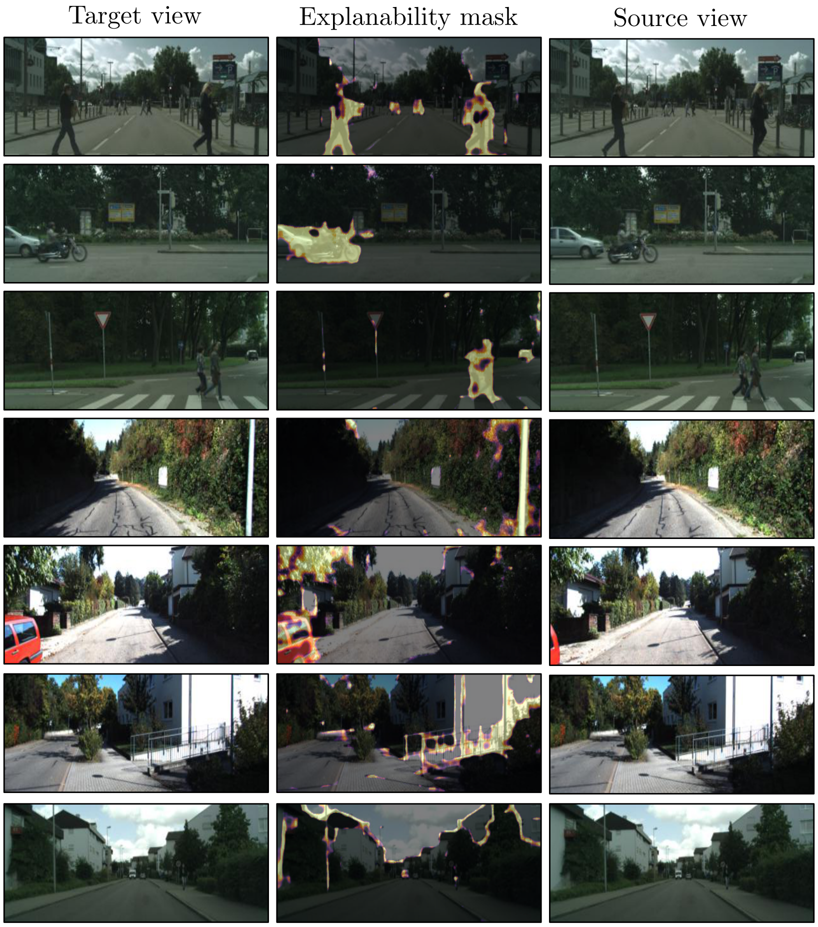


Figure 5.11: Sample visualizations of the explainability masks. Highlighted pixels are predicted to be unexplainable by the network due to motion (rows 1–3), occlusion/visibility (rows 4–5), or other factors (rows 7–8).

5.5 Discussion

In this chapter, we have presented an end-to-end learning pipeline that utilizes the task of view synthesis for supervision of single-view depth and camera pose estimation. The system is trained on unlabeled videos, and yet performs comparably with approaches that require ground-truth depth or pose for training. Despite good performance on the benchmark evaluation, our method is by no means close to solving the general problem of unsupervised learning of 3D scene structure inference. A number of major challenges are yet to be addressed: 1) our current framework does not explicitly estimate scene dynamics and occlusions (although they are implicitly taken into account by the explainability masks), both of which are critical factors in 3D scene understanding. Direct modeling of scene dynamics through motion segmentation (e.g. [197, 161]) could be a potential solution; 2) our framework assumes the camera intrinsics are given, which forbids the use of random Internet videos with unknown camera types/calibration – we plan to address this in future work; 3) depth maps are a simplified representation of the underlying 3D scene. It would be interesting to extend our framework to learn full 3D volumetric representations (e.g., in the manner of [191]).

Another interesting area for future work would be to investigate in more detail the representation learned by our system. In particular, the pose network likely uses some form of image correspondence in estimating the camera motion, whereas the depth estimation network likely recognizes common structural features of scenes and objects. It would be interesting to probe these, and investigate the extent to which our network already performs, or could be re-purposed to perform, tasks such as object detection and semantic segmentation.

Chapter 6

Learning Multiplane Images via View Synthesis

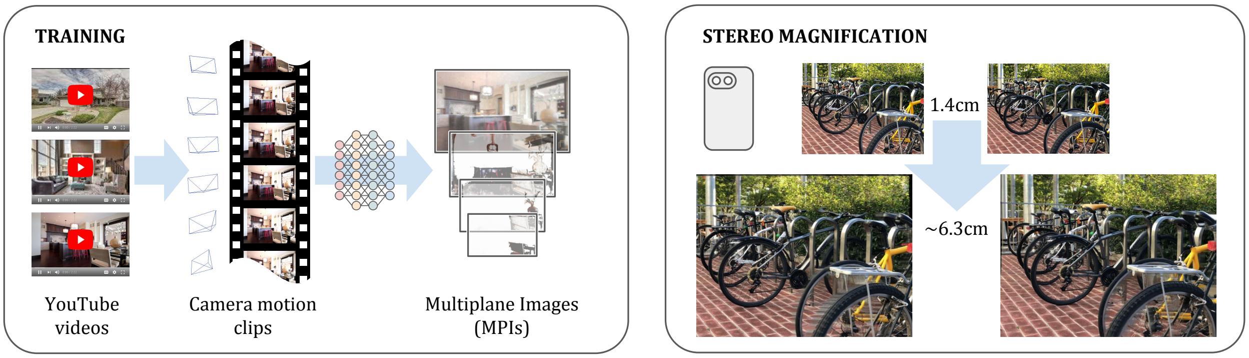


Figure 6.1: We extract camera motion clips from YouTube videos and use them to train a neural network to generate the Multiplane Image (MPI) scene representation from narrow-baseline stereo image pairs. The inferred MPI representation can then be used to synthesize novel views of the scene, including ones that extrapolate significantly beyond the input baseline. (Video stills in this and other figures in this chapter are used under Creative-Commons license from YouTube user *SonaVisual*.)

In the previous chapter, we used depth maps as the geometric scene representation. This chapter shifts focus to a different representation – *multiplane images* – that is capable of capturing richer information about the scene in both geometry and appearance than depth maps. A multiplane image (MPI) consists of a set of fronto-parallel planes at fixed depths from a reference camera coordinate frame, where each plane encodes an RGB image and an alpha map that capture the scene appearance at the corresponding depth. The MPI representation can be used for efficient and realistic rendering of novel views of the scene.

While MPI can be used as a general scene representation for view synthesis, this chapter explores an intriguing scenario: extrapolating views from imagery captured by narrow-baseline stereo cameras, including VR cameras and now-widespread dual-lens camera phones.

We call this problem *stereo magnification*. We describe a framework (see Figure 6.1) that uses a massive new data source for learning view extrapolation: online videos on YouTube. Using data mined from such videos, we train a deep network that predicts an MPI from an input stereo image pair. This inferred MPI can then be used to synthesize a range of novel views of the scene, including views that extrapolate significantly beyond the input baseline. We show that our method compares favorably with several recent view synthesis methods, and demonstrate applications in magnifying narrow-baseline stereo images [1].

6.1 Introduction

Photography has undergone an upheaval over the past decade. Cellphone cameras have steadily displaced point-and-shoot cameras, and have become competitive with digital SLRs in certain scenarios. This change has been driven by the increasing image quality of cellphone cameras, through better hardware and also through computational photography functionality such as high dynamic range imaging [72] and synthetic defocus [5, 62]. Many of these recent innovations have sought to replicate capabilities of traditional cameras. However, cell phones are also rapidly acquiring new kinds of sensors, such as multiple lenses and depth sensors, enabling applications beyond traditional photography.

In particular, dual-lens cameras are becoming increasingly common. While stereo cameras have been around for nearly as long as photography itself, recently a number of dual-camera phones, such as the iPhone 7, have appeared on the market. These cameras tend to have a very small baseline (distance between views) on the order of a centimeter. We have also seen the recent appearance of a number of “virtual-reality ready” cameras that capture stereo images and video from a pair of cameras spaced approximately eye-distance apart [61].

Motivated by the proliferation of stereo cameras, we explore the problem of synthesizing new views from such narrow-baseline image pairs. While much prior work has explored the problem of *interpolating* between a set of given views [22], we focus on the problem of *extrapolating* views significantly beyond the two input images. Such view extrapolation has many applications for photography. For instance, we might wish to take a narrow-baseline (∼1cm) stereo pair on a cell phone and extrapolate to an IPD-separated (∼6.3cm) stereo pair so as to create a photo with a compelling 3D stereo effect. Or, we might wish to take an IPD-separated stereo pair captured with a VR180 camera and extrapolate to an entire set of views along a line say half a meter in length, so as to enable full parallax with a small range of head motion. We call such view extrapolation from pairs of input views *stereo magnification*. The examples above involve magnifying the baseline by a significant amount—up to about 8x the original baseline.

The stereo magnification problem is challenging. We have just two views as input, unlike in common view interpolation scenarios that consider multiple views. We wish to be able to

[1] This work was originally published as *Stereo Magnification: Learning view synthesis using multiplane images.* In SIGGRAPH, 2018 [233].

handle challenging scenes with reflection and transparency. Finally, we need the capacity to render pixels that are occluded and thus not visible in either input view. To address these challenges, our approach is to *learn* to perform view extrapolation from large amounts of visual data, following recent work on deep learning for view interpolation [47, 100]. However, our approach differs in key ways from prior work. First, we seek a scene representation that can be predicted once from a pair of input views, then reused to predict many output views, unlike in prior work where each output view must be predicted separately. Second, we need a representation that can effectively capture surfaces that are hidden in one or both input views. We propose a layered representation called a Multiplane Image (MPI) that has both of these properties. Finally, we need training data that matches our task. Simply collecting stereo pairs is not sufficient, because for training we also require additional views that are some distance from an input stereo pair as our ground truth. We propose a simple, surprising source for such data—online video, e.g., from YouTube, and show that large amounts of suitable data can be mined at scale for our task.

In experiments we compare our approach to recent view synthesis methods, and perform a number of ablation studies. We show that our method achieves better numerical performance on a held-out test set, and also produces more spatially stable output imagery since our inferred scene representation is shared for synthesizing all target views. We also show that our learned model generalizes to other datasets without re-training, and is effective at magnifying the narrow baseline of stereo imagery captured by cell phones and stereo cameras.

6.2 Background

Classical approaches to view synthesis View synthesis—i.e., taking one or more views of a scene as input, and generating novel views—is a classic problem in computer graphics that forms the core of many image-based rendering systems. Many approaches focus on the interpolation setting, and operate by either interpolating rays from dense imagery ("light field rendering") [125, 64], or reconstructing scene geometry from sparse views [32, 239, 77]. While these methods yield high-quality novel views, they do so by compositing the corresponding input pixels/rays, and typically only work well with multiple (> 2) input views. View synthesis from stereo imagery has also been considered, including converting 3D stereoscopic video to multi-view video suitable for glasses-free automultiscopic displays [168, 33, 19, 103] and 4D light field synthesis from a micro-baseline stereo pair [229], as well as generalizations that reconstruct geometry from multiple small-baseline views [221, 67]. While we also focus on stereo imagery, the techniques we present can also be adapted to single-view and multi-view settings. We also target much larger extrapolations than prior work.

Learning-based view synthesis More recently, researchers have applied powerful deep learning techniques to view synthesis. View synthesis can be naturally formulated as a learning problem by capturing images of a large number of scenes, withholding some views of each scene as ground truth, training a model that predicts such missing views from one or more given views, and comparing these predicted views to the ground truth as the loss or objective that the learning seeks to optimize. Recent work has explored a number of deep network architectures, scene representations, and application scenarios for learning view synthesis.

Flynn et al. [47] proposed a view interpolation method called DeepStereo that predicts a volumetric representation from a set of input images, and trains a model using images of street scenes. Kalantari et al. [100] use light field photos captured by a Lytro camera [141] as training data for predicting a color image for a target interpolated viewpoint. Both of these methods predict a representation in the coordinate system of the *target* view. Therefore, these methods must run the trained network for each desired target view, making real-time rendering a challenge. Our method predicts the scene representation once, and reuses it to render a range of output views in real time. Further, these prior methods focus on interpolation, rather than extrapolation as we do.

Other recent work has explored the problem of synthesizing a stereo pair [213], large camera motion [235], or even a light field [182] from a *single* image, an extreme form of extrapolation. Our work focuses on the increasingly common scenario of narrow-baseline stereo pairs. This two-view scenario potentially allows for generalization to more diverse scenes and larger extrapolation than the single-view scenario. The recent single-view method of Srinivasan et al., for instance, only considers relatively homogeneous datasets such as macro shots of flowers, and extrapolates up to the small baseline of a Lytro camera, whereas our method is able to operate on diverse sets of indoor and outdoor scenes, and extrapolate views sufficient to allow slight head motions in a VR headset.

Finally, a variety of work in computer vision has used view synthesis as an indirect form of supervision for other tasks, such as predicting depth, shape, or optical flow from one or more images [55, 59, 234, 191, 197, 134]. However, view synthesis is not the explicit goal of such work.

Scene representations for view synthesis A wide variety of scene representations have been proposed for modeling scenes in view synthesis tasks. We are most interested in representations that can be predicted once and then reused to render multiple views at runtime. To achieve such a capability, representations are often volumetric or otherwise involve some form of *layering*. For instance, layered depth images (LDIs) are a generalization of depth maps that represent a scene using several layers of depth maps and associated color values [176]. Such layers allow a user to "see around" the foreground geometry to the occluded objects that lie behind. Zitnick et al., represent scenes using per-input-image depth maps, but also solve for alpha matted layers around depth discontinuities to achieve high-quality interpolation [239]. Perhaps closest to our representation is that of Penner and Zhang [158]. They achieve softness by explicitly modeling confidence, whereas we model transparency which leads to a different method of compositing and rendering. Additionally, whereas we build one representation of a scene, they produce a representation for each input view and

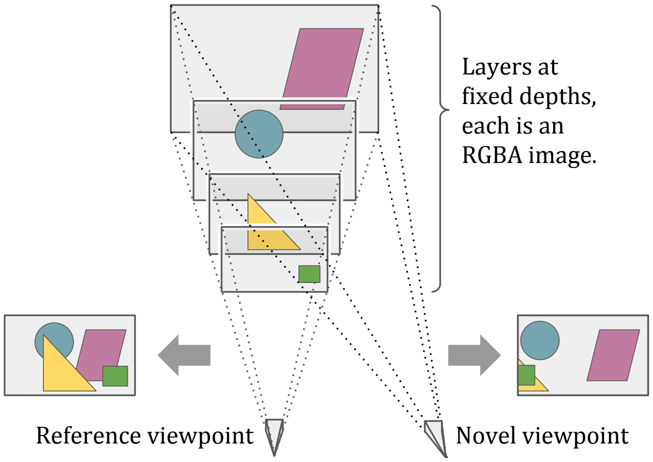


Figure 6.2: An illustration of the multiplane image (MPI) representation. An MPI consists of a set of fronto-parallel planes at fixed depths from a reference camera coordinate frame, where each plane encodes an RGB image and an alpha map that capture the scene appearance at the corresponding depth. The MPI representation can be used for efficient and realistic rendering of novel views of the scene.

then interpolate between them. Our representation is also related to the classic layered representation for encoding moving image sequences by Wang and Adelson [201], and to the layered attenuators of Wetzstein, et al. [206], who use actual physical printed transparencies to construct lightfield displays. Finally, Holroyd et al [82] explore a similar representation to ours but in physical form.

The multiplane image (MPI) representation we use combines several attractive properties of prior methods, including handling of multiple layers and "softness" of layering for representing mixed pixels around boundaries or reflective/transparent objects. Crucially, we also found it to be suitable for learning via deep networks.

6.3 Approach

Given two images I_1 and I_2 with known camera parameters, our goal is to learn a deep neural net to infer a global scene representation suitable for synthesizing novel views of the same scene, and in particular extrapolating beyond the input views. In this section, we first describe our scene representation and its characteristics, and then present our pipeline and objective for learning to predict such representation. Note that while we focus on stereo input in this work, our approach could be adapted to more general view synthesis setups with either single or multiple input views.

6.3.1 Multiplane image representation

The global scene representation we adopt is a set of fronto-parallel planes at a fixed range of depths with respect to a reference coordinate frame, where each plane d encodes an RGB color image C_d and an alpha/transparency map α_d. Our representation, which we call a *Multiplane Image* (MPI), can thus be described as a collection of such RGBA layers $\{(C_1, \alpha_1), \ldots, (C_D, \alpha_D)\}$, where D is the number of depth planes. An MPI is related to the *Layered Depth Image* (LDI) representation of Shade, et al. [176], but in our case the pixels in each layer are fixed at a certain depth, and we use an alpha channel per layer to encode visibility. To render from an MPI , the layers are composed from back-to-front order using the standard "over" alpha compositing operation. Figure 6.2 illustrates an MPI . The MPI representation is also related to the "selection-plus-color" layers used in DeepStereo [47], as well as to the volumetric representation of Penner and Zhang [158].

We chose MPIs because of their ability to represent geometry and texture including occluded elements, and because the use of alpha enables them to capture partially reflective or transparent objects as well as to deal with soft edges. Increasing the number of planes (which we can think of as increasing the resolution in disparity space) enables an MPI to represent a wider range of depths and allows a greater degree of camera movement. Furthermore, rendering views from an MPI is highly efficient, and could allow for real-time applications.

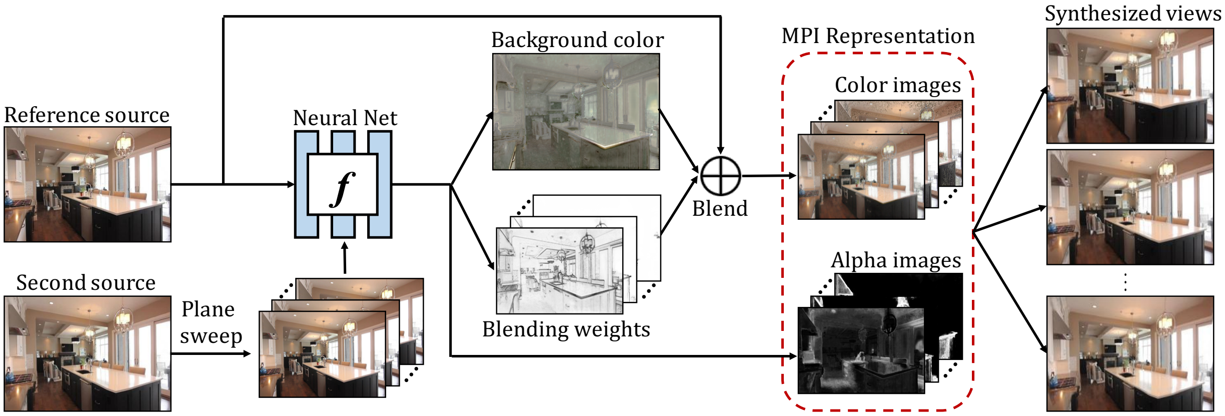


Figure 6.3: Overview of our end-to-end learning pipeline. Given an input stereo image pair, we use a fully-convolutional deep network to infer the multiplane image representation. For each plane, the alpha image is directly predicted by the network, and the color image is blended by using the reference source and the predicted background image, where the blending weights are also output from the network. During training, the network is optimized to predict an MPI representation that reconstructs the target views using a differentiable rendering module (see Section 6.3.3). During testing, the MPI representation is only inferred once for each scene, which can then be used to synthesize novel views with minimal computation (homography + alpha compositing).

Our representation recalls the *multiplane camera* invented at Walt Disney Studios and used in traditional animation [207]. In both systems, a scene is composed of a series of partially transparent layers at different distances from the camera.

6.3.2 Learning from stereo pairs

We now describe our pipeline (see Figure 6.3) for learning a neural net that infers MPIs from stereo pairs. In addition to the input images I_1 and I_2, we take as input their corresponding camera parameters $c_1 = (p_1, k_1)$ and $c_2 = (p_2, k_2)$, where p_i and k_i denote camera extrinsics (position and orientation) and intrinsics, respectively.

The reference coordinate frame for our predicted scene is placed at the camera center of the first input image I_1 (i.e., p_1 is fixed to be the identity pose). Our training set consists of a large set of $\langle I_1, I_2, I_t, c_1, c_2, c_t \rangle$ tuples, where I_t and $c_t = (p_t, k_t)$ denote the target ground-truth image and its camera parameters, respectively. We aim to learn a neural network, denoted by $f_\theta(\cdot)$, that infers an MPI representation using $\langle I_1, I_2, c_1, c_2 \rangle$ as input, such that when the MPI is rendered at c_t it should reconstruct the target image I_t.

Network input To encode the pose information from the second input image I_2, we compute a plane sweep volume (PSV) that reprojects I_2 into the reference camera at a set of D fixed depth planes.[2] Although not required, we choose these depth planes to coincide with those of the output MPI . This plane sweep computation results in a stack of reprojected images $\{\hat{I}_2^1, \ldots, \hat{I}_2^D\}$, which we concatenate along the color channels, resulting in a $H \times W \times 3D$ tensor $\hat{\hat{I}}_2$. We further concatenate $\hat{I}_2$ with I_1 to obtain the input tensor (of size $H \times W \times 3(D+1)$) to the network. Intuitively, the PSV representation allows the network to reason about the scene geometry by simply comparing I_1 to each planar reprojection of I_2—the scene depth at any given pixel is typically at the depth plane where I_1 and the reprojected I_2 agree. Many stereo algorithms work on this principle, but here we let the network automatically learn such relationships through the view synthesis objective.

Network output A straightforward choice of the network output would be a separate RGBA image for each depth plane, where the color image captures the scene appearance and the alpha map encodes the visibility and transparency. However, such an output would be highly over-parameterized, and we found a more parsimonious output to be beneficial. In particular, we assume the color information in the scene can be well modeled by just two images, a foreground and a background image, where the foreground image is simply the reference source I_1, and the background image is predicted by the network, and is intended to capture the appearance of hidden surfaces. Hence, for each depth plane, we compute each RGB image C_d as a per-pixel weighted average of the foreground image I_1 and the predicted background image $\hat{I}_b$:

$$C_d = w_d \odot I_1 + (1 - w_d) \odot \hat{I}_b \ , \tag{6.1}$$

[2] For a rectified stereo pair, reprojected images would simply be shifted versions of I_2, though we consider more general configurations in our setup.

where $\odot$ denotes the Hadamard product, and the blending weights w_d are also predicted by the network. Intuitively, I_1 would have a higher weight at nearer planes where foreground content is dominant, while $\hat{I}_b$ is designed to capture surfaces that are occluded in the reference view. Note that the background image need not itself be a natural image, since the network can exploit the alpha and blending weights to selectively and softly use different parts of it at different depths. Indeed, there may be regions of a given background image that are never used in new views.

In summary, the network outputs the following quantities: 1) an alpha map α_d for each plane, 2) a global RGB background image $\hat{I}_b$ and 3) a blending weight image w_d for each plane representing the relative proportion of the foreground and background layers at each pixel. If we predict D depth layers each with a resolution of $W \times H$, then the total number of output parameters is $WH \cdot (2D + 3)$ (vs. $WH \cdot 4D$ for a direct prediction of an MPI). These quantities can then be converted to an MPI .

6.3.3 Differentiable view synthesis using MPIs

Given the MPI representation with respect to a reference frame, we can synthesize a novel view $\hat{I}_t$ by applying a planar transformation (inverse homography) to the RGBA image for each plane, followed by a alpha-composition of the transformed images into a single image in a back-to-front order. Both the planar transformation and alpha composition are differentiable, and can be easily incorporated into the rest of the learning pipeline.

Planar transformation Here we describe the planar transformation that inverse warps each MPI RGBA plane onto a target viewpoint. Let the geometry of the MPI plane to be transformed (i.e. the source) be $\mathbf{n} \cdot \mathbf{x} + a = 0$, where $\mathbf{n}$ denotes the plane normal, $\mathbf{x} = [u_s, v_s, 1]^T$ the source pixel homogeneous coordinates, and a the plane offset. Since the source MPI plane is fronto-parallel to the reference source camera, we have $\mathbf{n} = [0, 0, 1]$ and $a = -d_s$, where d_s is the depth of the source MPI plane. The rigid 3D transformation matrix mapping from source to target camera is defined by a 3D rotation R and translation $\mathbf{t}$, and the source and target camera intrinsics are denoted k_s and k_t, respectively. Then for each pixel (u_t, v_t) in the target MPI plane, we use the standard inverse homography [71] to obtain

$$\begin{bmatrix} u_s \\ v_s \\ 1 \end{bmatrix} \sim k_s \left(R^T + \frac{R^T \mathbf{t} \mathbf{n} R^T}{a - \mathbf{n} R^T \mathbf{t}} \right) k_t^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix} \tag{6.2}$$

Therefore, we can obtain the color and alpha values for each target pixel $[u_t, v_t]$ by looking up its correspondence $[u_s, v_s]$ in the source image. Since $[u_s, v_s]$ may not be an exact pixel coordinate, we use bilinear interpolation among the 4-grid neighbors to obtain the resampled values (following [92, 235]).

Alpha compositing After applying the planar transformation to each MPI plane, we then obtain the predicted target view by alpha compositing the color images in back-to-front order using the standard *over* operation [160].

6.3.4 Objective

Given the MPI inference and rendering pipeline, we can train a network to predict MPIs satisfying our view synthesis objective. Formally, for a training set of $\langle I_1, I_2, I_t, c_1, c_2, c_t \rangle$ tuples, we optimize the network parameters by:

$$\min_\theta \sum_{\langle I_1, I_2, I_t, c_1, c_2, c_t \rangle} \mathcal{L}(\mathcal{R}(f_\theta(I_1, I_2, c_1, c_2), c_t), I_t) \ , \tag{6.3}$$

where $\mathcal{R}(\cdot)$ denotes the rendering pipeline described in Section 6.3.3 that synthesizes a novel view from the target camera c_t using the inferred MPI $f_\theta(I_1, I_2, c_1, c_2)$, and $\mathcal{L}(\cdot)$ is the loss function between the synthesized view and the ground-truth. In this work, we use a deep feature matching loss (also referred to as the "perceptual loss" [97, 36, 228]), and specifically use the normalized VGG-19 [180] layer matching from [21]:

$$\mathcal{L}(\hat{I}_t, I_t) = \sum_l \lambda_l \|\phi_l(\hat{I}_t) - \phi_l(I_t)\|_1 \ , \tag{6.4}$$

where $\{\phi_l\}$ is a set of layers in VGG-19 (`conv1_2`, `conv2_2`, `conv3_2`, `conv4_2`, and `conv5_2`) and the weight hyperparameters $\{\lambda_l\}$ are set to the inverse of the number of neurons in each layer.

6.3.5 Implementation details

Unless specified otherwise, we use $D = 32$ planes set at equidistant disparity (inverse depth) with the near and far planes at 1m and 100m, respectively.

Network architecture We use a fully-convolutional encoder-decoder architecture (see Table 6.1 for detailed specification). The encoder pathway follows similar design as VGG-19 [180], while the decoder consists of deconvolution (fractionally-strided convolution) layers with skip-connections from lower layers to capture fine texture details. Dilated convolutions [222, 20] are also used in intermediate layers `conv4_1,2,3` to model larger scene context while maintaining the spatial resolution of the feature maps. Each layer is followed by a `ReLU` nonlinearity and layer normalization [7] except for the last layer, where `tanh` is used and no layer normalization is applied. Each of the last layer outputs (32 alpha images, 32 blending weight images, and 1 background RGB image) is further scaled to match the corresponding valid range (e.g. $[0, 1]$ for alpha images).

Layer	k	s	d	chns	in	out	input
conv1_1	3	1	1	99/64	1	1	$I_1 + \hat{I}_2$
conv1_2	3	2	1	64/128	1	2	conv1_1
conv2_1	3	1	1	128/128	2	2	conv1_2
conv2_2	3	2	1	128/256	2	4	conv2_1
conv3_1	3	1	1	256/256	4	4	conv2_2
conv3_2	3	1	1	256/256	4	4	conv3_1
conv3_3	3	2	1	256/512	4	8	conv3_2
conv4_1	3	1	2	512/512	8	8	conv3_3
conv4_2	3	1	2	512/512	8	8	conv4_1
conv4_3	3	1	2	512/512	8	8	conv4_2
conv5_1	4	.5	1	1024/256	8	4	conv4_3 + conv3_3
conv5_2	3	1	1	256/256	4	4	conv5_1
conv5_3	3	1	1	256/256	4	4	conv5_2
conv6_1	4	.5	1	512/128	4	2	conv5_3 + conv2_2
conv6_2	3	1	1	128/128	2	2	conv6_1
conv7_1	4	.5	1	256/64	2	1	conv6_2 + conv1_2
conv7_2	3	1	1	64/64	1	1	conv7_1
conv7_3	1	1	1	64/67	1	1	conv7_2

Table 6.1: Our network architecture, where **k** is the kernel size, **s** the stride, **d** kernel dilation, **chns** the number of input and output channels for each layer, **in** and **out** are the accumulated stride for the input and output of each layer, and **input** denotes the input source of each layer with + meaning concatenation. See Section 6.3.5 for more details.

Training details We implement our system in TensorFlow [1]. We train the network using the ADAM solver [114] for 600K iterations with learning rate $0.0002, \beta_1 = 0.9, \beta_2 = 0.999$, and batch size 1. During training, the images and MPI have a spatial resolution of 1024×576, but the model can be applied to arbitrary resolution at test time in a fully-convolutional manner. Training takes about one week on a Tesla P100 GPU.

6.4 Data

For training we require triplets of images together with their relative camera poses and intrinsics. Creating such a dataset from scratch would require carefully capturing simultaneous photos of a variety of scenes from three or more appropriate viewpoints per scene. Instead, we identified an existing source of massive amounts of such data: video clips on YouTube shot from a moving camera. By sampling frames from such videos, we can obtain very large amounts of data comprising multiple views of the same scene shot from a variety of baselines. For this approach to work, we need to be able to identify suitable video clips, i.e., clips shot from a moving camera but with a static scene, with minimal artifacts such as motion blur or rolling-shutter distortion, and without other editing effects such as titles and overlays. Finally, given a suitable clip, we must estimate the camera parameters for each frame.

While many videos on YouTube are not useful for our purposes, we found a surprisingly large amount of suitable content, across several categories of video. One such category is real estate footage. Typical real estate videos feature a series of shots of indoor and outdoor scenes (the interior of a room or stairway, exterior views of a house, footage of the surrounding area, etc). Shots typically feature smooth camera movement and little or no scene movement. Hence, we decided to build a dataset from real estate videos as a large and diverse source of multi-view training imagery.

Accordingly, the rest of this section describes the dataset we collected, consisting of over 7,000 video clips from 1 to 10 seconds in length, together with the camera position, orientation and field of view for each frame in the sequence. To build this dataset, we devised a pipeline for mining suitable clips from YouTube. This pipeline consists of four main steps: 1) identifying a set of candidate videos to download, 2) running a camera tracker on each video to both estimate an initial camera pose for each frame and to subdivide the video into distinct shots/clips, 3) performing a full bundle adjustment to derive high-quality poses for each clip, and 4) filtering to remove any remaining unsuitable clips.

6.4.1 Identifying videos

We manually found a number of YouTube channels that published real estate videos exclusively or almost exclusively, and used the YouTube API to retrieve videos IDs listed under each channel. This yielded a set of approximately 1,500 candidate videos.

6.4.2 Identifying and tracking clips with SLAM

We wish to subdivide each video into individual clips, and identify clips that have significant camera motion. We found few readily available tools for performing camera tracking on arbitrary videos in the wild. Initially, we tried to use structure-from-motion methods developed in computer vision, such as COLMAP [174]. These methods are optimized for photo collections, and we found them to be slow and prone to failure when applied to video sequences. Instead, we found that for our purposes we could adapt modern algorithms for SLAM (Simultaneous Localization and Mapping) developed in the robotics community.

Visual SLAM methods take as input a series of frames, and build and maintain a sparse or semi-dense 3D reconstruction of the scene while estimating the viewpoint of the current frame in a way consistent with this reconstruction. We use the ORB-SLAM2 system [151], though other methods could also apply [49, 41].

SLAM algorithms are not designed to process videos containing multiple shots with cuts and dissolves between them, and they typically care only about the accuracy of the *current* frame's pose—in particular, as the scene is refined over time, earlier frames are not updated and may become inconsistent with the current state of the world. To deal with these issues, our approach is as follows: **1**. Feed successive frames of the video to ORB-SLAM2 as normal. **2**. When the algorithm reports that it has begun to track the camera, mark the start of a clip. **3**. When ORB-SLAM2 fails to track $K = 6$ consecutive frames, or when we reach a maximum sequence length L, consider the clip to have ended. **4**. Keeping the final scene model constant, reprocess all frames in the clip so as to estimate a consistent pose for each camera. **5**. Re-initialize ORB-SLAM2 so it is ready to start tracking a new clip on subsequent frames. In this way, we use ORB-SLAM2 not just to track frames, but also to divide a video into clips using tracking failure as a way to detect shot boundaries.

Since SLAM methods, including ORB-SLAM2 , require known camera intrinsics such as field of view (which are unknown for arbitrary online videos), we simply assume a field of view of 90 degrees. This assumption worked surprisingly well for the purposes of identifying good clips. Finally, for the sake of speed, at this stage we process a lower resolution version of the video. The result of the above processing is a set of clips or sequences for each video, along with a preliminary set of camera parameters.

6.4.3 Refining poses with bundle adjustment

We next process each sequence at higher resolution, using a standard structure-from-motion pipeline to extract features from each frame, match these features across frames, and perform a global bundle adjustment using the Ceres non-linear least squares optimizer [3]. We initialize the cameras using the poses found by ORB-SLAM2 , and add a weak penalty to the optimization that encourages the parameters not to stray too far from their initial values. The output for each sequence is a set of adjusted camera poses, an estimated field of view, and a sparse point cloud representing the scene. An example output is illustrated in Figure 6.4.

One difficulty with this process is that there is no way to determine global scene scale, so our reconstructed camera poses are up to an arbitrary scale per clip. This ambiguity will become important when we represent scenes with MPIs , because our representation is based on layers at specific depths, as described in Section 6.3.5. Hence, we “scale-normalize” each sequence using the estimated 3D point cloud, scaling it so that the nearest scene geometry is approximately a fixed distance from the cameras. In particular, for each frame we compute the 5th percentile depth among all point depths from that frame’s camera. Computing this depth across all cameras in a sequence gives us a set of “near plane” depths. We scale the sequence so that the 10th percentile of this set of depths is 1.25m. (Recall that our MPI representation uses a near plane of 1m.)

6.4.4 Filtering and clipping

If the source video contains cross-fades, some frames may show a blend of two scenes. We discard ten frames from the beginning and end of each clip, which eliminates most such frames.

Occasionally the estimated camera poses for a sequence do not form a smooth track, which can indicate that we were unable to track the camera accurately. We define a frame to be *smooth* if its camera position p_i is sufficiently close to the average of the two adjacent camera positions, specifically if $\|p_i - (p_{i+1} + p_{i-1})/2\| < 0.2 \times \|p_{i+1} - p_{i-1}\|$. For each sequence, we find the longest consecutive subsequence in which all frames are smooth, and discard the rest.

Finally we discard all remaining sequences of fewer than 30 frames. From an input set of approximately 1500 videos, this pipeline produces a set of ∼7,000 sequences with a total of ∼750K frames.

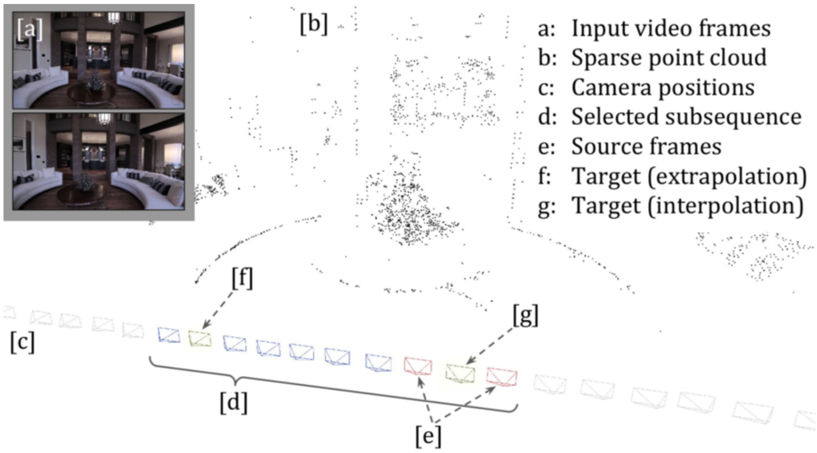


Figure 6.4: Dataset output and frame selection, showing estimated camera trajectory and sparse point cloud. See section 6.4.5 for a detailed description.

6.4.5 Choosing training triplets

Figure 6.4 shows an example of the result of this processing, including input video frames [a] (just two frames are shown here), and the sparse point cloud [b] and camera track [c] resulting from the structure from motion pipeline. As described in Section 6.3.2, for our application we require tuples $\langle I_1, I_2, I_t, c_1, c_2, c_t \rangle$, including cases where I_t is an *extrapolation* from I_1 and I_2. We sample tuples from our dataset by first selecting from each sequence a random subsequence [d] of length 10, with stride (gap between selected frames) chosen randomly from 1 to 10. From this subsequence we then randomly choose two different frames and their poses to be the inputs I_1, I_2, c_1, and c_2 [e], and a third frame to be the target I_t, c_t.

Depending on which frames are chosen, the target frame may require extrapolation [f] (of up to a factor of nine times the distance between I_1 and I_2, assuming a linearly moving camera) or interpolation [g] from the inputs. We chose to learn to predict views from a variety of positions relative to the source imagery so as not to overfit to generating images at a particular distance during training.

6.5 Experiments and results

In this section we evaluate the performance of our method, and compare it with several view synthesis baselines. Our test set consists of 1,329 sequences that did not overlap with the training set. For each sequence we randomly sample a triplet (two source frames and one target frame) for evaluation. We first visualize the MPI representation inferred by our model, and then provide detailed comparison with other recent view synthesis methods. We further validate our model design with various ablation studies, and finally highlight the utility of our method through several applications. For quantitative evaluation, we use the standard SSIM [203] and PSNR metrics.

6.5.1 Visualizing the multiplane images

We visualize examples of the MPI representation inferred by our network in Figure 6.5. Despite having no direct color or alpha ground-truth for each MPI plane during training, the inferred MPI is able to capture the scene appearance in a layer-wise manner (near to far) respecting the scene geometry, which allows realistic rendering of novel views from the representation.

We also demonstrate view extrapolation capability of the MPI representation in Figure 6.6, where we use the central two frames of a registered video sequence as input, and synthesize the previous and future frames with the inferred MPI . Please see the supplemental video for animations of these rendered sequences.

6.5.2 Comparison with Kalantari et al.

We compare our model with Kalantari et al. [100], a state-of-the-art learning-based view synthesis method. A critical difference compared to our method is that Kalantari et al. has an *independent* rendering process for each novel view of the scene, and needs to re-run the entire inference pipeline every time a new view is queried, which is computationally prohibitive for real-time applications. In contrast, our method predicts a scene-level MPI representation that can render any novel viewpoint in real-time with minimal computation (inverse homography + alpha compositing).

We train and test two variants of their method on our data: 1) same network architecture (4 convolution layers) and pixel reconstruction loss from the original paper; 2) our network architecture (which is deeper with skip connections) with perceptual loss. For fair comparison, we use the same number of input planes as ours for constructing the plane sweep volume in their input. See Section 6.5.4 for discussion on the effect of varying the number of depth planes.

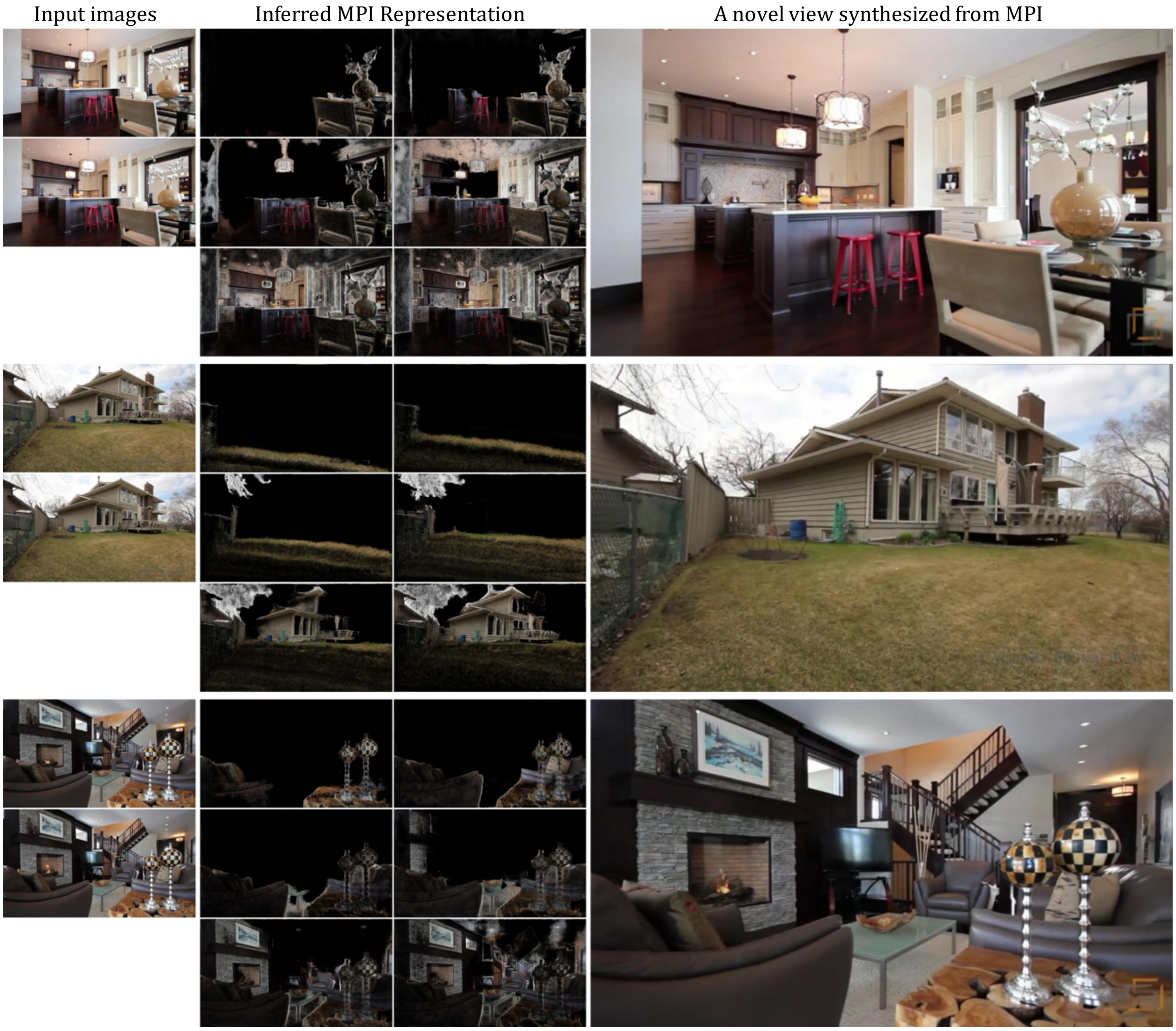

Input images
Inferred MPI Representation
A novel view synthesized from MPI

Figure 6.5: Sample visualization of the input image pair (left), our inferred MPI representation (middle), where we show the alpha-multiplied color image at a subset of the depth planes from near to far (top to bottom, left to right), and novel views rendered from the MPI (right). The predicted MPI is able to capture the scene appearance in a layer-wise manner (near to far) respecting the scene geometry.

Table 6.2 shows mean SSIM and PSNR similarity metrics for each method across our test set. To measure if one method is consistently better than another, we also rank the methods on each test triplet and compute the average rank for each method. An average rank of 1.0 for PSNR, for example, would mean that this method always had the highest PSNR score.

We find that 1) our network architecture is significantly more effective than the simple 4-layer network used in the original Kalantari paper; 2) the VGG perceptual loss helps im-

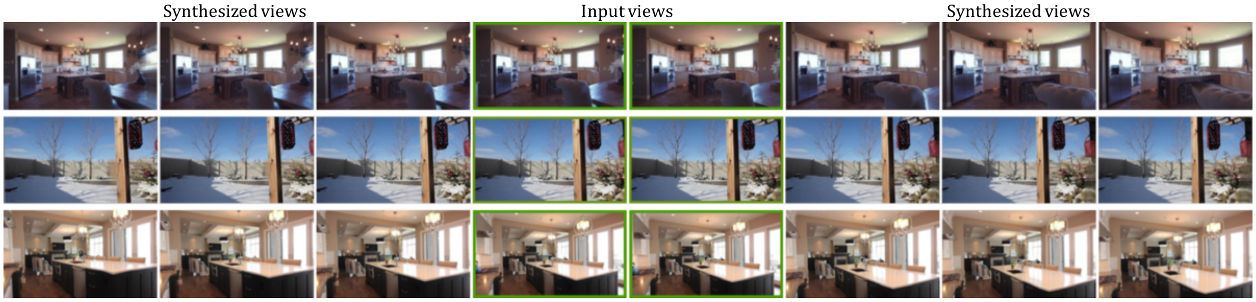


Figure 6.6: Sample view extrapolation results using multiplane images . The central two frames (green) are the input to our network, and the inferred MPI is used to render both past and future frames in the same video sequence.

Method	Network	Loss	SSIM		PSNR	
			Mean	Rank	Mean	Rank
Kalantari	Kalantari	pixel	0.696	4.0	31.41	3.7
Kalantari	Ours	VGG	0.822	2.1	32.93	2.0
Ours	Ours	Pixel	0.812	2.6	32.42	2.8
Ours	**Ours**	**VGG**	**0.835**	**1.4**	**33.10**	**1.5**

Table 6.2: .
Quantitative comparison between our model and variants of the baseline Kalantari model [100]. Higher SSIM/PSNR mean and lower rank are better. See Section 6.5.2 for more details.

prove the performance over the pixel reconstruction loss (see Section 6.5.4 for discussion); 3) our model outperforms the better of the two Kalantari variants (VGG with our network architecture), indicating the high-quality of novel views rendered from the MPI representation.

We also observe that when rendering continuous view sequences of the same scene, our results tend to be more spatially coherent than Kalantari, and produce fewer frame-to-frame artifacts. We hypothesize that this is because, unlike the Kalantari model, we infer a single scene-level MPI representation that is shared for rendering all target views, which implicitly imposes a smoothness prior when rendering nearby views. Please see the video for qualitative comparisons of our method to Kalantari on rendered sequences.

6.5.3 Comparison with extrapolation methods

We compare with a non-learning view extrapolation approach by Zhang et al. [229], which reconstructs a 4D light field from micro-baseline stereo pairs using disparity-assisted phase based synthesis (DAPS). For fair comparison, we directly apply our model trained on the real estate data to the HCI light field dataset [204]. As shown in Figure 6.7, our model generalizes well on the HCI dataset without any fine-tuning, and compares favorably with Zhang et al. around depth boundaries, where our method introduces fewer distortion artifacts. We find that the method of Zhang et al. performs well for small view extrapolations, but breaks down more quickly around object boundaries with increasing extrapolation distance.

We also trained appearance flow [235] on our dataset, but found rendered views exhibited significant artifacts, such as straight lines becoming distorted. This method appears more suited to object-centric synthesis than to scene rendering, and it is not able to fully exploit correlations between views since the trained network operates on each input image separately.

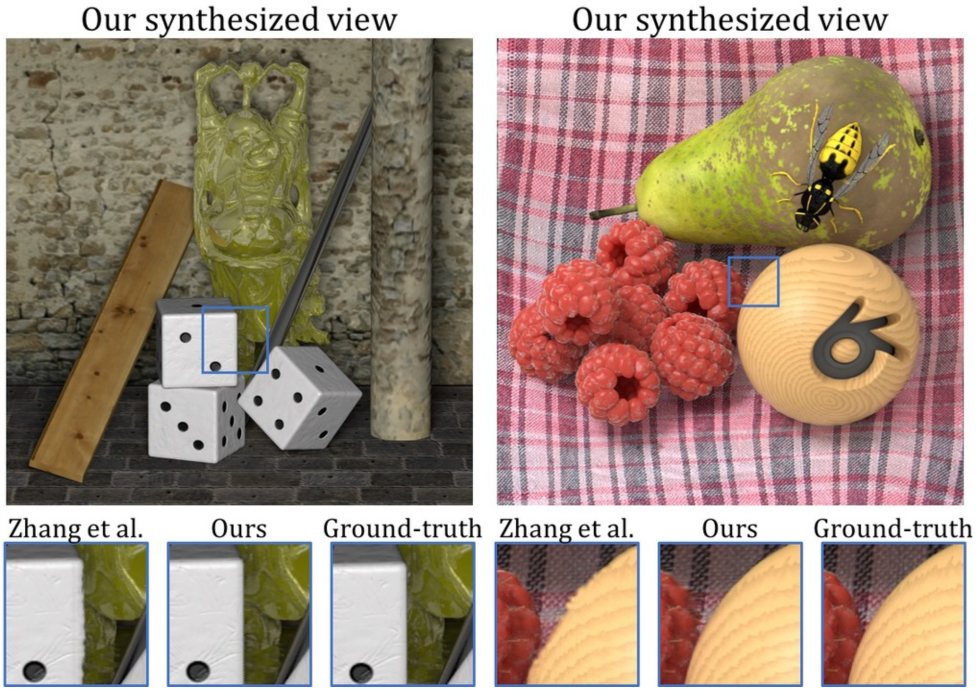


Figure 6.7: Comparison with Zhang et al. [229] on the HCI light field dataset [204]. Note the differences around object boundaries.

6.5.4 Ablation studies

Perceptual loss To illustrate the effect of the perceptual loss, we compare our final model with a baseline model trained using L1 loss in the RGB pixel space. As shown in Figure 6.8, our final model trained using the perceptual loss better preserves object structure and texture details in the synthesized results than the baseline. The benefit of training with perceptual loss is further verified with quantitative evaluation in Table 6.2.

Color layer prediction In Section 6.3.2, we propose that our network create the color values for each MPI plane as a weighted average of a network predicted "background" image and the reference source image. Here we compare several variants of the color prediction format (ordered by increasing level of representation flexibility):

1. None. No color image or blending weights are predicted by the network. The reference source image is used as the color image at each MPI plane.

2. Single image. The network predicts a single color image shared for all MPI planes.

3. Background + blending weights (our preferred format). The network predicts a background image and blending weights. The reference source is used as the foreground image.

4. Foreground + background + blending weights. In contrast to the previous variant, instead of using the reference source as the foreground image, the network predicts an extra foreground image for blending with the background.

5. All images. The network directly outputs the color image at each MPI plane.

We compare the performance of these variants in Table 6.3 and show a qualitative example in Figure 6.9. Although "BG+blending weights" slightly outperforms the other variants, all the variants (other than "FG+BG+blending weights") produce competitive results. The "None" and "Single image" variants suffer in areas where the target view contains details that are occluded in the reference image but visible in the second input image. The "BG+blending weights" format can represent these areas better since not all MPI planes need to have the same color data. The "FG+BG+blending weights" variant is slightly more powerful as the foreground image is not restricted, and the "All images" variant, with a separate color image for each plane, is the only variant that can fully represent a scene with depth complexity greater than 2. However, in our experiments these last two variants both performed slightly worse than "None". We hypothesize that the larger output space and less utilization of the reference image makes the learning harder with these output formats, and that the relatively small camera movement limits the depth complexity required.

Number of depth planes As shown in Table 6.4, our model performance improves as more depth planes are used in the inferred MPI representation. We are currently limited to 32 planes due to memory constraints, but could overcome this with future hardware or alternative networks. As seen in Figure 6.10, the greater the offset between the reference view and the rendered view, the more planes are needed to render the scene accurately.

6.5.5 Applications

In this section we describe two applications of our trained model: 1) taking a narrow-baseline stereo pair from a cell phone camera and extrapolating to an average human

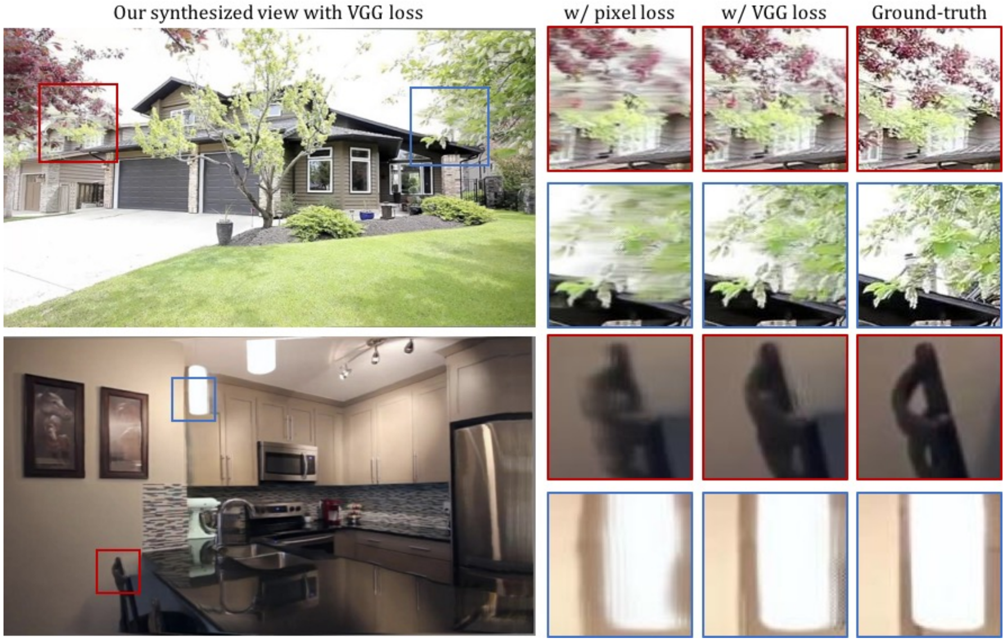


Figure 6.8: Comparison between the models trained using pixel reconstruction loss and VGG perceptual loss. The latter better preserves object structure, and tends to produce sharper synthesized views.

Color layer prediction	SSIM		PSNR	
	Mean	Rank	Mean	Rank
None	0.833	2.3	33.06	2.1
Single image	0.822	3.9	32.51	3.9
BG + Blend weights	**0.835**	**1.6**	**33.09**	**1.6**
FG + BG + Blend weights	0.819	4.1	32.50	3.7
All images	0.825	3.2	32.53	3.8

Table 6.3: Quantitative evaluation of variants of network color output, ordered by increasing degree of flexibility (top to bottom). Higher SSIM/PSNR mean and lower rank are better.

interpupillary-distance (IPD)-spaced stereo pair, and 2) taking an image pair from a large-baseline stereo camera and extrapolating a “1D lightfield” of views between and beyond the source images.

Cell phone image pairs → IPD stereo pair We captured a set of image pairs with an iPhone X, a recent dual-lens camera phone with a baseline of ∼1.4cm, using an app that saves both captured views. Because the focal lengths of the two cameras are different, the app crops the wider-angle image to match the narrower field-of-view image. For each image

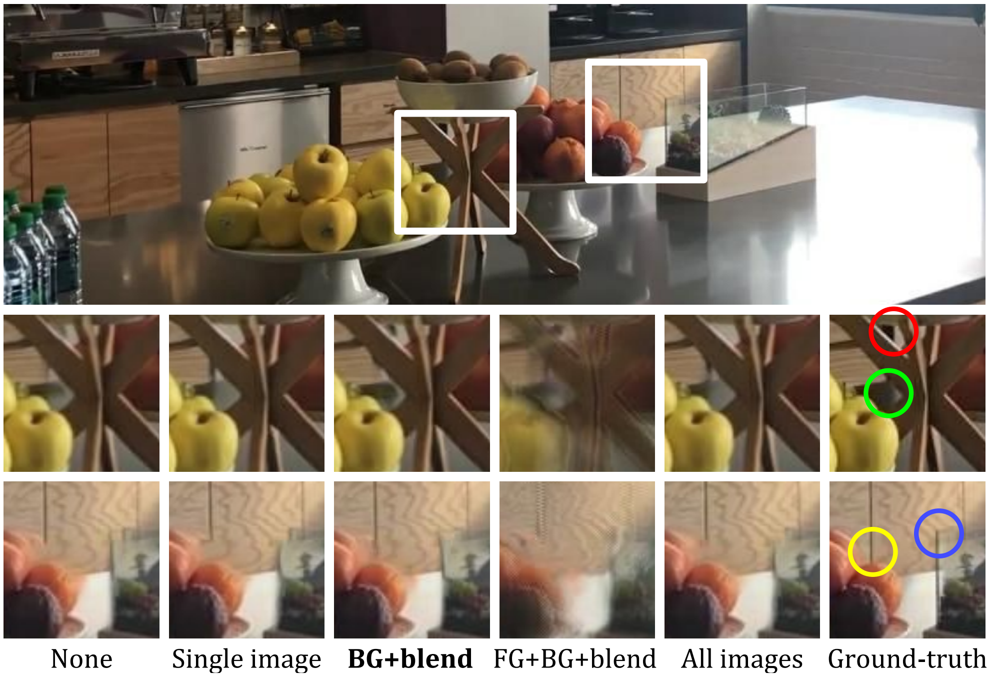


Figure 6.9: Comparison between different color prediction formats. Note in particular the rendering of disoccluded background details, such as the rear wall (red), its reflection in the table surface (green), cupboard door (yellow) and corner of vase (blue). All the variants (except “FG+BG+blend”) produce competitive results with slight differences. See Section 6.5.4 for more details.

MPI depth planes	SSIM		PSNR	
	Mean	Rank	Mean	Rank
D = 8	0.766	2.99	32.12	2.96
D = 16	0.812	1.98	32.73	1.97
D = 32	**0.835**	**1.03**	**33.09**	**1.07**

Table 6.4: Evaluating the effect of varying the number of depth planes for the MPI representation. Higher SSIM/PSNR mean and lower rank are better.

pair, we ran a calibration procedure to refine the camera intrinsics using their nominal values as initialization. We then applied our model (trained on real estate data) to magnify the baseline to ∼6.3cm (a magnification factor of 4.5x). Several results are shown in Figure 6.11 as anaglyph images, and in the supplemental video as sway animation. Figure 6.11 highlights how the extrapolated images provide a more compelling sense of 3D, and illustrates how our model can generalize to new scenarios that are atypical of real estate scenes (such as the sculpture of Mark Twain in the first example). Finally, notice that our method can handle

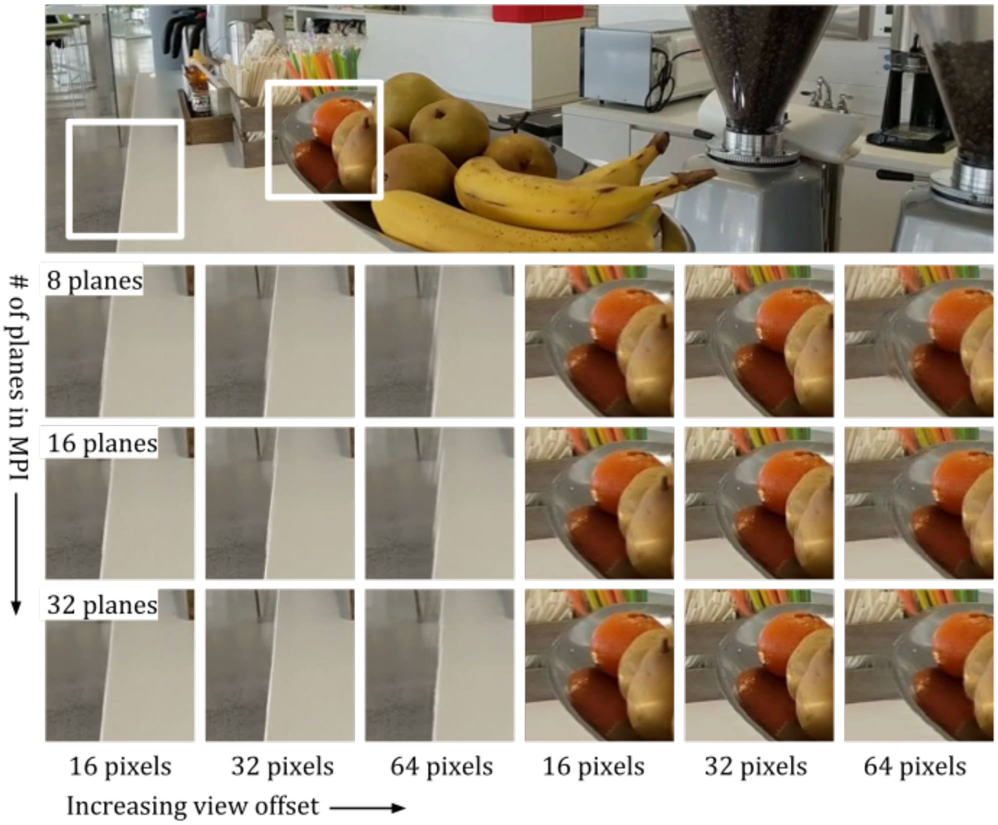
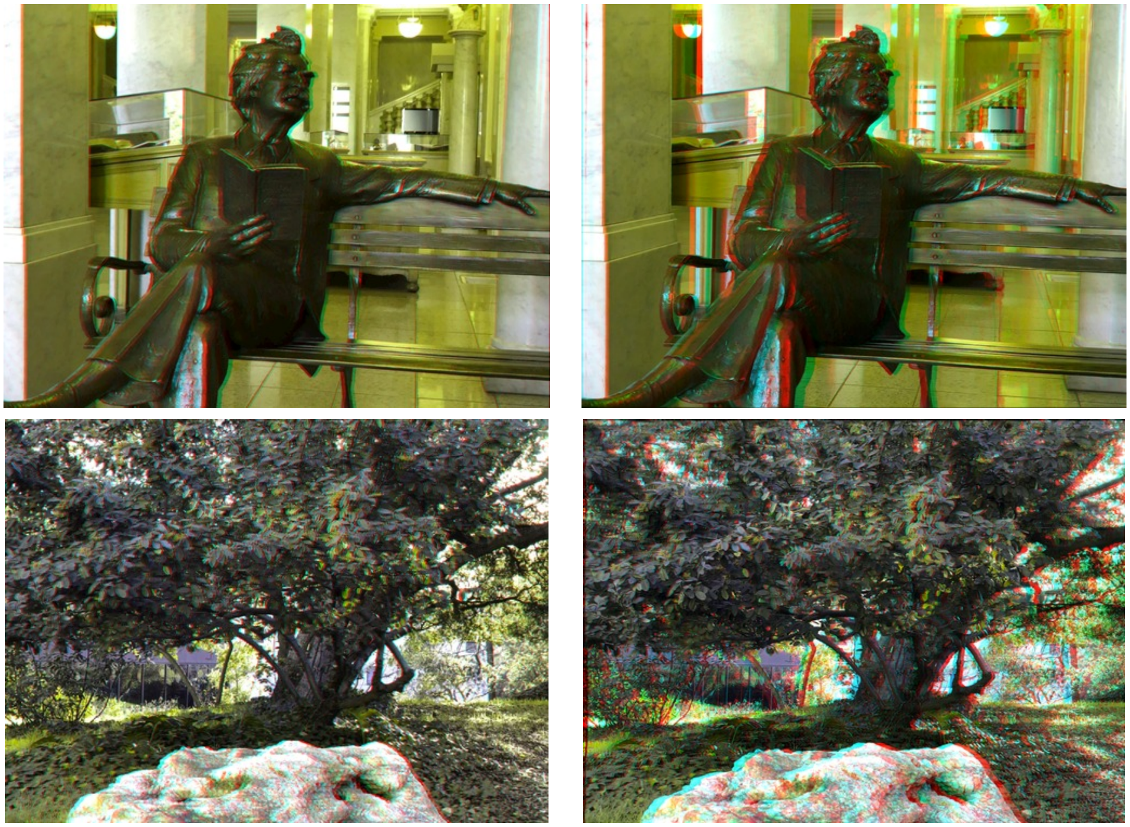


Figure 6.10: Effect of varying the number of depth planes at different view offsets. For two regions of the top image, we show view extrapolations from MPIs with varying numbers of planes. The number of pixels shown is the disparity between front and back planes relative to the reference view. The larger the number of planes, the farther the view can be extrapolated before introducing artifacts. Note the edge of the counter in the first example, and the edges of objects in the second example. (Best viewed zoomed in.)

interesting materials (e.g. the reflective glass and glossy floor in the first scene).

Stereo pairs to extended 1D lightfield We also demonstrate taking a large-baseline stereo pair and synthesizing a continuous "1D lightfield"—i.e., a set of views along a line passing through the source views. For this application, we downloaded stereo pairs shot by a Fujifilm FinePix Real 3D W1 stereo point-and-shoot camera with a baseline of 7.7cm, and extrapolated to a continuous set of views with a baseline of 26.7cm (a magnification factor of ∼3.5x). Figure 6.12 shows an example input and output as anaglyphs; see the supplemental video for animations of the resulting sequences. This input baseline, magnification factor, and scene content represent a challenging case for our model, and artifacts such as stretching in the background can be observed. Nonetheless, the results show plausible interpolations and extrapolations of the source imagery.



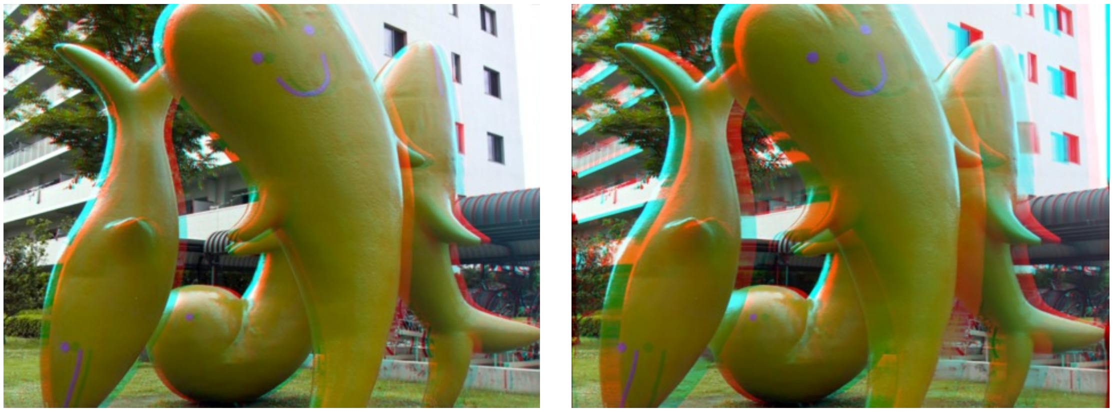
Original baseline Magnified baseline

Figure 6.11: Example stereo magnifications for dual-lens camera. Left: raw stereo pairs captured by an iPhone X, displayed as red-cyan anaglyph images, with a baseline of ∼1.4cm. Right: the same images but with baseline synthetically magnified to ∼6.3cm. Note the significantly enhanced stereo effect. (Best viewed zoomed in and with 3D glasses.)

6.6 Discussion

Having trained on a large and varied dataset, our view synthesis system based on multiplane images is able to handle both indoor and outdoor scenes. We successfully applied it to scenes which are quite different from those in our training dataset. The learned MPIs are effective at representing surfaces which are partially reflective or transparent. Figure 6.13 (a) and (b) show two examples of such surfaces, rendered as anaglyphs with stereo-magnification.

Our method has certain limitations. When fine detail appears in front of a complex background, our model can struggle to place it at the correct depth. Figure 6.13 (c) shows a case where overhead cables appear to jump between two different depths. This may suggest that depth decisions are being made too locally. Figure 6.13 (d) shows the result of extrapolating beyond the limits of the MPI representation. When the disparity between adjacent layers exceeds one pixel we may see duplicated edges, producing a “stack of cards”



Original baseline

Magnified baseline

Figure 6.12: Example stereo magnifications for Fujifilm Real 3D stereo camera. Left: a raw stereo pair from the camera, displayed as red-cyan anaglyph images, with a baseline of ∼7.7cm. Right: the same images but with baseline synthetically magnified to ∼26.7cm. (Best viewed zoomed in and with 3D glasses.) (Photo used under CC license from Flickr user heiwa4126.)

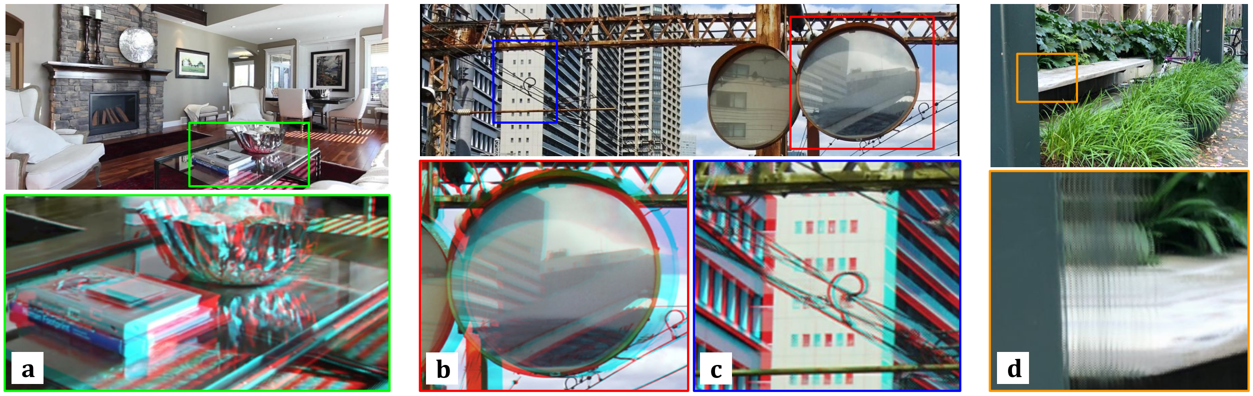


Figure 6.13: Challenging cases. Reference images at top, rendered anaglyph details at bottom: (a) glass table with reflection and transparency, (b) reflection in a dusty curved mirror, (c) fine wires are confused with background, (d) extrapolation beyond the limits of the representation gives a ‘stack of cards’ effect.

effect.

In conclusion, we presented a new representation, training setup, and approach to learning view extrapolation from video data. We believe this framework can also generalize to a variety of different tasks, including extrapolating from more than two input images or from only one, and generating lightfields allowing view movement in multiple dimensions.

Part III

Learning gradual image transformation

Chapter 7

Learning Gradual Image Transformation

The world around us is constantly changing: day turning to dusk, fall to winter, frown to smile. In this chapter, we seek to uncover the gradual changes implicit in a dataset of static photos. We train a neural network to act as a *differential transformer* that takes an image as input and perturbs it slightly in a target direction (e.g., a sub-domain of the dataset). Multiple applications of this transformer then "walk" along the natural image manifold toward the target. However, capturing direct training data for such transient transformations is often difficult. For instance, to capture season progression of a scene, one would need to place the camera at the same spot over a period of several months. Similarly, to capture the aging process of a person, one would need photos of the same person over many years.

Instead of using training data in the form of multiple images of the same instance, we propose a method that is able to learn the gradual transformation given only two unpaired sets of images – one in the source domain (e.g. non-winter) and one in the target domain (e.g. winter). Here "unpaired" means there are no explicit associations between the two sets. At test time, our model applies to a single input image from the source domain, and generates a sequence of gradual progression towards the target domain. We demonstrate that this method can create realistic "movies" of seasons and lighting changes for scenes and attribute progression for facial images.

7.1 Introduction

No image is an island. Rather, it is but a speck in the vast space of possible natural images, connected to its almost-identical cousins by various infinitesimally small transformations. Yet, of the many conceivable ways to transform an image, only a tiny subset are what we would consider meaningful, and discovering these from raw data is a difficult open problem. A classic way of attacking it is by attempting to "disentangle" the underlying global factors of variation within an image set. Typically, this is done by learning an invertible mapping from the data manifold to a latent vector space where the dimensions are either uncorrelated (e.g., PCA [99]) or independent (e.g., ICA [88], VAEs [113]). One then hopes that the disentangled dimensions will correspond to meaningful transformations. Although impressive results have been achieved in certain highly-constrained settings, such as on faces [12], in general the latent dimensions only rarely correspond to interpretable transformations [34]. One possible reason is that these methods model variation through an explicit, global representation, with transformations modeled as directions in a vector space. But the natural image manifold is so vast and heterogeneous, that it is hard to place a global coordinate system on it, i.e. have a global z vector where each dimension controls a specific, globally-consistent factor of variation.

Often, when modeling the full space is too difficult, one can make progress by modeling local neighborhoods instead. That is, rather than represent the data manifold through a global embedding, we implicitly represent it via a set of locally valid transformations. Classic manifold learning methods such as locally-linear embedding [169] and Isomap [189] represent transformations locally and non-parametrically by interpolating between datapoints, but ultimately use these to recover a global embedding.

In this work, we sidestep the global problem entirely, instead representing transformations as parametric functions, implemented as deep nets, that locally perturb data points. We call these *differential transformations*. Specifically, we learn a function $G : \mathcal{X} \rightarrow \mathcal{X}$ that describes a differential transformation starting from datapoint $x \in \mathcal{X}$ that keeps us on the data manifold, $\mathcal{X}$. Applying a series of such transformations, $G \circ \cdots \circ G \circ x$, while regularizing G to only make small changes, produces a smooth progression that "walks" along the manifold in a particular direction.

At training time, our model learns from a set of images from a particular domain (e.g., faces, scenes), with some of the images labeled as having a target attribute (e.g., smiling, snow). At test time, the input to our model is a single image and our goal is to apply small consistent transformations to the input image such that it exhibits more and more of the target attribute.

We achieve this via the interplay of three loss terms. A differential loss encourages that the transformation move a small amount toward a target domain (e.g., winter). At the same time, an adversarial loss keeps the transformation from walking off the data manifold of natural images. Finally, a content-preservation loss penalizes transformations that change the "content" of the image, while allowing its style to be modified. Together, these terms produce smooth and natural transformations that can be used to turn a static image into a movie of desired stylistic change.

7.2 Background

Image generation is a vast field of active research. Our approach touches on a few topics in this area.

Generative adversarial networks (GANs). GANs learn a mapping from random noise to images such that the output cannot be distinguished from real images by an adversary. Our objective can be understood as a GAN where the input is a natural image rather than a random noise vector. This can be seen as an alternate strategy to generating samples from the distribution of natural images, where we perturb existing images to generate new ones rather than starting from scratch.

Image-to-image translation. Image-to-image translation is the problem of converting one visual representation of a scene into another, e.g., sketch to photo, or winter scene to summer scene [91]. Our work is especially related to image-to-image methods that directly model the input-output mapping with a black-box function approximator that does not optimize an objective over internal latent variables. These methods do not require architectural bottlenecks, which allow them to scale relatively easily to high-resolution mappings (e.g., [119, 179, 97, 91, 21]). Often, these methods are given paired data $\{x, y\}$, and learn a regressor $G : x \to y$. In such settings we are given supervision at the level of source and target image *instances*. Other methods consider the unpaired setting in which supervision is at the level of source and target image *sets* $\mathcal{X}$ and $\mathcal{Y}$, with no explicit correspondence between the two, and the goal is to learn the mapping $G : \mathcal{X} \to \mathcal{Y}$ [238, 111, 219, 132, 186]. In the present work, we investigate if we can learn meaningful image-to-image translation functions of the form $G : \mathcal{X} \to \mathcal{X}$, that walk along the manifold $\mathcal{X}$, with side supervision guiding the direction in which we walk.

Attribute-conditioned image generation. Attribute-conditioned generative models learn to synthesize realistic images that exhibit a given attribute. Many approaches model attributes in a latent embedding space. Given image sets $\mathcal{X}$ and $\mathcal{Y}$ corresponding to different values of an attribute (e.g., blond versus brunette), a progression of synthesized results can be produced by traversing the latent space between the domains [195]. In this setting we are given supervision in the form of image sets. Other methods consider a setting where supervision is given in the form of {image, attribute-value} pairs. Here the task is to generate a realistic image conditioned on the latent embedding and given attributes [214, 117, 121, 159]. These methods explicitly model attributes as latent variables in a generative model, and achieve transformations by modifying the latent state.

In contrast, our method avoids latent variables and instead directly models attribute *changes* via an image-to-image transformation function. By avoiding low-dimensional latent representations, our method can scale to high-resolution visual changes. In addition, previous methods have mostly focused on binary attribute changes (e.g., convert a summer scene to a winter scene), rather than continuous changes (show the gradual progression of snow accumulating as the season changes). The previous methods can be extended to modeling gradual changes by training on binary attributes and then at inference time interpolating continuously in latent space between (and beyond) the binary end points, as was done in FaderNets [121]. Such methods rely on the fact that small changes in latent space tend to produce small changes in image style, but this criterion is not explicitly enforced. In contrast, we add an explicit term to encourage smooth and gradual changes in style.

Style transfer. Our task is also closely related to the problem of style transfer. Many approaches to style transfer apply the "style" of an exemplar image to the "content" of another image, e.g., [56, 140, 127, 75]. Our method differs from these in that we do not require exemplar style targets, instead we learn the style from a target *set* of images, and only at training time. Methods for artistic style transfer have been extended to this same setting, where no exemplar is required at test time, but these methods have only been successfully applied on artistic domains where style is well modeled by Gram matrix statistics [193, 97]. Our method additionally differs from past style transfer work in that these methods typically do not explicitly enforce gradual changes.

Video prediction. The problem of future frame prediction is another instance of differential transformation. Methods that tackle this problem generate future frames given a current frame (or frames) as input, typically treating the problem as supervised regression [237, 144], or as generative modeling of a video sequence [198]. Our method differs in that we do not train on video data, instead trying to infer smooth transformations from a set of static images, using only domain-level supervision (i.e. certain images are labeled as belonging to the target domain).

7.3 Approach

Given an image set $\mathcal{X}$, we learn a function G that predicts a sequence of transformations satisfying three key properties: 1) *plausibility*—lying on the image manifold defined by set $\mathcal{X}$, 2) *consistency*—frames of the sequence are related by a common small-step transformation and 3) *identity*—transformed frames maintain the identity of the objects and scenes of the input image. We dub this problem *differential image transformation.* Prior work that tackles this problem is embedding-based and hopes to learn a single knob that linearly controls the degree of transformation. In contrast, our key idea is to formulate the problem as an image-to-image translation and *explicitly* force G to learn differential transformations such that iterative application of G result in a sequence of small changes to the input image towards a specified direction. We use the following three losses to ensure this.

7.3.1 Adversarial loss

To constrain each generated image to be a plausible sample from the image set $\mathcal{X}$, we apply the LS-GAN [143] formulation of the adversarial objective [60], which can be expressed as

$$\mathcal{L}_{\mathsf{adv}} = \mathbb{E}_{x\sim p_{\mathsf{data}}(x)}[(D(x)-1)^2 + D(G(x))^2] \ , \tag{7.1}$$

where G tries to generate images $G(x)$ that look similar to images from $\mathcal{X}$, while D aims to distinguish between generated samples $G(x)$ and real samples x. The above objective can be formulated as a minimax game between D and G: $\min_G \max_D \mathcal{L}_{\mathsf{adv}}(G, D)$.

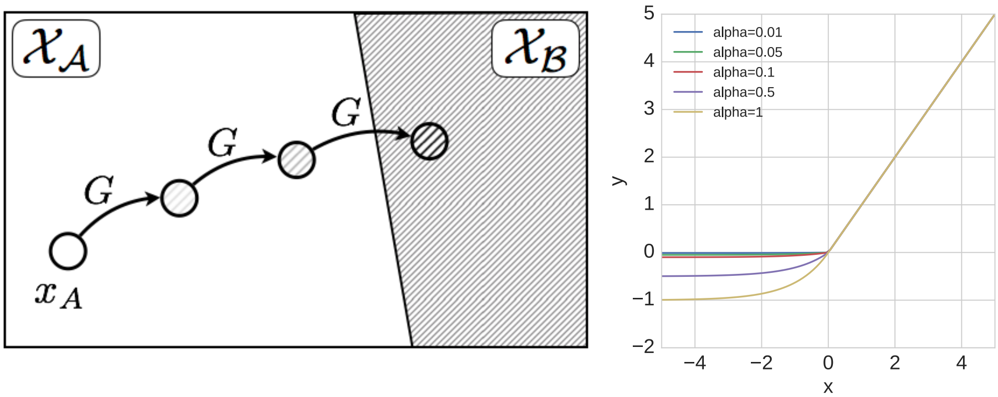


Figure 7.1: *Left*: Visualization of the differential loss, Eqn. 7.2. G is encouraged to gradually move its input x_A toward domain $\mathcal{X}_B$, so that $G(x_A)$ is closer to $\mathcal{X}_B$ than x_A, $G(G(x_A))$ is closer still and so forth. *Right*: Graph of Eqn. 7.3 for varying values of α that determines the saturation point of the differential loss.

7.3.2 Differential loss

While the adversarial loss forces the generated image $G(x)$ to lie on the image manifold defined by $\mathcal{X}$, it does not guarantee that $G(x)$ moves in a meaningful and consistent direction along the manifold. Therefore, we introduce a differential loss that encourages $G(x)$ to be *closer* to a target domain than the input x. Intuitively, if the differential loss is successfully minimized, iterative application of G should move the generated images closer and closer to the target domain in a consistent way.

We assume that $\mathcal{X}$ is roughly split into two sub-domains $\mathcal{X} = \{\mathcal{X}_\mathcal{A}, \mathcal{X}_\mathcal{B}\}$. Given an image sample x_A from $\mathcal{X}_A$, we would like the generated image $G(x_A)$ to be *closer* to $\mathcal{X}_\mathcal{B}$ than the input x_A, as depicted in Figure 7.1 (left). In other words, we want to learn a function G such that $\texttt{dist}(G(x_A), \mathcal{X}_\mathcal{B}) < \texttt{dist}(x_A, \mathcal{X}_\mathcal{B})$ for some distance metric $\texttt{dist}(\cdot)$. This could naively be achieved by minimizing $\texttt{dist}(G(x_A), \mathcal{X}_\mathcal{B}) - \texttt{dist}(x_A, \mathcal{X}_\mathcal{B})$. However, such an objective does not encourage differential learning as the loss is linear with respect to $\texttt{dist}(G(x_A), \mathcal{X}_B)$, so a direct optimization could get unbounded reward for taking larger and larger steps towards $\mathcal{X}_\mathcal{B}$. Instead, we use a modified loss function (with the same form as the ELU [27] activation function) that is attenuated via a parameter α in the negative half-space to encourage learning only small, local changes. Intuitively, a small α encourages small differential changes, as shown in Figure 7.1 (right). The differential objective then becomes

$$\mathcal{L}_{\mathsf{diff}} = \sigma(\texttt{dist}(G(x_A), \mathcal{X}_\mathcal{B}) - \texttt{dist}(x_A, \mathcal{X}_\mathcal{B}))\ , \tag{7.2}$$

where

$$\sigma(y) = \begin{cases} y & \text{if } y > 0, \\ \alpha(\exp(y) - 1) & \text{otherwise.} \end{cases} \tag{7.3}$$

The hyperparameter α controls the value to which the loss saturates. Note that this is only one of several possible forms of differential loss. Other forms could include, for example, the hinge loss or using a constant (small) step size.

We use a learned distance metric for $\texttt{dist}(\cdot)$. Specifically, in addition to the generator G and the discriminator D, we also train a sub-domain classier C that distinguishes between images from $\mathcal{X}_A$ and images from $\mathcal{X}_B$. However, instead of using the standard cross entropy loss, we optimize C with the mean-square error (MSE) with respect to the target label (0 for x_A and 1 for x_B), which we have found to empirically work better. The classifier score (output of the final layer) is used to measure the distance between generated images and the target domain $\mathcal{X}_B$.

7.3.3 Content loss and the full objective

Since our goal is to only make stylistic changes to the input image, we use an additional content loss (in line with prior style transfer works [56, 97]) in order to preserve the identity of the objects depicted in the input image. In particular, we use the MSE loss on the `conv3_3` features from VGG-16 [180] between x_A and $G(x_A)$.

Our full objective becomes

$$\mathcal{L}_{\mathsf{full}} = \mathcal{L}_{\mathsf{adv}} + \beta\mathcal{L}_{\mathsf{diff}} + \gamma\mathcal{L}_{\mathsf{content}} \ , \tag{7.4}$$

where β and γ are hyperparameters balancing the importance among the loss terms.

7.3.4 Implementation details

Architecture. We adopt the generator and discriminator architecture from CycleGAN [238] who base their generator on Johnson *et al.* [97]. The domain classifier follows the same architecture as the discriminator.

Training details. We use $\alpha = 0.25, \beta = 0.5, \gamma = 0.25$ for all our experiments. We use the Adam optimizer [114] with a batch size of 1, learning rate of 0.0002 for G and 0.0001 for D.

7.4 Experiments

We apply our approach to a variety of datasets, and compare with baselines using automatic as well as perceptual metrics. For evaluation we mainly use two datasets: 1) *Transient attributes database* [119] that contains $8,571$ images from 101 webcams annotated with 40 transient attribute labels (e.g. season, weather conditions, lighting), and 2) *CelebA-HQ* [101] that consists of $30,000$ high-quality celebrity facial images (with attribute annotations) mined and processed from the original CelebA dataset [133]. For the transient attributes database, we randomly split the webcams into 90 for training and 11 for testing. Note that we do not use the webcam information for training our model, and treat the data as an unordered set of images. For the CelebA-HQ dataset, we use the official split.

We compare our method with two baselines:

- CycleGAN [238] (iterative) – After using CycleGAN to learn the generator that transforms images from domain A to domain B, we iteratively apply the generator to the previous output to synthesize a sequence of transformations.
- Fader Networks [121] – A state-of-the-art method that aims to learn disentangled semantic codes for continuous attribute manipulation.

For both baselines, we train the models on our datasets using the provided code online.

7.4.1 Qualitative results

We first demonstrate the effectiveness of our method on several applications that require synthesizing differential transformations (for more results please refer to the supplementary material).

Season progression We again use the transient attributes database, and choose “snow” as the target domain. As shown in Figure 7.2 (top), our method successfully learns to synthesize snow coverage and change lighting in the scene to be more and more characteristic of winter.

Time-of-day progression We show that our method can synthesize the effect of sunset progression from a single image. We train our model on the transient attributes dataset using the “sunrise / sunset” label. As shown in Figure 7.2 (middle), our method is able to synthesize the sunset progression from a single image with realistic and smooth lighting changes.

Facial attribute manipulation We apply our method to the CelebA-HQ [101] dataset for manipulating three facial attributes: “smile”, “young” and “mustache”. As shown in Figure 7.2 (bottom), our method is able to synthesize realistic facial attribute changes.

7.4.2 Qualitative comparison with the baselines

CycleGAN (iterative) As shown in Figure 7.3, since CycleGAN is designed to directly map images from the source domain to the target, it is not able to synthesize realistic and smooth progression even if applied iteratively on the output from the previous step. Notice that the output image falls off the natural image manifold much more quickly than ours.
Fader Networks We show comparison with Fader Networks in Figure 7.4. Our method tends to produce sharper and more realistic details than Fader Networks, which suggests that modeling the full space of transformations using a small latent code representation might cause loss of visual details from the input image, and formulating the problem as image-to-image translation might be more desirable for producing results with high perceptual quality.

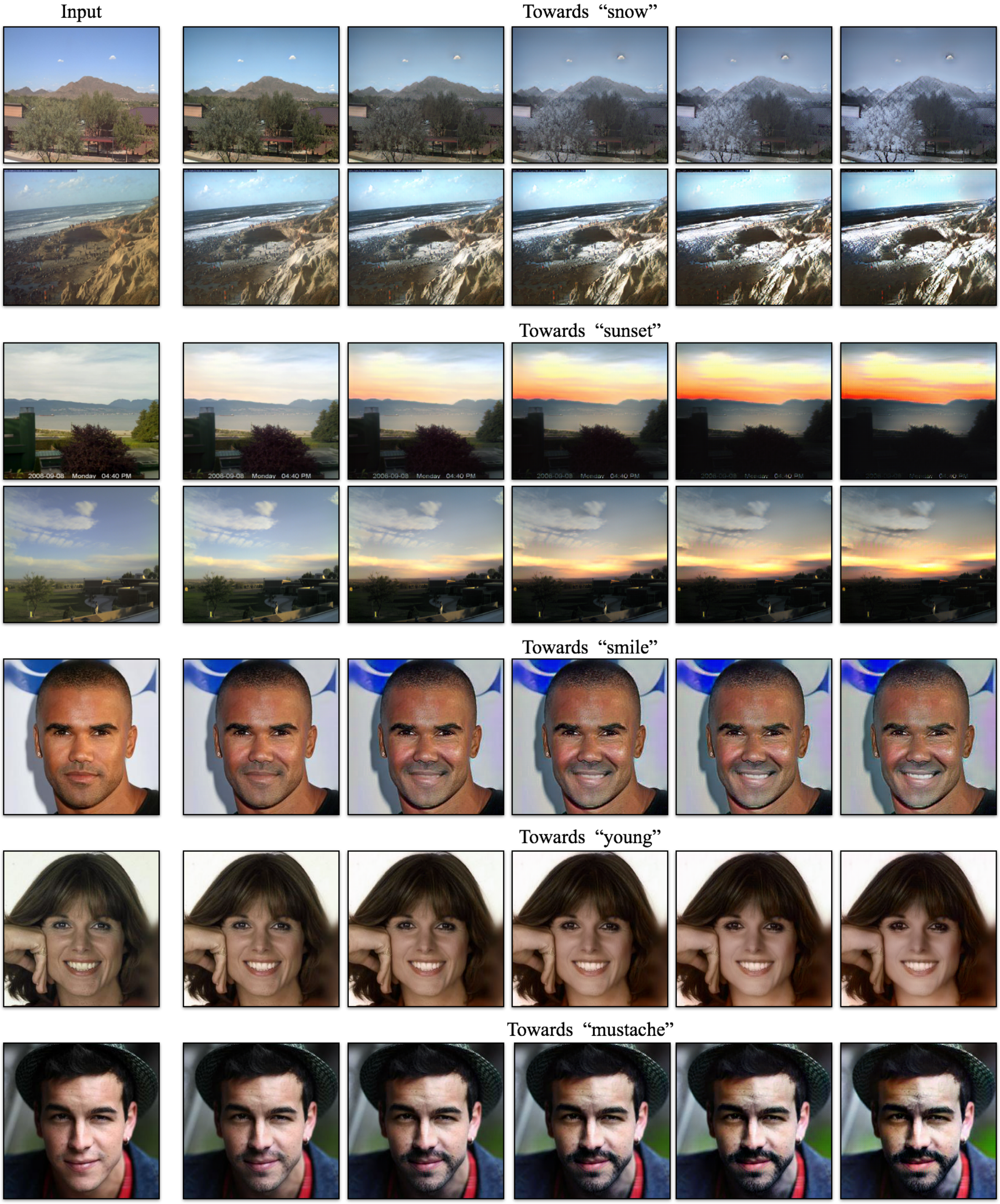


Figure 7.2: Qualitative results of our method. The leftmost column displays the input images. Rows display iterative applications of the learned transformation toward various target domains.

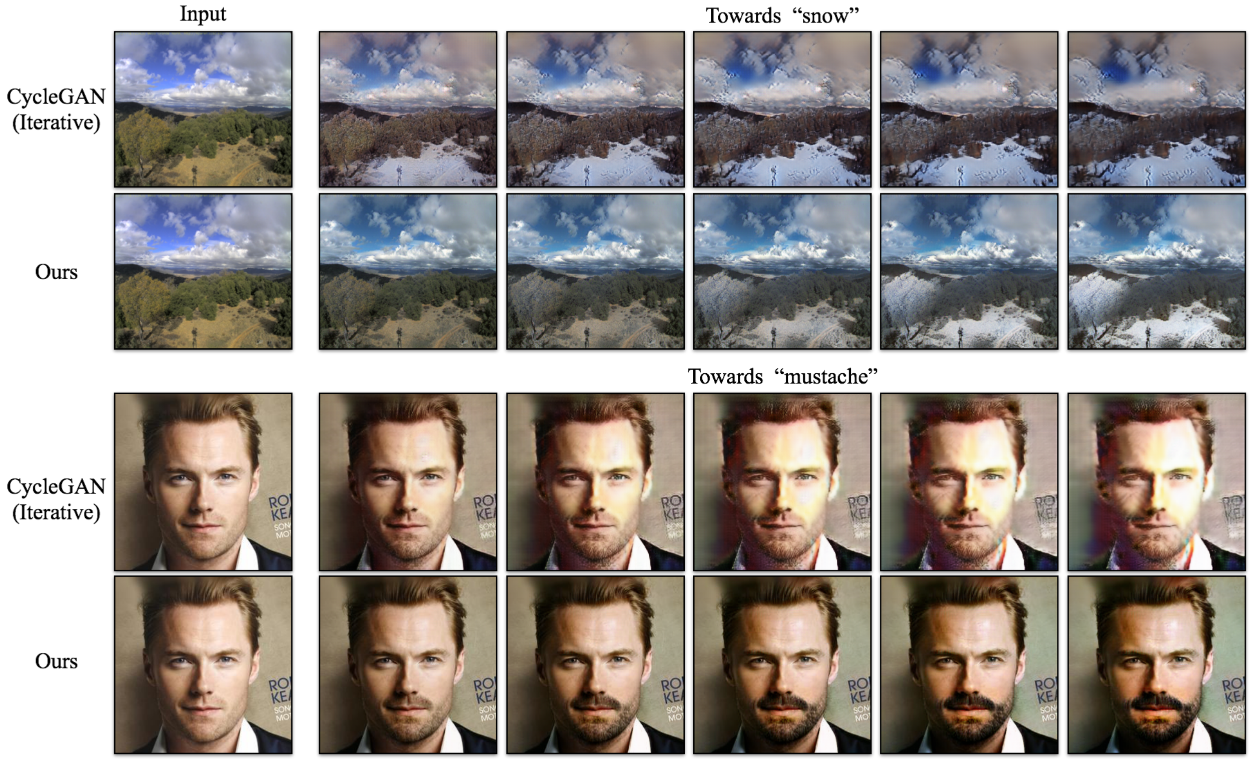


Figure 7.3: Qualitative comparison between the CycleGAN (iterative) baseline and our method. CycleGAN is trained to perform the full translation from source to target domain in one step and multiple applications of the same translation either have no effect or fall off the image manifold. In contrast, our method successfully performs differential changes to images from multiple domains.

7.4.3 Evaluation of transformation consistency

For evaluating the consistency of differential transformations induced by each method, we utilize scores from a domain classifier. For each consecutive pair of frames G_t and G_{t+1} in the synthesized sequence, they are deemed consistent if G_{t+1} is closer to $\mathcal{X}_B$ than G_t, i.e. $\texttt{dist}(G_{t+1}, \mathcal{X}_B) > \texttt{dist}(G_t, \mathcal{X}_B)$, where $\texttt{dist}(\cdot)$ is computed using the domain classifier score. We compute the *percentage of consistent pairs* (PCP) as a measure of the overall consistency among generated sequences. To alleviate bias, we do not use the same classifier that defines the differential loss in our method, but rather train a separate one with a different architecture (ResNet-18 [73]) when evaluating PCP. Specifically, we fine-tune the ResNet-18 network pretrained on ImageNet with our datasets.

We compare the performance of our method with the baselines in Table 7.1. Our method significantly outperforms the baselines in all scene-level transformations ("Snow", "Sunny" and "Sunset"), and two of the facial attribute variations ("Mustache" and "Young"). We perform slightly worse than Fader Networks on "Smile".

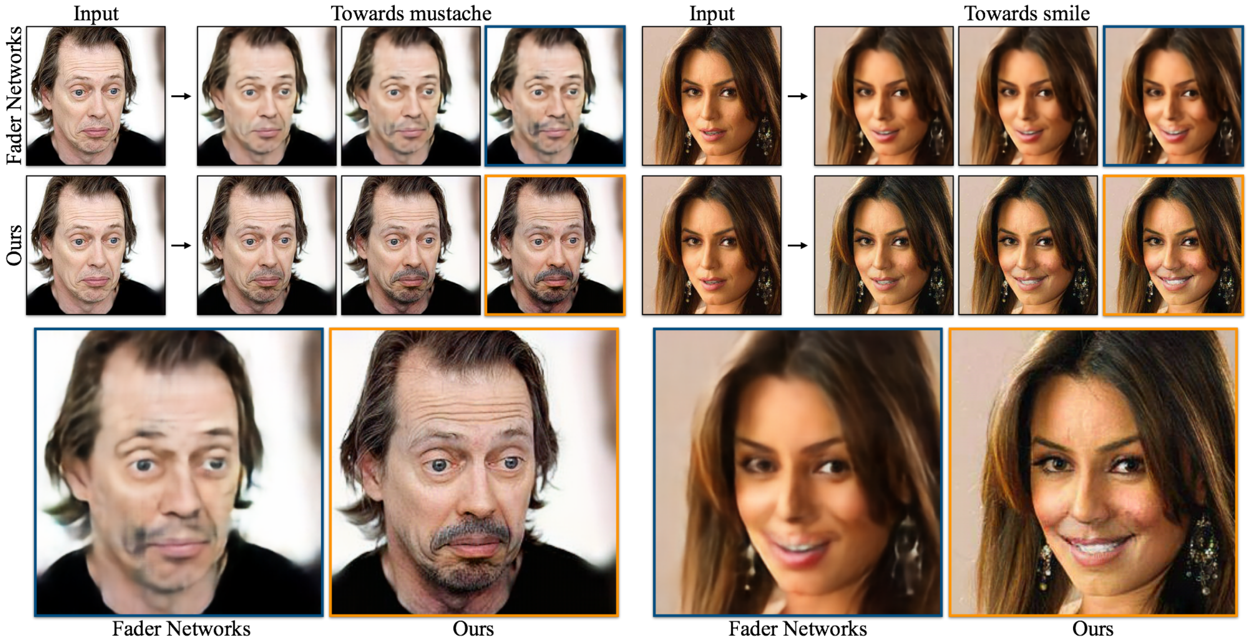


Figure 7.4: Comparison of our method against FaderNetworks [121]. *Top*: Our method generates more realistic progressions than FaderNetworks towards mustache and smile. *Bottom*: The advantage of using direct image translation over working in a bottleneck-bound embedding space becomes clear in high resolution. Our method preserves fine details while FaderNetworks does not.

	Snow	Sunny	Sunset	Mustache	Smile	Young
CycleGAN (iterative)	0.756	0.620	0.645	0.557	0.556	0.741
Fader Networks	0.765	0.574	0.668	0.685	**0.723**	0.738
Ours	**0.870**	**0.718**	**0.781**	**0.817**	0.683	**0.854**

Table 7.1: Evaluating the transformation consistency for different methods with the PCP metric (higher is better).

7.4.4 Evaluation of perceptual quality

To measure the overall quality and realism of the synthesized images, we conducted real vs fake perceptual studies on Amazon Mechanical Turk.

Experimental setup. We followed the same experiment protocol from Zhang *et al.* [227]. Participants were shown a series of pairs of image sequences: one sequence was real time-lapse photos and one sequence was fake photos (generated by our method or a baseline). Participants were asked to click on the sequence they thought was real. Sequences of 7 images of resolution 192 × 192px (downsampled from 256 × 256px in order to fit the monitor screen size) were shown for one second each, and after each pair, participants were given

Method	AMT labelled real (%)
FaderNetworks [121]	18.20
Ours	**41.60**

Table 7.2: Results of real vs. fake perceptual studies on day to sunset progression images synthesized from the transient attributes dataset [119]. Our method produced images that fool participants into thinking they are real more often than the baseline.

unlimited time to respond. We picked 200 input images at random and generated from them a sequence of transformations using each condition. We used 20 real sequences in total that did not start from the same input images as the condition sequences. Each task consisted of 15 pairs of sequences and was performed by 3 different workers. Each worker was only allowed to participate in one experiment. Workers were given a training set of 5 pairs of sequences before the start of the task and were given feedback indicating whether they had correctly identified the real time-lapse sequence in each training pair.

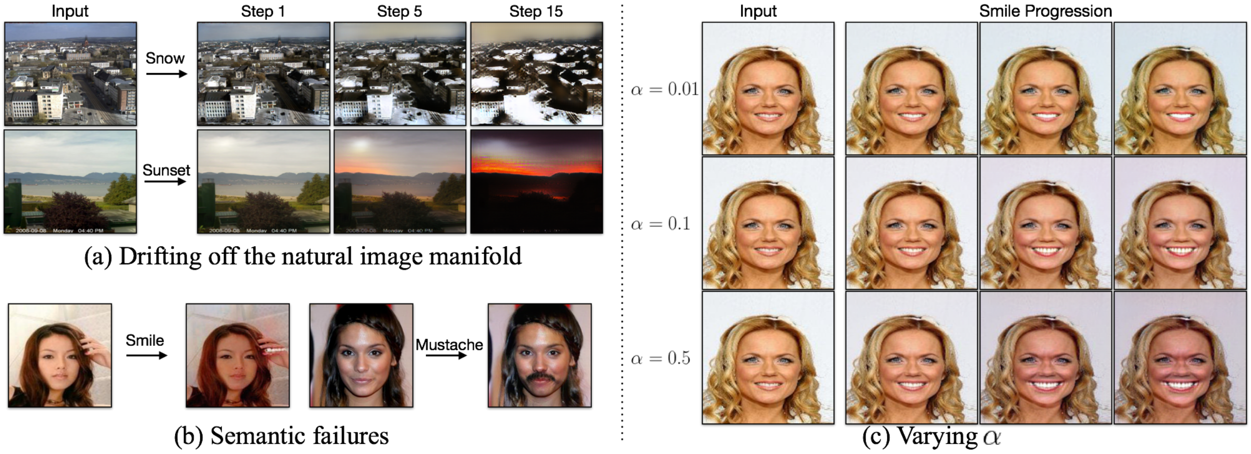
Using this experimental setup, we compared the results of our method with those of Fader Networks [121] on the task of day to sunset progression on the transient attributes images [119]. For each input image we synthesized a 7-step progression from daytime to sunset using each of the methods in comparison.

We assess the quality of each method using the rate at which its output fooled the participants. As shown in Table 7.2, our synthesized results were selected by participants as more "real" than the original paired ground truth real sequence 41.60% of the time. In comparison, the sequences synthesized by Fader Networks [121] fooled the participants only 18.20% of the time. One reason for this difference may be that Fader Networks operates in the embedding space while our method is able to do a direct translation from pixels to pixels, thus preserving more high-frequency details from the original image.

7.4.5 Ablation studies

Effect of the saturation point We demonstrate how changing the value of α (which determines the saturation point in the differential loss) affects the network prediction in Figure 7.5. As expected, larger α results in larger differential for each application of G, and could lead to noticeable artifacts if it becomes too large. Meanwhile, if α is too small, the network is reluctant to change from the input image.

Effect of different loss terms We found that the content loss is helpful in preserving the input identity and improving the perceptual quality of the results when undergoing iterative transformations. The differential loss is necessary for the generator to learn progressive transformations. The adversarial loss is helpful in producing results with high perceptual quality by keeping them on the natural image manifold. Please refer to the supplementary material for more details.



(a) Drifting off the natural image manifold

(b) Semantic failures

(c) Varying α

Figure 7.5: *Left*: Failure cases. *(a)* Repeated applications of G result in images that no longer lie on the manifold of real images. *(b)* Smiling progression produces a toothy smile on the hand instead of the mouth. A mustache is erroneously drawn on a woman. *Right*: The effect of changing the saturation point (determined by α) in the differential loss. If α is too small (top) predictions tend to be identical to the input. If α is too large (bottom) prediction become caricatures and fall off the manifold of natural images.

7.4.6 Failure modes

Figure 7.5 displays several failure modes of our method. Most commonly, the GAN loss may fail to keep the generated images on the manifold of natural images after many applications of G. Additionally, G sometimes fails to modify the input image correctly. The modification may be applied at the wrong spatial location (such as adding a smile on the hand) or fail to perform high-level semantic reasoning and apply transformations where it is not necessarily meaningful (e.g. adding mustache to women).

7.4.7 Additional results

Having demonstrated our method on standard tasks, we now show results on other problems that our differential formulation allows us to easily tackle.

7.4.7.1 Painting animation

We show that our method, trained on the transient attributes dataset of natural images as described in Section 7.4.1, can synthesize at test time progressions from a single image taken from a completely different domain of artistic paintings. Figure 7.6 displays such inferred progressions towards winter and towards sunset. Here our model generalizes despite the fact that it was only trained to produce images from the natural image manifold.

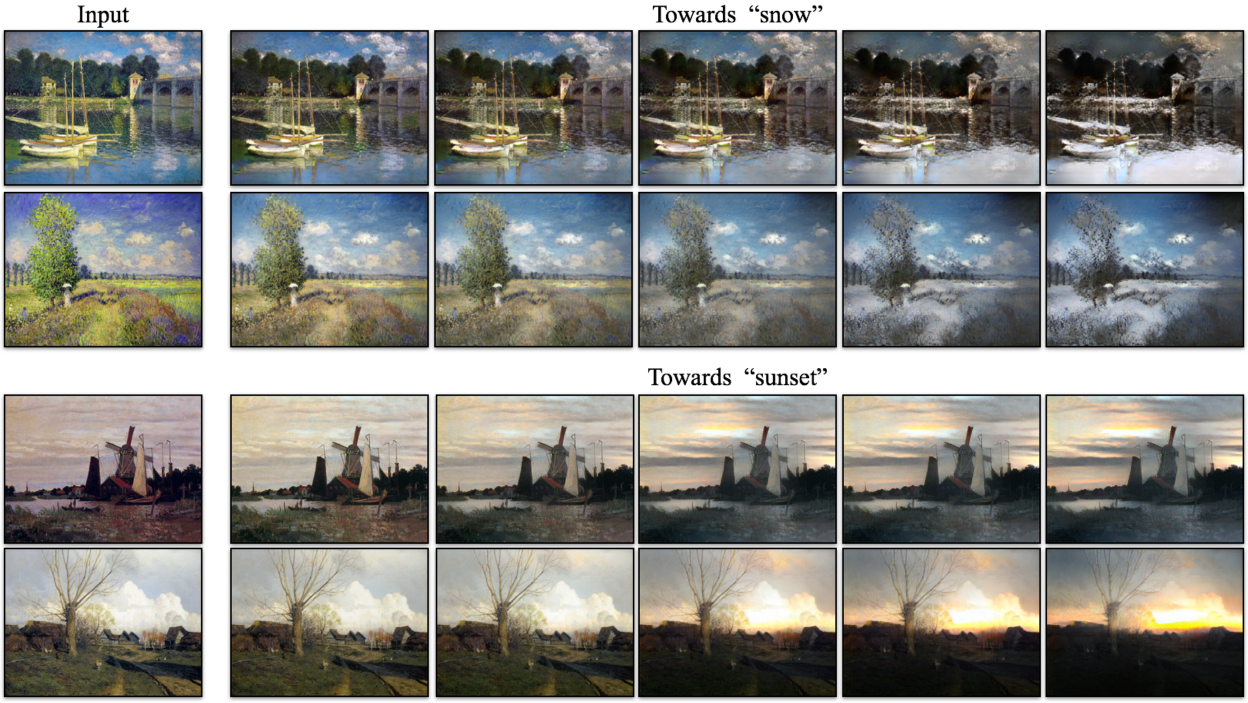


Figure 7.6: Progressions from a single painting using models that were trained on natural images. Our method generalizes to a different domain despite the fact that it was only trained to generate images from the natural image manifold.

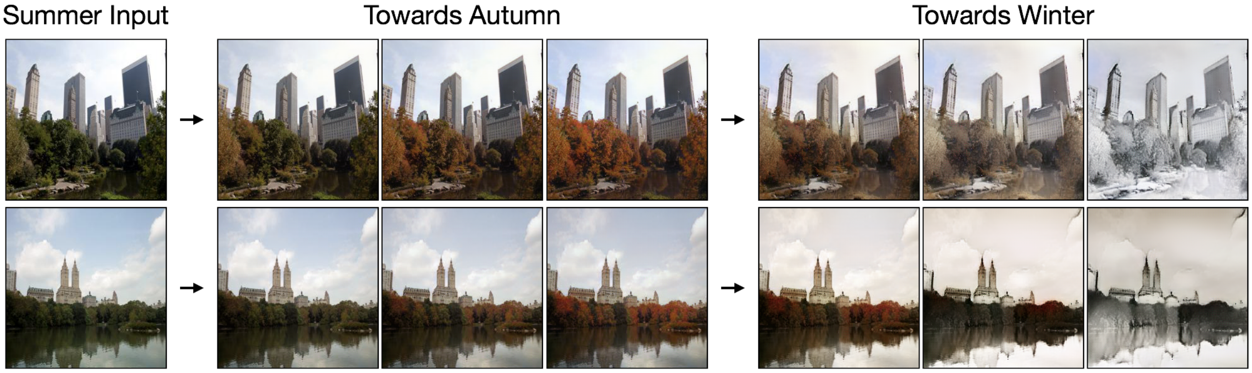


Figure 7.7: Multi-domain season transformation. We use a photograph taken by a Flickr user during the Summer at Central Park, NYC as input. We transform this image to Autumn and then Winter by using the end point of one generated sequence (Summer to Autumn) as the input to the next transformation (Autumn to Winter).

7.4.7.2 Multi-domain transformations

We demonstrate that our method is able to transform smoothly between multiple domains by using the end of a generated output sequence towards domain A as input to a transformation towards domain B. Figure 7.7 shows such a concatenation of transformations from Summer to Autumn to Winter.

This pair of transformations was trained on Flickr images taken between 2006 and 2015 in Central Park in Manhattan that were tagged with time stamps and GPS coordinates. To separate the data into multiple season domains we trained 12-way month classifiers and picked the top $5K$ images for each month based on prediction confidence. We split the months of the year into season domains according to the official season begin and end dates. In order to obtain a clean set of images that are clearly representative of each season, we filtered the set of images to only those which were correctly predicted to have been taken in the corresponding season. We trained separate differential translation models for transitioning between each pair of consecutive seasons. Finally, we generated progressions of multiple seasons by taking the output from one transformation as the input to the next season transformation.

This demonstration is interesting for two reasons. First, we are able to concatenate separately-trained transformations without leaving the natural image manifold. Second, our method can successfully be trained on user-uploaded photographs from Flickr that are only roughly geo-localized and does not require any alignment or special preprocessing of the data between the two source and target domains.

7.5 Discussion

We have demonstrated that our method can learn realistic differential transformations from weak, domain-level supervision. Unlike most related methods, our method does not use a low-dimensional latent embedding, which may allow it to more easily to model high-dimensional outputs. Our work has several limitations. First, the types of transformations that we are able to learn are mostly focused on changes in the appearance of the image, with the scene geometry left largely untouched. We believe this is mainly an artifact of the generator architecture being used. Second, repeated applications of our transformations still may result in the synthesized images "drifting off" the natural image manifold. This may be due to the optimization difficulty in balancing between the GAN and the differential objectives.

Chapter 8

Conclusions

This thesis investigated how to use deep learning for a variety of vision and graphics tasks without requiring direct labeled data for training. For the task of dense semantic correspondence, we use the concept of cycle consistency as the supervisory signal that allows us to train the correspondence network on real images (where we don't have direct labels) by linking them to the synthetic images (where we do have direct labels). For learning depth and ego-motion, we exploit the observation that if both quantities are estimated correctly, they would be able to reconstruct the nearby views in the same video sequence (under mild assumptions about scene motion and disocclusion). Therefore, using the task of view synthesis as supervision, we are able to learn depth and ego-motion estimation from video data without any direct labels. We further demonstrate that using the similar methodology, we could train a deep network to recover a layered scene representation (i.e. multiplane images) from narrow-baseline stereo pairs with supervision from the task of view synthesis. Finally, we show that it is possible to uncover gradual transformations implicit in a collection of static photos without direct labeled data.

Furthermore, we can view each learning methodology above as a specific instance of the broader family of *meta-supervison*, where the supervision is not on what the output is but *how* it should behave. We believe that meta-supervison could be quite effective in other domains too where large-scale direct labels are not available:

Intrinsic image decomposition Given a single image, the goal of intrinsic image decomposition (IID) is to infer the confounding factors of reflectance and shading whose product constitutes the luminance of the given image. While IID is a classic computer vision problem with many important applications, it is by far largely unsolved even with the significant advancement of deep learning mainly due to the difficulty in obtaining ground-truth reflectance and shading at scale to power learning-based methods. We believe that meta-supervision could be the key ingredient for unlocking the success in this domain. One potential direction is to utilize timelapse/webcam type of data, where we could use reflectance constancy as a source for meta-supervision (along with other constraints like luminance reconstruction from the product of reflectance and shading).

Scene dynamics Estimating the dynamics of a scene (i.e. where and how the objects move) is a challenging problem without a feasible mechanism for collecting labeled data in real-world scenes. We are hopeful that a supervision methodology similar to the one we described in Chapter 5 could be effective in learning scene dynamics without requiring direct ground-truth. Some recent works have shown promising results along this line of research [220, 218, 162].

Building the visual memex Rich and interpretable understanding of the visual world arguably requires explicit reasoning of object-object and object-scene relationships. One way to approach this is to build a visual memex graph, with nodes being object/scene instances (instead of categories) and edges representing different types of associations between them, including spatial/semantic context, shared attributes, visual similarity, co-occurrence statistics and many more. With a visual memex at a sufficiently large scale, computational understanding of our visual world could become a walk on the graph that propagates information for the downstream recognition tasks. However, building a scalable visual memex is not trivial, and cannot rely on human annotations due to the exponential growth of the graph size. We hypothesize that meta-supervision is a promising solution since some of the desired properties of the memex (e.g. cycle consistency) could be formulated as objective functions during the memex construction. In some sense, the FlowWeb representation described in Chapter 2 can be viewed as a pixel-level memex, with nodes being pixels and edges being their correspondences, which we hope could inspire future work on building a more general visual memex.

Bibliography

[1] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". In: *OSDI*. 2016.

[2] A.Dosovitskiy, J.T.Springenberg, and T.Brox. "Learning to Generate Chairs with Convolutional Neural Networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition*. 2015.

[3] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver*. `http://ceres-solver.org`. 2016.

[4] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. "Learning to See by Moving". In: *ICCV*. 2015.

[5] Apple. *Portrait mode now available on iPhone 7 Plus with iOS 10.1*. `https://www.apple.com/newsroom/2016/10/portrait-mode-now-available-on-iphone-7-plus-with-ios-101/`. 2016.

[6] Mathieu Aubry et al. "Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 3762–3769. ISBN: 978-1-4799-5118-5. DOI: `10.1109/CVPR.2014.487`. URL: `http://dx.doi.org/10.1109/CVPR.2014.487`.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[8] Connelly Barnes et al. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". In: *SIGGRAPH* 28.3 (2009).

[9] Connelly Barnes et al. "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Transactions on Graphics (TOG)* (2009).

[10] Connelly Barnes et al. "The Generalized PatchMatch Correspondence Algorithm". In: *ECCV*. 2010.

[11] James Bergen et al. "Hierarchical model-based motion estimation". In: *ECCV*. Springer. 1992, pp. 237–252.

[12] Volker Blanz and Thomas Vetter. "A morphable model for the synthesis of 3D faces". In: *SIGGRAPH*. 1999, pp. 187–194.

[13] Chris Buehler et al. “Unstructured lumigraph rendering”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 425–432.

[14] Daniel J Butler et al. “A naturalistic open source movie for optical flow evaluation”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 611–625.

[15] Zhe Cao et al. “Realtime multi-person 2d pose estimation using part affinity fields”. In: *arXiv preprint arXiv:1611.08050* (2016).

[16] Joao Carreira et al. “Virtual View Networks for Object Reconstruction”. In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015.

[17] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[18] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015).

[19] Alexandre Chapiro et al. “Optimizing stereo-to-multiview conversion for autostereoscopic displays”. In: *Computer graphics forum*. 2014.

[20] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[21] Qifeng Chen and Vladlen Koltun. “Photographic image synthesis with cascaded refinement networks”. In: *ICCV*. 2017.

[22] Shenchang Eric Chen and Lance Williams. “View Interpolation for Image Synthesis”. In: *Proc. SIGGRAPH*. 1993.

[23] Tao Chen et al. “3-sweep: Extracting editable objects from a single photo”. In: *ACM Transactions on Graphics (TOG)* (2013).

[24] Xianjie Chen et al. “Detect What You Can: Detecting and Representing Objects using Holistic Models and Body Parts”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[25] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. “NEIL: Extracting Visual Knowledge from Web Data”. In: *ICCV*. 2013.

[26] Brian Cheung et al. “Discovering hidden factors of variation in deep networks”. In: *arXiv preprint arXiv:1412.6583* (2014).

[27] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *CoRR* abs/1511.07289 (2015). arXiv: `1511.07289`. URL: `http://arxiv.org/abs/1511.07289`.

[28] Timothy F Cootes, Gareth J Edwards, Christopher J Taylor, et al. “Active appearance models”. In: *TPAMI* 23.6 (2001), pp. 681–685.

[29] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *CVPR*. 2016.

[30] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE. 2005.

[31] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2005.

[32] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach". In: *Proc. SIGGRAPH*. 1996.

[33] Piotr Didyk et al. "Joint view expansion and filtering for automultiscopic 3D displays". In: *Proc. SIGGRAPH*. 2013.

[34] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *arXiv preprint arXiv:1605.08803* (2016).

[35] Carl Doersch et al. "What makes Paris look like Paris?" In: *SIGGRAPH* 31.4 (2012), p. 101.

[36] Alexey Dosovitskiy and Thomas Brox. "Generating images with perceptual similarity metrics based on deep networks". In: *NIPS*. 2016.

[37] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[38] Alexei A Efros and William T Freeman. "Image quilting for texture synthesis and transfer". In: ACM. 2001, pp. 341–346.

[39] Alexei A Efros and Thomas K Leung. "Texture synthesis by non-parametric sampling". In: *ICCV*. Vol. 2. IEEE. 1999, pp. 1033–1038.

[40] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *NIPS*. 2014.

[41] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 40.3 (2018).

[42] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[43] Alon Faktor and Michal Irani. ""Clustering by Composition"–Unsupervised Discovery of Image Categories". In: *ECCV*. 2012.

[44] Christoph Fehn. "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV". In: *Electronic Imaging 2004*. International Society for Optics and Photonics. 2004, pp. 93–104.

[45] Philipp Fischer et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *ICCV*. 2015.

[46] Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. “Image-based rendering using image-based priors”. In: *IJCV* 63.2 (2005), pp. 141–151.

[47] John Flynn et al. “DeepStereo: Learning to Predict New Views From the World’s Imagery”. In: *CVPR*. 2016.

[48] John Flynn et al. “DeepStereo: Learning to Predict New Views from the World’s Imagery”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[49] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. “SVO: Fast Semi-Direct Monocular Visual Odometry”. In: *ICRA*. 2014.

[50] David F Fouhey et al. “Single image 3D without a single 3D image”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1053–1061.

[51] Kunihiko Fukushima. “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”. In: *Biological Cybernetics* (1980).

[52] Yasutaka Furukawa and Carlos Hernández. “Multi-view stereo: A tutorial”. In: *Foundations and Trends® in Computer Graphics and Vision* 9 (2015).

[53] Yasutaka Furukawa et al. “Towards internet-scale multi-view stereo”. In: *CVPR*. IEEE. 2010, pp. 1434–1441.

[54] Matheus Gadelha, Subhransu Maji, and Rui Wang. “3D Shape Induction from 2D Views of Multiple Objects”. In: *arXiv preprint arXiv:1612.05872* (2016).

[55] Ravi Garg et al. “Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue”. In: *ECCV*. 2016.

[56] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *CVPR* (2016).

[57] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3354–3361.

[58] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CVPR*. 2014.

[59] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. “Unsupervised Monocular Depth Estimation with Left-Right Consistency”. In: *CVPR*. 2017.

[60] Ian Goodfellow et al. “Generative adversarial nets”. In: *NIPS*. 2014.

[61] Google. *Introducing VR180 cameras*. `https://vr.google.com/vr180/`. 2017.

[62] Google. *Portrait mode on the Pixel 2 and Pixel 2 XL smartphones*. `https://research.googleblog.com/2017/10/portrait-mode-on-pixel-2-and-pixel-2-xl.html`. 2017.

[63] Ross Goroshin et al. “Unsupervised learning of spatiotemporally coherent metrics”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4086–4093.

[64] Steven J. Gortler et al. “The Lumigraph”. In: *Proc. SIGGRAPH*. 1996.

[65] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. “Densepose: Dense human pose estimation in the wild”. In: *arXiv preprint arXiv:1802.00434* (2018).

[66] Saurabh Gupta et al. “Aligning 3D Models to RGB-D Images of Cluttered Scenes”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2015.

[67] Hyowon Ha et al. “High-quality Depth from Uncalibrated Small Motion Clip”. In: *CVPR*. 2016.

[68] Yoav HaCohen et al. “Optimizing Color Consistency in Photo Collections”. In: *SIGGRAPH* 32.4 (2013), 85:1–85:9.

[69] Xufeng Han et al. “MatchNet: Unifying feature and metric learning for patch-based matching”. In: *CVPR*. 2015, pp. 3279–3286.

[70] Ankur Handa et al. “gvnn: Neural Network Library for Geometric Computer Vision”. In: *arXiv preprint arXiv:1607.07405* (2016).

[71] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[72] Samuel W. Hasinoff et al. “Burst photography for high dynamic range and low-light imaging on mobile cameras”. In: *Proc. SIGGRAPH Asia*. 2016.

[73] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[74] Kaiming He et al. “Mask r-cnn”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2980–2988.

[75] Mingming He et al. “Neural Color Transfer between Images”. In: *arXiv preprint arXiv:1710.00756* (2017).

[76] Kyle Heath et al. “Image webs: Computing and exploiting connectivity in image collections.” In: *CVPR*. 2010.

[77] Peter Hedman et al. “Casual 3D Photography”. In: *Proc. SIGGRAPH Asia*. 2017.

[78] Aaron Hertzmann et al. “Image analogies”. In: ACM. 2001, pp. 327–340.

[79] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. “Transforming auto-encoders”. In: *Artificial Neural Networks and Machine Learning–ICANN*. 2011.

[80] Derek Hoiem, Alexei A. Efros, and Martial Hebert. “Automatic Photo Pop-up”. In: 2005.

[81] Derek Hoiem, Alexei A Efros, and Martial Hebert. “Automatic photo pop-up”. In: *ACM transactions on graphics (TOG)* (2005).

[82] Michael Holroyd et al. “Computing and fabricating multilayer models”. In: *Proc. SIGGRAPH Asia*. 2011.

[83] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. “Tour into the picture: using a spidery mesh interface to make animation from a single image”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997.

[84] Gary B. Huang, Vidit Jain, and Erik Learned-Miller. “Unsupervised Joint Alignment of Complex Images”. In: *ICCV*. 2007.

[85] Q. Huang and L. Guibas. “Consistent Shape Maps via Semidefinite Programming”. In: *SGP*. 2013.

[86] Qixing Huang, Hai Wang, and Vladlen Koltun. “Single-view reconstruction via joint analysis of image and shape collections”. In: *ACM Trans. Graph.* 34.4 (2015), p. 87. DOI: 10.1145/2766890. URL: http://doi.acm.org/10.1145/2766890.

[87] Qi-Xing Huang and Leonidas Guibas. “Consistent shape maps via semidefinite programming”. In: *Computer Graphics Forum* 32.5 (2013), pp. 177–186.

[88] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*. Vol. 46. John Wiley & Sons, 2004.

[89] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: 2015.

[90] Michal Irani and P Anandan. “About direct methods”. In: *International Workshop on Vision Algorithms*. Springer. 1999, pp. 267–277.

[91] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CVPR* (2017).

[92] Max Jaderberg et al. “Spatial Transformer Networks”. In: *NIPS*. 2015.

[93] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks”. In: *Advances in Neural Information Processing Systems*. 2015.

[94] D. Jayaraman and K. Grauman. “Learning image representations tied to egomotion”. In: *IEEE International Conference on Computer Vision*. 2015.

[95] Dinesh Jayaraman and Kristen Grauman. “Learning image representations tied to ego-motion”. In: *ICCV*. 2015.

[96] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *arXiv preprint arXiv:1408.5093* (2014).

[97] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *ECCV*. 2016.

[98] Nebojsa Jojic, Brendan J Frey, and Anitha Kannan. “Epitomic analysis of appearance and shape”. In: *IEEE International Conference on Computer Vision*. 2003.

[99] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[100] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. "Learning-Based View Synthesis for Light Field Cameras". In: *Proc. SIGGRAPH Asia*. 2016.

[101] Tero Karras et al. "Progressive growing of GANs for improved quality, stability, and variation". In: *ICLR*. 2018.

[102] Kevin Karsch, Ce Liu, and Sing Bing Kang. "Depth transfer: Depth extraction from video using non-parametric sampling". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11 (2014), pp. 2144–2158.

[103] Petr Kellnhofer et al. "3DTV at Home: Eulerian-Lagrangian Stereo-to-Multiview Conversion". In: *Proc. SIGGRAPH*. 2017.

[104] Ira Kemelmacher-Shlizerman and Steve Seitz. "Collection flow". In: *CVPR*. 2012.

[105] Ira Kemelmacher-Shlizerman and Steven M Seitz. "Face reconstruction in the wild". In: *ICCV*. 2011.

[106] Ira Kemelmacher-Shlizerman et al. "Exploring photobios". In: *SIGGRAPH* 30.4 (2011), p. 61.

[107] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A convolutional network for real-time 6-DOF camera relocalization". In: *ICCV*. 2015, pp. 2938–2946.

[108] Alex Kendall et al. "End-to-End Learning of Geometry and Context for Deep Stereo Regression". In: *arXiv preprint arXiv:1703.04309* (2017).

[109] Natasha Kholgade et al. "3d object manipulation in a single photograph using stock 3d models". In: *ACM Transactions on Graphics (TOG)* (2014).

[110] Jaechul Kim et al. "Deformable Spatial Pyramid Matching for Fast Dense Correspondences". In: *CVPR*. 2013.

[111] Taeksoo Kim et al. "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks". In: *ICML*. 2017.

[112] Taeksoo Kim et al. "Learning to discover cross-domain relations with generative adversarial networks". In: *arXiv preprint arXiv:1703.05192* (2017).

[113] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: (2014).

[114] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[115] Philipp Krähenbühl and Vladlen Koltun. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials.* NIPS, 2011.

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

[117] Tejas D Kulkarni et al. "Deep Convolutional Inverse Graphics Network". In: *NIPS*. 2015.

[118] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. "Semi-Supervised Deep Learning for Monocular Depth Map Prediction". In: *arXiv preprint arXiv:1702.02706* (2017).

[119] Pierre-Yves Laffont et al. "Transient attributes for high-level understanding and editing of outdoor scenes". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 149.

[120] Iro Laina et al. "Deeper depth prediction with fully convolutional residual networks". In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, pp. 239–248.

[121] Guillaume Lample et al. "Fader Networks: Manipulating Images by Sliding Attributes". In: *CoRR* abs/1706.00409 (2017). arXiv: `1706.00409`. URL: `http://arxiv.org/abs/1706.00409`.

[122] Erik Learned-Miller. "Data Driven Image Models through Continuous Joint Alignment". In: *TPAMI* 28.2 (2005), pp. 236–250.

[123] Yang LeCun et al. "Backpropagation applied to hand-written zip code recognition". In: *Neural Computation.* 1989.

[124] Y. J. Lee and K. Grauman. "Collect-Cut: Segmentation with Top-Down Cues Discovered in Multi-Object Images". In: *CVPR*. 2010.

[125] Marc Levoy and Pat Hanrahan. "Light Field Rendering". In: *Proc. SIGGRAPH.* 1996.

[126] Marc Levoy and Pat Hanrahan. "Light field rendering". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 31–42.

[127] Jing Liao et al. "Visual attribute transfer through deep image analogy". In: *arXiv preprint arXiv:1705.01088* (2017).

[128] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[129] Ce Liu, Jenny Yuen, and Antonio Torralba. "SIFT Flow: Dense Correspondence across Scenes and Its Applications." In: *TPAMI* 33.5 (2011), pp. 978–994.

[130] Fayao Liu et al. "Learning depth from single monocular images using deep convolutional neural fields". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2016), pp. 2024–2039.

[131] Miaomiao Liu, Mathieu Salzmann, and Xuming He. "Discrete-continuous depth estimation from a single image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 716–723.

[132] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 700–708.

[133] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[134] Ziwei Liu et al. "Video Frame Synthesis Using Deep Voxel Flow". In: *ICCV*. 2017.

[135] Jonathan L Long, Ning Zhang, and Trevor Darrell. "Do Convnets Learn Correspondence?" In: *Advances in Neural Information Processing Systems*. 2014, pp. 1601–1609.

[136] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *CVPR*. 2015, pp. 3431–3440.

[137] Matthew M Loper and Michael J Black. "OpenDR: An approximate differentiable renderer". In: *ECCV*. Springer. 2014, pp. 154–169.

[138] Yin Lou, Noah Snavely, and Johannes Gehrke. "MatchMiner: Efficient Spanning Structure Mining in Large Image Collections". In: *ECCV*. 2012.

[139] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[140] Fujun Luan et al. "Deep photo style transfer". In: *CoRR, abs/1703.07511* (2017).

[141] Lytro. *Lytro*. `https://www.lytro.com/`. 2018.

[142] Tomasz Malisiewicz and Alexei A. Efros. "Beyond Categories: The Visual Memex Model for Reasoning About Object Relationships". In: *NIPS*. 2009.

[143] Xudong Mao et al. "Multi-class Generative Adversarial Networks with the L2 Loss Function". In: *arXiv preprint arXiv:1611.04076* (2016).

[144] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *ICLR* (2016).

[145] Iain Matthews and Simon Baker. "Active appearance models revisited". In: *IJCV* 60.2 (2004), pp. 135–164.

[146] Nikolaus Mayer et al. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4040–4048.

[147] Xu Miao and Rajesh P. N. Rao. "Learning the Lie Groups of Visual Invariance". In: *Neural Computation* (2007).

[148] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. "Shuffle and learn: unsupervised learning using temporal order verification". In: *European Conference on Computer Vision*. Springer. 2016, pp. 527–544.

[149] Hossein Mobahi, Ce Liu, and William T. Freeman. "A Compositional Model for Low-Dimensional Image Set Representation". In: *CVPR*. 2014.

[150] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015).

[151] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[152] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *ICCV*. IEEE. 2011, pp. 2320–2327.

[153] A. Nguyen et al. "An optimization approach to improving collections of shape maps". In: *SGP*. 2011.

[154] Byong Mok Oh et al. "Image-based modeling and photo editing". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001.

[155] Deepak Pathak et al. "Learning Features by Watching Objects Move". In: *CVPR*. 2017.

[156] Xingchao Peng et al. "Learning Deep Object Detectors from 3D Models". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1278–1286.

[157] Yigang Peng et al. "RASL: Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images". In: *TPAMI* 34.11 (Nov. 2012).

[158] Eric Penner and Li Zhang. "Soft 3D Reconstruction for View Synthesis". In: *Proc. SIGGRAPH Asia*. 2017.

[159] Guim Perarnau et al. "Invertible Conditional GANs for image editing". In: *NIPS Workshop on Adversarial Training*. 2016.

[160] Thomas Porter and Tom Duff. "Compositing Digital Images". In: *Proc. SIGGRAPH*. 1984.

[161] René Ranftl et al. "Dense monocular depth estimation in complex dynamic scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4058–4066.

[162] Anurag Ranjan et al. "Adversarial Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation". In: *arXiv preprint arXiv:1805.09806* (2018).

[163] Konstantinos Rematas et al. "Novel Views of Objects from a Single Image". In: *arXiv preprint arXiv:1602.00328* (2015).

[164] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[165] Jerome Revaud et al. "EpicFlow: Edge-preserving interpolation of correspondences for optical flow". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[166] Danilo Jimenez Rezende et al. "Unsupervised learning of 3d structure from images". In: *Advances In Neural Information Processing Systems*. 2016, pp. 4997–5005.

[167] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for benchmarks". In: *International conference on computer vision (ICCV)*. Vol. 2. 2017.

[168] Christian Riechert et al. “Fully automatic stereo-to-multiview conversion in autostereoscopic displays”. In: *The Best of IET and IBC* 4 (Sept. 2012).

[169] Sam T Roweis and Lawrence K Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *science* 290.5500 (2000), pp. 2323–2326.

[170] M. Rubinstein et al. “Unsupervised Joint Object Discovery and Segmentation in Internet Images”. In: *CVPR*. 2013.

[171] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *IJCV* 115.3 (2015), pp. 211–252.

[172] Aswin C. Sankaranarayanan et al. “Go with the Flow: Optical Flow-based Transport Operators for Image Manifolds”. In: *Annual Allerton Conference on Communication, Control, and Computing*. 2011.

[173] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. “Make3D: Learning 3D Scene Structure from a Single Still Image”. In: *TPAMI* 31.5 (May 2009), pp. 824–840.

[174] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *CVPR*. 2016.

[175] Steven M Seitz and Charles R Dyer. “View morphing”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 21–30.

[176] Jonathan Shade et al. “Layered depth images”. In: *Proc. SIGGRAPH*. 1998.

[177] *ShapeNet*. `http://www.shapenet.org`.

[178] Roger N. Shepard and Jacqueline Metzler. “Mental Rotation of Three-Dimensional Objects”. In: *Science* (1971).

[179] Yichang Shih et al. “Data-driven hallucination of different times of day from a single outdoor photo”. In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), p. 200.

[180] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[181] Shuran Song and Jianxiong Xiao. “Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images”. In: *CVPR*. 2016.

[182] Pratul P. Srinivasan et al. “Learning to Synthesize a 4D RGBD Light Field from a Single Image”. In: *ICCV*. 2017.

[183] Hao Su et al. “3D-Assisted Image Feature Synthesis for Novel Views of an Object”. In: *International Conference on Computer Vision*. 2015.

[184] Hao Su et al. “Estimating Image Depth Using Shape Collections”. In: *Transactions on Graphics (Special issue of SIGGRAPH 2014)* (2014).

[185] Richard Szeliski. “Prediction error as a quality metric for motion and stereo”. In: *ICCV*. Vol. 2. IEEE. 1999, pp. 781–788.

[186] Yaniv Taigman, Adam Polyak, and Lior Wolf. "Unsupervised Cross-Domain Image Generation". In: *arXiv preprint arXiv:1611.02200* (2016).

[187] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3d models from single images with a convolutional network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337.

[188] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Single-view to Multi-view: Reconstructing Unseen Views with a Convolutional Network". In: *arXiv preprint arXiv:1511.06702* (2015).

[189] Joshua B Tenenbaum, Vin De Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *Science* ().

[190] Antonio Torralba. *http://people.csail.mit.edu/torralba/gallery/*. 2001. URL: `http://people.csail.mit.edu/torralba/gallery/`.

[191] Shubham Tulsiani et al. "Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency". In: *CVPR*. 2017.

[192] G. Tzimiropoulos and M. Pantic. "Optimization problems for fast AAM fitting in-the-wild". In: *ICCV*. 2013.

[193] Dmitry Ulyanov et al. "Texture networks: Feed-forward synthesis of textures and stylized images". In: *Int. Conf. on Machine Learning (ICML)*. 2016.

[194] Benjamin Ummenhofer et al. "DeMoN: Depth and Motion Network for Learning Monocular Stereo". In: *arXiv preprint arXiv:1612.02401* (2016).

[195] Paul Upchurch et al. "Deep feature interpolation for image content changes". In: *arXiv preprint arXiv:1611.05507* (2016).

[196] Sara Vicente et al. "Reconstructing PASCAL VOC". In: *CVPR*. 2014.

[197] Sudheendra Vijayanarasimhan et al. "SfM-Net: Learning of Structure and Motion from Video". In: *arXiv preprint* (2017).

[198] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Generating Videos with Scene Dynamics". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 613–621. URL: `http://papers.nips.cc/paper/6194-generating-videos-with-scene-dynamics.pdf`.

[199] F. Wang, Q. Huang, and L. Guibas. "Image Co-Segmentation via Consistent Functional Maps Image Co-Segmentation via Consistent Functional Maps". In: *ICCV*. 2013.

[200] F. Wang et al. "Unsupervised Multi-Class Joint Image Segmentation". In: *CVPR*. 2014.

[201] John YA Wang and Edward H Adelson. "Representing moving images with layers". In: *IEEE Trans. on Image Processing* 3.5 (1994).

[202] Xiaolong Wang and Abhinav Gupta. "Unsupervised learning of visual representations using videos". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2794–2802.

[203] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[204] Sven Wanner, Stephan Meister, and Bastian Goldluecke. "Datasets and benchmarks for densely sampled 4d light fields". In: *VMV*. 2013.

[205] Philippe Weinzaepfel et al. "Deepflow: Large displacement optical flow with deep matching". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013.

[206] G. Wetzstein et al. "Layered 3D: Tomographic Image Synthesis for Attenuation-based Light Field and High Dynamic Range Displays". In: *Proc. SIGGRAPH*. 2011.

[207] Wikipedia. *Multiplane camera*. `https://en.wikipedia.org/wiki/Multiplane_camera`. 2017.

[208] K. Wilson and N. Snavely. "Network Principles for SfM: Disambiguating Repeated Structures with Local Context". In: *ICCV*. 2013.

[209] John Wright et al. "Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices via Convex Optimization". In: *NIPS*. 2009.

[210] Changchang Wu. *VisualSFM: A visual structure from motion system*. 2011.

[211] Zhirong Wu et al. "3D ShapeNets: A Deep Representation for Volumetric Shape Modeling". In: *CVPR*. 2015.

[212] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild". In: *WACV*. 2014.

[213] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. "Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks". In: *ECCV*. 2016.

[214] Xinchen Yan et al. "Attribute2Image: Conditional Image Generation from Visual Attributes". In: *arXiv preprint arXiv:1512.00570* (2015).

[215] Xinchen Yan et al. "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1696–1704.

[216] Jimei Yang et al. "Weakly-supervised Disentangling with Recurrent Transformations for 3D View Synthesis". In: *NIPS*. 2015.

[217] Yi Yang and Deva Ramanan. "Articulated human detection with flexible mixtures of parts". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.12 (2013), pp. 2878–2890.

[218] Zhenheng Yang et al. "Every Pixel Counts: Unsupervised Geometry Learning with Holistic 3D Motion Understanding". In: *arXiv preprint arXiv:1806.10556* (2018).

[219] Zili Yi, Hao Zhang, Ping Tan Gong, et al. "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation". In: *arXiv preprint arXiv:1704.02510* (2017).

[220] Zhichao Yin and Jianping Shi. "GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2018.

[221] Fisher Yu and David Gallup. "3D Reconstruction from Accidental Motion". In: *CVPR*. 2014.

[222] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *ICLR*. 2016.

[223] Christopher Zach, Manfred Klopschitz, and Manfred Pollefeys. "Disambiguating visual relations using loop constraints." In: *CVPR*. 2010.

[224] Jure Zbontar and Yann LeCun. "Stereo matching by training a convolutional neural network to compare image patches". In: *Journal of Machine Learning Research* 17.1-32 (2016), p. 2.

[225] Jure Žbontar and Yann LeCun. "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches". In: *arXiv preprint arXiv:1510.05970* (2015).

[226] Li Zhang et al. "Single-view modelling of free-form scenes". In: *The Journal of Visualization and Computer Animation* (2002).

[227] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful Image Colorization". In: *ECCV*. 2016.

[228] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Networks as a Perceptual Metric". In: *CVPR*. 2018.

[229] Zhoutong Zhang, Yebin Liu, and Qionghai Dai. "Light field from micro-baseline image pair". In: *CVPR*. 2015.

[230] Youyi Zheng et al. "Interactive images: cuboid proxies for smart image manipulation." In: *ACM Transactions on Graphics (TOG)* (2012).

[231] Tinghui Zhou et al. "FlowWeb: Joint Image Set Alignment by Weaving Consistent, Pixel-wise Correspondences". In: *CVPR*. 2015.

[232] Tinghui Zhou et al. "Learning dense correspondence via 3d-guided cycle consistency". In: *CVPR*. 2016.

[233] Tinghui Zhou et al. "Stereo Magnification: Learning view synthesis using multiplane images". In: *SIGGRAPH*. 2018.

[234] Tinghui Zhou et al. "Unsupervised learning of depth and ego-motion from video". In: *CVPR*. 2017.

[235] Tinghui Zhou et al. "View synthesis by appearance flow". In: *ECCV*. 2016.

[236] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. "Multi-Image Matching via Fast Alternating Minimization". In: *ICCV*. 2015.

[237] Yipin Zhou and Tamara L Berg. “Learning Temporal Transformations from Time-Lapse Videos”. In: *ECCV*. 2016.

[238] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *ICCV*. 2017.

[239] C. Lawrence Zitnick et al. “High-quality Video View Interpolation Using a Layered Representation”. In: *Proc. SIGGRAPH*. 2004.