Consistent Depth of Moving Objects in Video

ZHOUTONG ZHANG, MIT CSAIL, Google Research
FORRESTER COLE, Google Research
RICHARD TUCKER, Google Research
WILLIAM T. FREEMAN, MIT CSAIL, Google Research
TALI DEKEL, Google Research, Weizmann Institute of Science

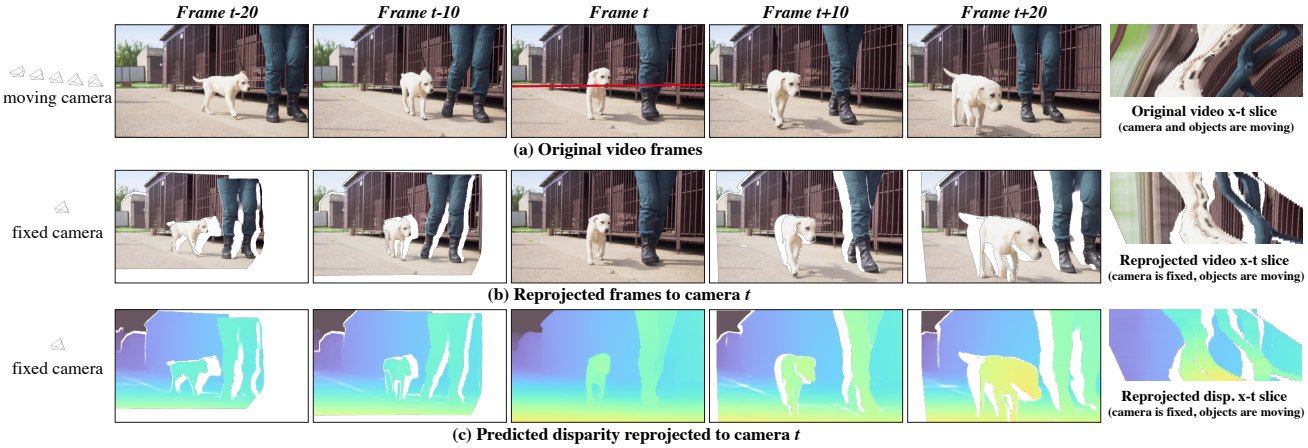

Fig. 1. Our method estimates *geometrically and temporally consistent* depth from a general video containing fast-moving objects and camera motion. The input video (a) contains a continuous camera dolly motion following the moving person and puppy. This is a difficult case for depth estimation due to the correlated motion between camera and subject. The video is shown reprojected into the camera at frame t using our predicted depth (b), with disparity maps of the re-projection shown below (c). On the right: $x-t$ slices for the horizontal line marked in red on frame t in (a). The slice of the original video (top) shows both camera and objects' motion (slanted lines in the background, twisted lines in the foreground). The slice of the re-projected frames (middle) shows the camera fixed relative to the background (vertical lines), and foreground objects moving relative to the camera (twisted lines).

We present a method to estimate depth of a dynamic scene, containing arbitrary moving objects, from an ordinary video captured with a moving camera. We seek a *geometrically and temporally consistent* solution to this underconstrained problem: the depth predictions of corresponding points across frames should induce plausible, smooth motion in 3D. We formulate this objective in a new test-time training framework where a depth-prediction CNN is trained in tandem with an auxiliary scene-flow prediction MLP over the entire input video. By recursively unrolling the scene-flow prediction MLP over varying time steps, we compute both short-range scene flow to impose local smooth motion priors directly in 3D, and long-range scene flow to impose multi-view consistency constraints with wide baselines. We demonstrate accurate and temporally coherent results on a variety of challenging videos containing diverse moving objects (pets, people, cars), as well as camera motion. Our depth maps give rise to a number of depth-and-motion aware video editing effects such as object and lighting insertion.

Authors' addresses: Zhoutong Zhang, MIT CSAIL, Google Research, ztzhang@mit.edu; Forrester Cole, Google Research, fcole@google.com; Richard Tucker, Google Research, richardt@google.com; William T. Freeman, MIT CSAIL, Google Research, billf@mit.edu; Tali Dekel, Google Research, Weizmann Institute of Science, tali.dekel@weizmann.ac.il.

0730-0301/2021/8-ART148
https://doi.org/10.1145/3450626.3459871

CCS Concepts: • **Computing methodologies** → **Computational photography**; **Shape inference**.

Additional Key Words and Phrases: depth, dynamic scenes, scene flow, video, moving objects

ACM Reference Format:
Zhoutong Zhang, Forrester Cole, Richard Tucker, William T. Freeman, and Tali Dekel. 2021. Consistent Depth of Moving Objects in Video. *ACM Trans. Graph.* 40, 4, Article 148 (August 2021), 12 pages. https://doi.org/10.1145/3450626.3459871

1 INTRODUCTION

Estimating the geometry of moving objects from *ordinary videos*—where both the camera and objects in the scene are moving—is an open, underconstrained problem. For any video of a moving object, there is a space of solutions that satisfy the visual evidence: the object could be far away and moving fast, or close and moving slowly. Thus, prior knowledge about objects' motion or shape is necessary to estimate geometry of moving objects in video.

Learning geometric priors directly from data has proven to be a promising approach that facilitates geometry estimation in unconstrained setups such as predicting depth from a single RGB image (RGB-to-depth), or from a video where both the camera and the people in the scene are moving. In such cases, epipolar geometry

constraints break and triangulation-based methods are not applicable, so the solution is determined by learned geometric priors. Such geometric priors are typically learned from a large dataset containing visual data (images/videos), and their corresponding depth maps, which are used as supervision during training. While powerful, a *purely data-driven* approach has two main weaknesses in the case of arbitrary moving objects in a monocular video: (i) *scarcity of training data*—video datasets of dynamic scenes with ground truth depth are still limited and only available for specific class of objects (e.g., moving people [Li et al. 2019, 2020a]), (ii) *temporal consistency*—feed-forward models for videos consider a fixed, short-range window of inputs (often one or two frames), resulting in inconsistent geometry and flickering depth over time.

Instead of relying entirely on data, we take a hybrid approach and initialize a solution from geometric priors learned from a large dataset, then optimize the solution to satisfy space-time geometric constraints. This approach is inspired by the recent work of [Luo et al. 2020], which finetunes a monocular depth prediction model for a video by assuming a static scene and requiring depth estimates for corresponding pixels to be consistent across frames. Our method generalizes this approach to videos of *dynamic scenes* by explicitly modeling *scene flow*, i.e., the 3D motion of objects in the scene.

We seek a geometrically and temporally consistent solution: the depth and scene flow at corresponding points should to be consistent over frames, and the scene flow should change smoothly in time. Our key contribution is a formulation of this ill-posed problem via the optimization of two networks: a *depth-prediction* CNN that takes an RGB frame as input and outputs a dense depth map; and an auxiliary *scene-flow-prediction* multi-layer perceptron (MLP) that takes a 3D point as input and outputs its 3D velocity vector. The depth network is first initialized using a data-driven prior (pretrained weights), and then finetuned in tandem with the scene-flow network for a given input video, using a smooth-motion prior and multi-view consistency losses. As in [Luo et al. 2020], we compute camera poses and optical flow between pairs of frames in a pre-processing step and use them as input.

Our scene-flow MLP plays two important roles: (1) it provides an explicit, 3D representation of scene flow that aggregates information over space and time through the shared weights of the network, and produces plausible flow in cases where an analytic solution derived from depth and optical flow is unstable (e.g., nearly parallel rays between two points, as shown in Section 5.1.). (2) it provides a mechanism to form scene flow estimates over varying time steps. As demonstrated by stereo and MVS methods, wide baseline correspondences are required to accurately estimate depth, but smooth-motion priors must be evaluated locally. The scene-flow MLP, which maps a 3D point at time t to its velocity vector (3D offset between t and $t + 1$), can be evaluated recursively by unrolling. This allows us to compute long-range scene flow to apply multi-view consistency losses between distant frames, and short-range scene flow to apply a smooth motion prior.

We demonstrate detailed, accurate, and temporally-consistent depth maps produced from ordinary videos with multiple, non-rigid, rapidly-moving objects (Fig. 1). We evaluate and ablate our method quantitatively and show 40–45% reduction in L_1 relative error vs. single-view depth prediction, and show various depth and motion aware editing effects in videos made possible by our depth and scene flow estimates.

2 RELATED WORK

Learning-based depth prediction. Deep learning based methods for predicting depth under different setups have demonstrated a dramatic improvement in recent years. Numerous methods have been proposed to predict depth from a single RGB image. Such methods typically use a feed-forward CNN-based model and can be divided into two main categories: *supervised methods* that regress to ground-truth depth maps given a large dataset of images and their corresponding depth maps [Chen et al. 2016; Eigen et al. 2014; Fu et al. 2018; Li and Snavely 2018; Ranftl et al. 2020; Wang et al. 2019; Xian et al. 2018], and *self-supervised methods* [Casser et al. 2019a; Godard et al. 2017, 2019; Yang et al. 2018; Yin and Shi 2018; Zhou et al. 2017] that learn depth prediction by optimizing for photometric consistency, either using stereo images or monocular video. To handle moving objects in training videos, self-supervised methods predict an optical flow field as an additional task (e.g. [Yang et al. 2018; Yin and Shi 2018]), or learn to mask out moving objects (e.g. [Godard et al. 2019]).

Once trained, single-frame methods are agnostic to moving objects, and can be applied to any video in a frame-by-frame fashion. However, because depth is predicted independently per-frame, such an approach results in temporally inconsistent estimates. The loss of epipolar constraints also means single-view prediction lags in accuracy over multi-view stereo (MVS) methods [Schönberger et al. 2016; Seitz et al. 2006] for stationary regions.

Geometry estimation of dynamic scenes. Methods for estimating geometry of a dynamic scene roughly fall into two major categories. The first category of works aim to solve this problem by considering either a multi-camera setup where epipolar geometry constrains can be applied, or using RGBD data (e.g., [Bansal et al. 2020; Basha et al. 2012, 2013; Dou et al. 2016; Innmann et al. 2016; Newcombe et al. 2015; Richardt et al. 2016; Wedel et al. 2011]). The second category of works aim to tackle this problem using monocular videos, which is a more challenging and ill-posed task. To deal with such challenges, some approaches aim to reconstruct the geometry of a dynamic scene by limiting in the type of scenes and objects' motion [Park et al. 2010b; Ranftl et al. 2016; Russell et al. 2014; Simon et al. 2017].

Other methods combine data-driven priors with multi-view stereo methods [Li et al. 2019; Rematas et al. 2018; Yoon et al. 2020]. For example, [Li et al. 2019] uses parallax between two RGB frames to estimate depth in static parts of the image, then inpaints the depth of dynamic regions using a depth-prediction network.

In contrast, our method optimizes a pre-trained depth prediction network over the entire video at test-time to produce consistent estimations. The fusion method of [Yoon et al. 2020] uses the entire video to compute MVS estimates that are combined with a single-view CNN prediction using a learned module. Our method does not require MVS depth as input, only camera poses, and does not require training a fusion module. In addition, some methods [Klingner et al. 2020; Patil et al. 2020] use semantic information to guide depth prediction and identify moving objects during training. Those methods usually focuses on specific scenarios such as autonomous driving.

Our method does not require semantic information and aims to solve for consistent depth maps for general videos.

Test-time training for depth estimation. Recently, *test-time training* methods for depth estimation [Casser et al. 2019b; Chen et al. 2019; Luo et al. 2020] have appeared, which use deep neural networks as optimizers: given an objective function defined over the test data, the variables of optimization are the weights of a deep network that predicts the unknowns, rather than the unknowns themselves. By finetuning a pre-trained network on a single video, these methods both apply data-driven depth priors and aggregate information across the entire video. The method of [Casser et al. 2019b] segments the video and proposes separate motion models for each segment, whereas our scene-flow MLP learns a global motion model. Compared to GLNet [Chen et al. 2019], we explicitly model 3D motion, rather than deriving it from optical flow and depth. We show that this explicit model of motion is important for stability and long-range correspondence. Most closely related to our work is [Luo et al. 2020], which shares our goal of accurate, temporally-consistent depth from video. While their method tolerates small amounts of object motion, it fundamentally assumes a stationary scene, whereas our method is designed to handle large object motion.

Reconstruction from smooth motion priors and non-rigid SfM. Prior to the rise of data-driven methods, reconstruction of dynamic scenes was explored using Nonrigid Structure-from-Motion [Torresani et al. 2008].

Multi-view stereo methods typically treat dynamic objects as outliers and produce either empty or spurious estimates in moving regions. NSfM methods produce estimates for moving regions

by solving a reduced dimension approximation of the problem, or by applying additional losses such as smooth-motion priors (see [Jensen et al. 2020] for a recent survey).

The effectiveness of an explicit smooth-motion prior depends heavily on accurate tracking, a notoriously difficult problem in itself. NSfM methods use sparse feature tracks (e.g. [Vo et al. 2016]) or build an explicit geometric model and fit it to 2D observations (e.g. [Torresani et al. 2008]). However, feature tracking can only track sparse points, while explicit geometric models fail to capture real-world sequences with complex motions. Instead, we train an MLP to predict a dense scene flow field at all points in 3D, and apply a smooth-motion prior directly to the outputs of the MLP. A similar method of predicting scene flow was used by [Niemeyer et al. 2019] to align body scans, however they did not take advantage of the unrolling capability of the MLP.

Concurrent Work. Concurrent to our work, several methods have been proposed to tackle reconstruction of dynamic scenes captured by a moving camera. Dynamic NeRF methods aim at generalizing the original Neural Radiance Field framework [Mildenhall et al. 2020] to dynamic scenes by modeling and estimating the scene motion either using a locally-rigid deformation [Park et al. 2020], or a scene flow field [Li et al. 2020b]. Their training loss is primarily a reconstruction loss over the RGB video frames. These methods have demonstrated impressive results for synthesizing novel views of a dynamic scene, but they are currently limited to short (2 seconds) videos [Li et al. 2020b], or small motions [Park et al. 2020]. In contrast to NeRF-based methods, our main goal is to estimate general purpose depth maps for long videos with arbitrary camera and object motion.

Robust CVD [Kopf et al. 2020] also aims to produce depth maps from monocular video, in addition to estimating camera poses. Their method also applies a pre-trained depth prediction model but does not model the 3D motion in the scene, instead assuming that the depth error in dynamic regions can be removed using a spatially and temporally smooth scaling of the initial depth. In contrast, our method finetunes the weights of the depth prediction network while estimating a dense scene-flow field for each frame, and so can resolve initial depth errors that cannot be resolved by smooth scaling.

3 METHOD

3.1 Overview

Figure 2 illustrates the pipeline of our method. Our system takes a calibrated video as input where both the objects in the scene and the camera are naturally moving, and predicts per-frame depth maps. This is done via an *optimization framework* where two networks are trained, at test time, on the input video: (i) a *single-frame CNN depth prediction model* that takes an RGB image as input and outputs a depth map. (ii) a *scene flow MLP prediction model* that takes a 4D point (x, y, z, t) as input and outputs its 3D displacement to the next time step $t + 1$. Both the depth and the scene flow networks are *trained jointly* where the depth network is initialized with a pre-trained monocular depth prediction model (e.g., [Li et al. 2020a] or [Ranftl et al. 2020]), and then finetuned over the input video; this initialization equips our model with depth priors learned from external source of data [Luo et al. 2020]. The scene flow module allows us to explicitly model the 3D motion of arbitrary dynamic objects in the scene, and is trained from scratch.

In a pre-processing step, we compute camera poses of all the frames and optical flow fields between pairs of frames $\{I_i, I_j\}$ using off-the-shelf techniques (Section 3.2). The computed camera poses and optical flow fields are used to apply two types of re-projection losses: a *flow consistency loss*, i.e., we require that the 2D displacement field between two frames resulting by projecting the predicted depth and scene flow onto 2D would match the pre-computed 2D optical flow field between the frames; we further apply a *depth consistency loss* which encourages coherency between the depth and scene flow estimates across different frames. Finally, we impose a local linear motion prior on the predicted scene flow to reduce the inherent ambiguity between depth and motion. We next describe each of these losses in detail.

3.2 Pre-processing

We use a similar pre-processing strategy by presented by [Luo et al. 2020]. The input RGB frames I_i are first triangulated with ORB-SLAM2 [Mur-Artal et al. 2015] for initial pose estimates, which is later refined by the structure-from-motion stage of COLMAP [Schönberger and Frahm 2016] to produce camera poses R_i, t_i and sparse depth maps D_i^{sfm} for every frame. For sequences where motion masks are available, we use the multi-view stereo stage of COLMAP [Schönberger et al. 2016] for more accurate poses and denser depth maps. Then, forward optical flow $v_{i \to i+k}$ and

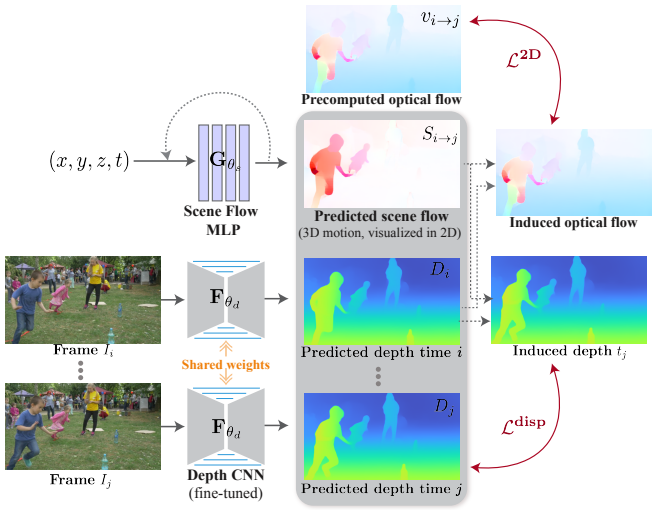

Fig. 2. **Pipeline.** Our system takes as input an ordinary video with freely-moving camera and scene objects and predicts a depth map for each frame of the video. We assume known camera poses and compute optical flow between frames in a per-processing step. Our system consists of two networks: A *depth prediction CNN* F_{θ_d}, which takes I_i (RGB frame) as input and outputs a depth map D_i; we initialize this network with a pre-trained monocular depth model (e.g., [Li et al. 2020a]); and a *scene flow MLP* G_{θ_s}, which takes a space-time point, (x, y, z, t), as input and outputs its scene flow vector, i.e., 3D displacement vector w.r.t. $t + 1$. The depth and scene flow networks are trained jointly in a self-supervised manner under two losses: $\mathcal{L}^{2D}$ measures the difference between the induced flow and the input optical flow, while $\mathcal{L}^{disp}$ measures the difference between the induced depth at time j and the predicted depth at time j. We further apply a constant velocity prior on the scene flow vectors (see Section 3.3).

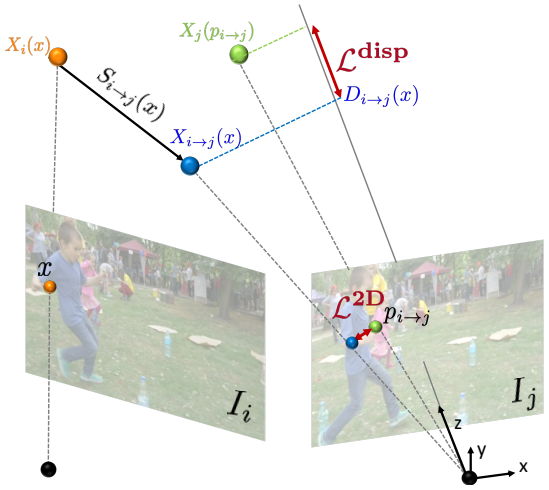

Fig. 3. **Depth and scene flow training losses.** A pixel x in frame i is back-projected into a 3D point $X_i(x)$ (orange point), using our predicted depth map value $D_i(x)$. Similarly, the corresponding pixel of x in frame j, denoted by $p_{i\to j}$, is backprojected to $X_j(p_{i\to j})$ (green point). We require consistency between $X_j(p_{i\to j})$ and $X_i(x)$ after displacement by our predicted scene flow $S_{i\to j}$, denoted by $X_{i\to j}(x)$ as follows: (i) $\mathcal{L}^{\mathbf{2D}}$ penalizes the 2D distance between the projection of $X_{i\to j}(x)$ onto frame j, and $p_{i\to j}$. (ii) $\mathcal{L}^{\mathbf{disp}}$ penalizes the difference in disparity between $X_{i\to j}(x)$ and $X_j(p_{i\to j})$. See details in Section. 3.3.

backwards optical flow $v_{i+k\to i}$ are computed between subsequent frames ($k = 1$) and between a subset of wide-baseline frame pairs ($k \in [2, 4, 6, 8]$) using RAFT [Teed and Deng 2020]. We then generate initial depth maps D_i^{init} using a single-frame depth prediction network [Li et al. 2020a; Ranftl et al. 2020]. Since such predictions are scale-invariant, we align the scale of t_i to roughly match the the scale of initial depth estimates. Specifically, the scale s is calculated by: $s = mean(median(D_i^{init}/D_i^{sfm}))$, and applied for all camera translations t_i.

For each pair of optical flow fields $v_{i\to i+k}$ and $v_{i+k\to i}$ we find occluded regions (as well as regions of inaccurate flow) using forward-backward consistency to generate an occlusion mask $MO_{i\to i+k}$ for masking the loss computation. All flows with inconsistency larger than 1 pixel are considered to be occluded or unreliable. Formally:

$$MO_{i\to i+k}(x) = \begin{cases} 1, & \text{if } |v_{i\to i+k}(x) + v_{i+k\to i}(x + v_{i\to i+k})|_2 > 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where x denotes pixel locations in frame i.

3.3 Test-time Learning of Depth and Scene Flow

Given the input video, along with the pre-computed optical-flow fields and camera poses (see Section 3.2), we turn to the task of fine-tuning a pre-trained, single-frame, depth-prediction network, F_{θ_d} that takes an RGB frame I_i as input and produces a depth map D_i, and training from scratch an auxiliary scene-flow-prediction network G_{θ_s} that takes a 3D point in world coordinate system X, and predicts its 3D world-space scene flow $S_{i\to i+1}$.

The networks' parameters θ_d and θ_s are the only variables of the optimization and are trained in conjunction: the inputs to the scene flow network are updated at each optimization step, based on the current depth estimate; and the scene flow losses are back-propagated to update our depth model at each step.

Theoretically, the scene flow module is redundant—it can be analytically computed given 2D dense correspondences between frames (optical flow) and depth estimates for each frame. However, such estimation can often be noisy and unstable especially when the camera rays at corresponding points are nearly parallel. Implicitly representing the scene flow using our MLP network acts as an auxiliary variable that is encouraged to match the analytically computed scene flow through the training losses. This design choice is crucial for making the optimization stable as demonstrated in Section. 5.1.

Our objective loss for optimizing G_{θ_s} and F_{θ_d} consists of three terms:

$$\arg\min_{\theta_d,\theta_s} \mathcal{L}^{2D} + \alpha\mathcal{L}^{disp} + \beta\mathcal{L}^{prior} \tag{2}$$

$\mathcal{L}^{2D}$ encourages the depth and scene flow between two frames to match the pre-computed optical flow when projected onto 2D. $\mathcal{L}^{disp}$ encourages the predicted depth D_i and scene flow w.r.t. frame j, $S_{i\to j}$ to be consistent with D_j, the predicted depth of frame j. $\mathcal{L}^{prior}$ is a smoothness prior imposed on the predicted scene flow.

The hyper-parameters α and β control the relative weighting of the different terms. We next describe each of the loss terms in detail.

3.4 Training Losses

Given a pair of frames $\{I_i, I_j\}$, we define the following losses on the predicted depth maps $\{D_i, D_j\}$, and the predicted scene flow between the frames $S_{i\to j}$. An visual illustration of these losses and our notation is presented in Fig. 3.

We unproject each pixel $x \in I_i$ into a 3D point $X_i(x)$ in world coordinates, using the predicted depth map D_i and camera poses:

$$X_i(x) = R_i(D_i(x)K_i^{-1}\tilde{x}) + t_i, \tag{3}$$

where $\tilde{x}$ is the 2D homogenous augmentation of x, K_i is the camera intrinsics matrix, R_i is the 3x3 camera rotation matrix, and t_i is the camera translation vector.

We can now compute the scene flow of $X_i(x)$ w.r.t. time $i+1$ by feeding it into our MLP scene flow network. That is,

$$S_{i\to i+1}(x) = G_{\theta_s}(X_i(x), i). \tag{4}$$

To compute the scene flow w.r.t. frame j where $j - i > 1$, we can simply unroll the scene flow MLP $j - i$ times:

$$\begin{aligned} &S_{i\to j}(x) = G_{\theta_s}(\tilde{X}_{j-1}, j-1), \\ &\tilde{X}_k = \tilde{X}_{k-1} + S_{k-1\to k}(x), \qquad k = i+1, ..., j-1. \end{aligned} \tag{5}$$

Finally, the 3D position of $X_i(x)$ at time j is given by:

$$X_{i\to j}(x) = X_i(x) + S_{i\to j}(x). \tag{6}$$

Consistency in 2D. We define a 2D consistency loss $\mathcal{L}^{2D}_{i\to j}(x)$ that measures the pixel distance between the corresponding pixel of x in frame j, and the projection of $X_{i\to j}(x)$ onto camera j.

Formally, let $v_{i\to j}(x)$ be the pre-computed optical flow vector of pixel $x \in I_i$ w.r.t. frame I_j. The corresponding pixel of x is given by:

$$p_{i\to j}(x) = x + v_{i\to j}(x). \tag{7}$$

The loss term $\mathcal{L}^{2D}$ is given by:

$$\begin{aligned} &\mathcal{L}^{2D}_{i\to j}(x) = \left\| M_j(X_{i\to j}(x)) - p_{i\to j}(x) \right\|_1, \\ &\mathcal{L}^{2D} = \textstyle\sum_{i,j} \sum_{x\in\{MO_{i\to j}=0\}} \mathcal{L}^{2D}_{i\to j}(x), \end{aligned} \tag{8}$$

where $\|\cdot\|_1$ denotes the L_1 norm, $MO_{i\to j}$ denotes the occlusion mask of frame i w.r.t frame j, and $M_j(X)$ denotes the projection of a world-space point X onto the j^{th} camera:

$$M_j(X_i) = \pi(K_j R_j^T(X_i - t_j)), \tag{9}$$

where π is the projection operator $\pi([x, y, w]^T) = [x/w, y/w]^T$.

Consistency in 3D. Similarly to [Luo et al. 2020], we define the disparity consistency loss $\mathcal{L}^{disp}$, which measures the consistency between the inverse depth value of $X_{i\to j}(x)$ under camera j, and the inverse of D_j:

$$\begin{aligned} &D_{i\to j}(x) = |K_j R_j^T(X_i + S_{i\to j}(x) - t_j)|_z \\ &\mathcal{L}^{disp}_{i\to j}(x) = \left\| \tfrac{1}{D_{i\to j}(x)} - \tfrac{1}{D_j(p_{i\to j}(x))} \right\|_1 \\ &\mathcal{L}^{disp} = \textstyle\sum_{i,j} \sum_{x\in\{MO_{i\to j}=0\}} \mathcal{L}^{disp}_{i\to j}(x) \end{aligned} \tag{10}$$

where $|\cdot|_z$ denotes taking the depth component of a 3D point, i.e. $z = |[x, y, z]|_z$.

Smooth 3D motion. To regularize the motion in 3D, we impose a constant velocity prior $\mathcal{L}^{prior}$:

$$\begin{aligned} &\mathcal{L}^{prior}_i(x) = \left\| S_{i\to i+1}(x) - G_{\theta_s}(X_i(x) + S_{i\to i+1}(x), i+1) \right\|_1, \\ &\mathcal{L}^{prior} = \textstyle\sum_i \sum_{x\in I_i} \mathcal{L}^{prior}_i(x) \end{aligned} \tag{11}$$

where x is the pixel location in frame i.

Optional motion masks. If motion segmentation is available for the video sequence, we can further constrain the motion in 3D by penalizing velocities in static regions. Formally, given a mask of static regions at frame i, denoted as MS_i, we introduce a static prior $\mathcal{L}^{static}$:

$$\begin{aligned} &\mathcal{L}^{static}_i(x) = \left\| S_{i\to i+1}(x) \right\|_1, \\ &\mathcal{L}^{static} = \textstyle\sum_i \sum_{x\in MS_i} \mathcal{L}^{static}_i(x). \end{aligned} \tag{12}$$

The total loss when using static/dynamic masks is given by:

$$\mathcal{L}^{total}_{mask} = \mathcal{L}^{2D} + \alpha\mathcal{L}^{disp} + \beta\mathcal{L}^{prior} + \gamma\mathcal{L}^{static}. \tag{13}$$

Otherwise, the loss is given by Eq. 2. We **do not** use any motion segmentation throughout our experiments unless explicitly noted; for real videos, only the ablation study (Fig. 8) includes a motion mask.

Training schedule. As discussed in Section 3.1, we initialize the depth network with a pre-trained monocular depth model. However, the scene flow network G_{θ_s} has to be trained from scratch. Naively initializing G_{θ_s} with random weights produces small initial scene flow values that are inconsistent with the initial depth estimate and pre-computed optical flow. Training both depth and scene flow networks from this initialization results in large changes in the depth prediction network in order to satisfy the consistency losses. We overcome this issue via a "warm-up" phase in which only G_{θ_s} is trained using $\mathcal{L}^{total}$ with $\beta = 0$ while the pre-trained depth network F_{θ_d} is kept fixed. In practice, we found G_{θ_s} converged after 5 epochs, and set the warm-up phase to 5 epochs in all our experiments, after which both networks are finetuned jointly using $\mathcal{L}^{total}$.

Note that even if the predicted scene flow matches the analytic scene flow perfectly after warm-up, there is still a training signal from the smooth-motion prior applied to the scene flow after the warm up phase. The scene flow MLP will deviate from its warmed-up state during training in order to find a solution that induces locally linear motion while satisfying the optical flow loss terms.

4 IMPLEMENTATION DETAILS

Network architectures. In our experiments , we use either the single-frame MannequinChallenge model [Li et al. 2019] or the MiDaS v2 model [Ranftl et al. 2020] for the depth prediction network F_{θ_d}. The scene flow network G_{θ_s} is a MLP with positional encoding [Mildenhall et al. 2020] for the input. Specifically, the positional encoding $\xi(x)$ for input x is defined as:

$$\xi(x) = [\sin \pi x, \cos \pi x, \sin 2\pi x, \cos 2\pi x, ... \sin N\pi x, \cos N\pi x],$$

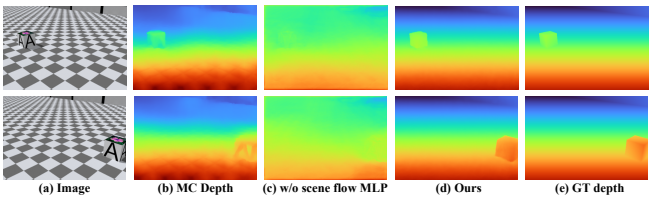

Fig. 4. **MLP scene flow vs. analytic scene flow.** (a) Sample frames from the *cube* video – the cube is moving along a straight line in 3D, while the camera is moving side-to-side. (b) Initial depth maps computed by MC [Li et al. 2019]. (c) Results produced when only the depth network is optimized, while the scene flow is computed analytically ('w/o scene flow MLP'); this baseline fails to converge to the correct solution (e). Our method (d) produces near perfect results under this setup. In both (c) and (d), we use ground truth optical flow and camera poses. See more details in Section 5.1.

where we use $N = 16$ for all the experiments, and the encoding is applied to both 3D locations and time. The MLP is comprised of 4 hidden layers, each with 256 units.

Hyperparameters. Throughout our experiments, we set $\alpha = 0.1$, $\beta = 1$ for the total loss defined in Eq. 2. When motion segmentations are present, we set $\gamma = 100$ in Eq. 13. We use a learning rate of 10^{-5} for the depth network when initialized with the MannequinChallenge model [Li et al. 2019], and a learning rate of 10^{-6} when initialized with the MiDaS v2 model [Ranftl et al. 2020]. The sceneflow network is optimized with a learning rate of 0.001. Both networks are optimized using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Batch size is set to 1.

Training Time. The networks are trained for 20 epochs for all experiments, so the training time of our method depends linearly on the number of frames. For a 60-frame video, the training time is about 75 minutes on a single NVIDIA P100 GPU. The training time of our method is about 2× longer than CVD [Luo et al. 2020], due to the extra scene flow network.

5 RESULTS

Our experiments demonstrate the quality of predicted depth and the importance of the scene flow network on two synthetic datasets and real-world videos. We first justify the design choice of the scene flow MLP against analytically deriving scene flow (Sec. 5.1). We then evaluate our method on the synthetic Sintel [Butler et al. 2012] dataset and show quantitative improvement over competing methods (Sec. 5.2). To demonstrate the real-world capabilities of the method, we show qualitative examples on videos with significant camera and object motion (Sec. 5.3). Finally, we show multiple video editing results based on our consistent depth estimates (Sec. 5.4).

5.1 MLP scene flow vs. Analytic scene flow

We examine the importance of representing scene flow implicitly via our MLP model vs. analytically computing scene flow (using the depth estimates, camera poses and optical flow fields). To do so, we generated a simple synthetic scene where a cube is moving along a straight line in 3D, towards the camera at a constant velocity, while the camera is moving along a sine curve that simulates handheld camera wobble (see Figure 4a, and full video in the SM). Using ground truth optical flow fields and camera poses, we then train:

(1) Our full framework, including both depth and scene flow networks, using Eq. 2.
(2) Only depth network only (no MLP scene flow). At each iteration of training, we compute scene flow given the current depth estimates:

$$\hat{S}_{i\to i+1}(x) = X_{i+1}(p_{i\to i+1}(x)) - X_i(x).$$

where $X_i(x)$ and $X_{i+1}(p_{i\to i+1}(x))$ are computed by unprojecting pixel $x \in I_i$, and its corresponding pixel $p_{i\to i+1}(x) \in I_{i+1}$ to 3D, using the depth estimates $\{D_i, D_{i+1}\}$ as defined in Eq. 3. We then plug the computed scene flow into Eq. 2 by replacing all $S_{i\to i+1}$ and $G_{\theta_s}(X_i, i)$ with $\hat{S}_{i\to i+1}$. Note that by construction, $\mathcal{L}^{2D}$ and $\mathcal{L}^{disp}$ are perfectly satisfied (the scene flow is derived from two depth maps and flow). Thus, the only loss that drives the optimization is $\mathcal{L}^{prior}$. Intuitively, this baseline simply optimizes the depth network such that the derived scene flow is locally smooth.

As can be seen in Figure 4, the baseline w/o scene flow MLP does not converge to the correct solution, even when ground truth optical flow and camera poses are used to derive scene flow.

We hypothesize the failure is due to the $\mathcal{L}^{prior}$ term being ill-posed for analytic scene flow. When the motion of the camera and the object are nearly aligned, even momentarily, $p_{i\to i+1}(x) \approx x$. In this case analytic scene flow is ill-posed (see Appendix A.1) and the gradient of $\mathcal{L}^{prior}$ is numerically unstable, leading to poor convergence.

In contrast, our scene flow network G_{θ_s} avoids this issue by acting as a slack variable that aggregates information across the entire video. When $\mathcal{L}^{prior}$ is defined in terms of the output of G_{θ_s} (Eq. 11), the scene flow at $X_i(x)$ potentially depends on *all the rays in the scene*. Each frame of the video has $\approx 2^{17}$ pixels, whereas G_{θ_s} has only $\approx 2^{18}$ parameters, so weights must be shared between rays.

In the presence of badly conditioned ray pairs these shared weights act as a regularizer, allowing our method to nearly perfectly reconstruct the ground truth cube (Fig. 4d).

5.2 Quantitative Evaluation

We quantitatively evaluate, ablate and compare our method on a subset of the Sintel dataset [Butler et al. 2012], as follows.

Dataset. We test the depth prediction accuracy of our method using the 23 sequences of the Sintel dataset. We use the motion masks provided by [Taniai et al. 2017] as MS_i and for separately evaluating performance in static and dynamic regions. Input optical flow is computed using RAFT [Teed and Deng 2020], not the ground-truth flow from the dataset.

Baselines. We compare our method with three baselines: the single-frame MannequinChallenge model(MC) [Li et al. 2019], single-frame MiDaS v2 [Ranftl et al. 2020] and CVD. For CVD with MC initialization, we use the original implementation provided by the authors. We found training the original implementation of CVD numerically unstable (training loss being NaN for some sequences)

Table 1. **Depth accuracy on Sintel.** We evaluate error for the entire depth map (Full), and separately for the dynamic and static regions. We evaluate the initial depth maps produced by MC (top) and MiDaS (bottom), along with the depth prediction results of CVD and our method using each of the depth initializations. Additionally, we evaluate our performance when the smooth motion prior is omitted from our objective (w/o $\mathcal{L}^{prior}$), and when static/dynamic segmentation masks are incorporated (Eq. 13). Our full method consistently improves upon the initial depth estimations, in both static and moving regions. Since CVD assumes a static scene, it fails to recover large moving regions. Given the motion masks, our performance further improves in static and dynamic regions, and is on par with CVD in static regions. Additionally, we observe a significant degradation in our performance when $\mathcal{L}^{prior}$ is omitted, demonstrating its importance in reducing the ambiguity between depth and 3D motion. CVD† denotes our implementation of CVD under MiDaS initialization, as the original code fails to train on some of the sequences (see Sec. 5 for more details.)

Method	Full			Dynamic			Static		
	L_1 rel. ↓	log RMSE ↓	RMSE ↓	L_1 rel. ↓	log RMSE ↓	RMSE ↓	L_1 rel. ↓	log RMSE ↓	RMSE ↓
MC [Li et al. 2019]	0.6508	1.1978	4.0830	0.5083	0.4974	3.6870	1.3691	1.3443	2.9392
CVD, MC init [Luo et al. 2020]	1.5208	0.6874	3.3531	2.9116	0.8814	4.9374	**0.3291**	0.2902	1.8832
Ours, MC init, w/o $\mathcal{L}^{prior}$	0.6753	0.7837	3.7396	0.8227	0.6575	3.7171	0.6654	0.6803	2.5594
Ours, MC init	0.5280	0.6907	3.1284	0.4841	0.5655	3.0244	0.9963	0.5723	2.2443
Ours, MC init, w/ mask	**0.3540**	**0.4877**	**2.8530**	**0.4707**	**0.5464**	**2.8469**	0.3494	**0.2845**	**1.8480**
MiDaS [Ranftl et al. 2020]	0.4506	0.7440	3.2763	0.5202	0.6361	2.5613	0.5743	0.6243	2.7371
CVD†, MiDas init	0.6245	1.0754	4.7060	1.7494	1.4744	6.9751	**0.2321**	**0.3154**	1.6662
Ours, MiDas init, w/o $\mathcal{L}^{prior}$	0.4095	1.7713	3.8590	0.5758	0.6400	2.6002	0.2865	0.5438	2.7245
Ours, MiDas init	0.3838	0.5097	2.6733	0.4520	0.4997	2.5374	0.3746	0.3814	1.9427
Ours, MiDas init, w/ mask	**0.3063**	**0.4708**	**2.4941**	**0.4468**	**0.3990**	**2.5018**	0.2455	0.3347	**1.6082**

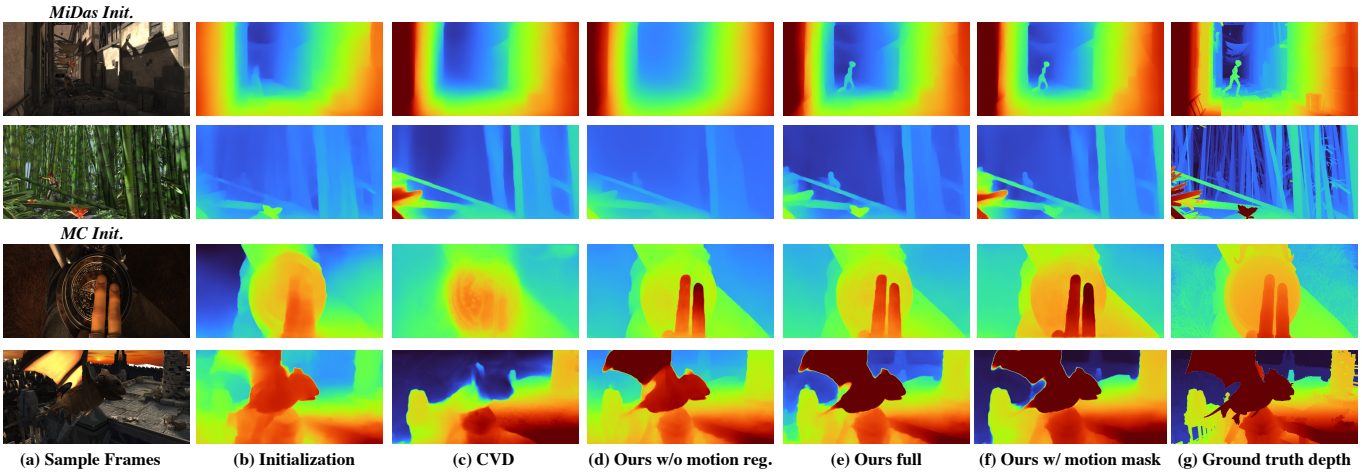

Fig. 5. **Qualitative results on Sintel.** Variants of our method are shown in (d,e,f) using MiDaS v2 [Ranftl et al. 2020] as initialization, along with initial depth estimates (b) and CVD (c). Top two rows show initialization with MiDaS, while bottom two show initialization with MC. Both MiDaS and MC initialization are roughly correct but show visible errors. CVD improves static regions over the initialization, but fails on moving regions. Without the acceleration regularizer, our method fails to recover the moving character (d, top row); with the regularizer (e), the character is put at the correct depth. Adding a motion mask to our method (f) further improves the depth prediction quality on static regions.

when initialized with MiDaS due to the near-zero disparity values MiDaS produces in background areas. We therefore implemented a modified version of CVD that clips near-zero disparity to a small, non-zero value using our frame work: removing the scene flow MLP and replace our losses with the ones proposed by the original paper.

Ablation. We compare our method when: the smooth motion prior is omitted from our objective ("w/o motion prior"), and w/o and w/ dynamic/static motion segmentation masks (Eq. 2 and Eq. 13, respectively.)

Metrics. We use three metrics to evaluate the quality of the depth maps: L_1 relative, log RMSE, and RMSE [Butler et al. 2012]. To examine temporal consistency, we do not match the scale of each individual frame against the ground truth during evaluation. Instead, we apply a single calibrated scale for the entire sequence during the preprocessing stage, as described in Sec. 3.2. We also evaluate separately on moving and static regions using the ground truth motion mask. We follow the common practice of excluding depth values that exceed 80m during evaluation, as distant, high-magnitude outliers otherwise confuse the results.

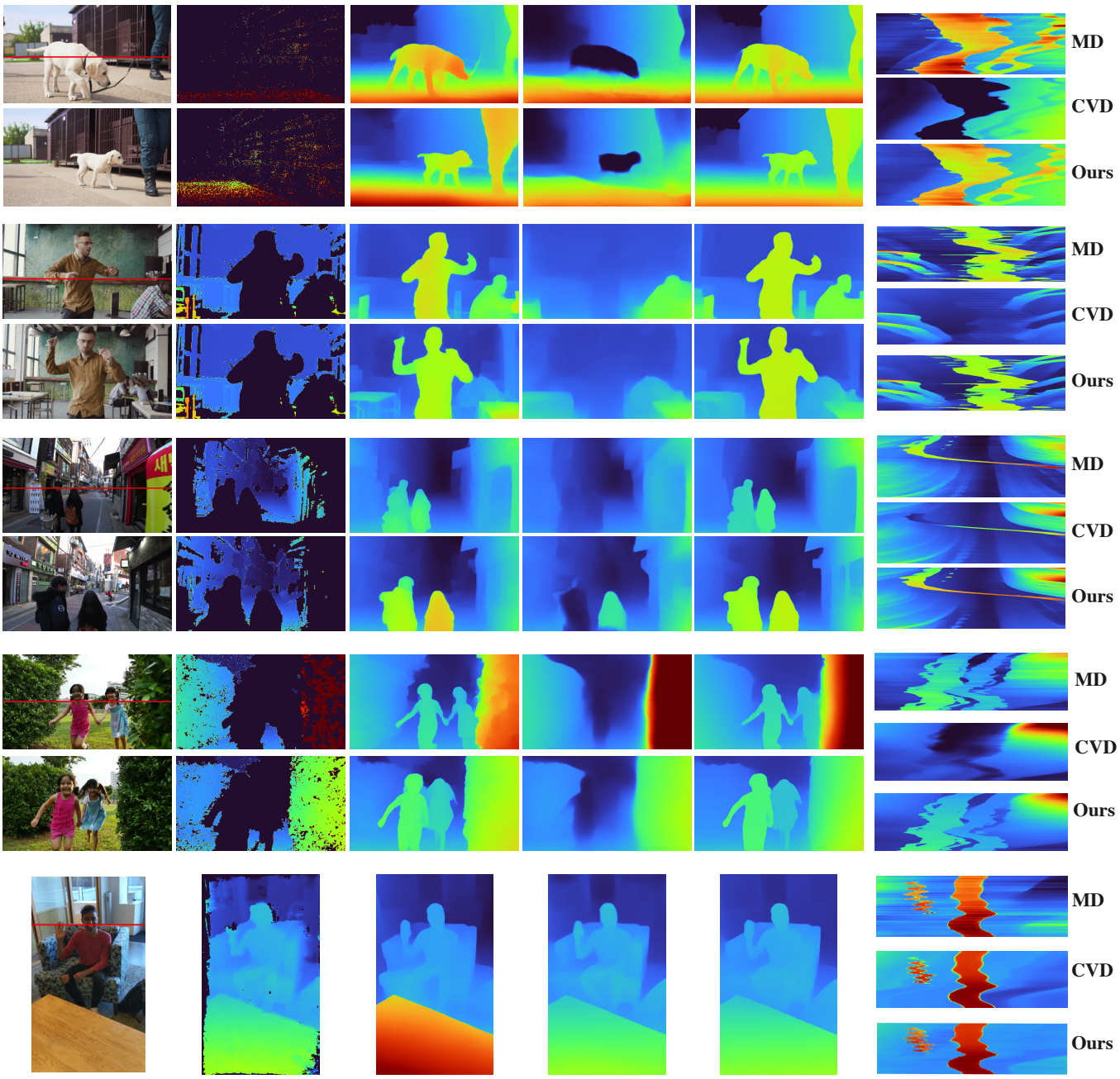

Fig. 6. **Results on real videos.** (a) Two representative video frames. (b) Sparse depth maps produced by running SfM/MVS (see Section 3.2); top two rows show COLMAP SfM output, and the rest COLMAP MVS results (dynamic regions are filtered out and assigned zero depth). (c) Initial depth maps produced by MiDaS (single-frame model), and by (d) CVD which optimizes single-frame depth prediction model over the input video. (e) Our predicted depth maps (no motion segmentation masks were used). (f) Corresponding $x - t$ slice for each of the methods (the sampled horizontal line is marked in red in (a)): MiDas produces inconsistent depth over time, apparent by the zigzagging patterns in the slice. CVD outputs temporally consistent depth estimates in the static regions, but fails in the dynamic regions (darker regions in the slice). Note that the last row depicts small, in-place motion (person is waving), in which case most of the person region is reconstructed by MVS. In all example, our method is able to produce temporally consistent depth estimates in both static and dynamic regions.

Fig. 7. **Scene flow results.** Colors correspond to the direction of the projected flow, while saturation corresponds to magnitude (white is zero flow). Optical flow (b) contains both background and foreground motion. Projected scene flow derived from the initial depth estimate (c) and optical flow (b) has small, but still noticeable motion in stationary areas. Our projected scene flow (d) is concentrated in the truly moving regions of the scene.

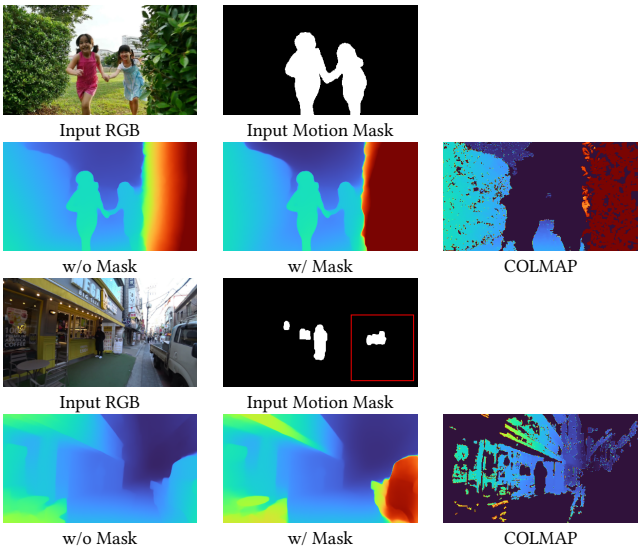

Fig. 8. **Results with a motion mask.** When an accurate motion mask is available as input, our method can incorporate it to improve static regions (top): note the hedge on right and sharper background. However, if the motion mask is inaccurate, it can negatively affect the results (bottom): note truck is not marked as moving (red box) and is spuriously pulled towards the camera.

As can be shown in Table 1, our method, when initialized with the corresponding single-frame depth prediction model, consistently improves upon the initialization in both moving and static regions by an average of 40-45% in L_1 relative error. Our method outperforms CVD in moving regions by a large margin due to our explicit handling of motion, while CVD still performs well in static regions. When motion masks are incorporated into our framework, we achieve similar performance to CVD in static regions. Finally, removing the motion prior $\mathcal{L}^{prior}$ for our objective leads to 150-175% increase in L_1 relative error, which demonstrates the importance of this regularization.

Figure 5 shows several qualitative results. Note that our method (e) using MiDaS v2 [Ranftl et al. 2020] as initialization improves depth in both moving and static regions compare to the initial depth estimates (b) even when motion segmentation masks are not used. Using the motion masks further improves our results (e,f). CVD improves the static regions but fails to produce correct depth for moving objects, tending to erase them to the background value (c).

5.3 Real Videos – Qualitative Results

We test our method on real world videos containing a variety of moving objects (people, pets, cars) performing complex motions such as running or dancing. Representative results are shown in Fig. 6, and the full set of videos is provided in the SM. All of our results on real videos are produced with MiDaS [Ranftl et al. 2020] initialization. In addition to depth maps, we also present depth x-t slices (Fig. 6f) for qualitative comparison of temporal consistency of our results compared to the initial depth (MiDas) and CVD.

While the per-frame depth maps produced by MiDaS (Fig. 6c) are generally sharp and accurate in terms of overall depth ordering, our method shows improved temporal consistency as shown by the reduced zigzagging patterns and color variation in the x-t slices.

The full-video method CVD produces good temporal consistency and can reconstruct some small motions (e.g. waving hand, Fig. 6 bottom). However, CVD tends to push large moving regions to infinity (Fig. 6d) due to the violation of epipolar constraints, while our method works well for the moving regions as well as the static regions. Note that no input segmentation of moving vs. static regions is used for these results. The CVD results shown in Fig. 6 is produced by the original implementation by the authors, using the MannequinChallenge model as initialization. CVD with MiDaS initialization fail to train on some of the sequences due to reasons described in Sec. 5.2

The output of the scene flow MLP is visualized in Fig. 7 by projecting to image space. The scene flow of our method (d) is almost zero on static regions, while scene flows derived from the initial depth estimates (c) assign significant motion to stationary objects, in some cases of similar magnitude as moving objects (ground and wall in first row, hedge and sky in second row). This spurious motion stems from the lack of temporal coherence in the initial depth maps. Note that our method produces accurate, non-zero scene flow even when optical flow is close to zero (head of left person in second row, center person in bottom row).

When an accurate motion mask is available for a real video, our method can incorporate it to improve results in static parts of the scene (Fig. 8, top). Using an inaccurate motion mask can harm results, however (Fig. 8, bottom): the truck is incorrectly marked as stationary, and is spuriously pulled towards the camera as a result. We do not use motion masks for any results other than Fig 8.

5.4 Applications

We explore several applications of our depth maps, including inserting virtual objects into a video, inserting a new light source that

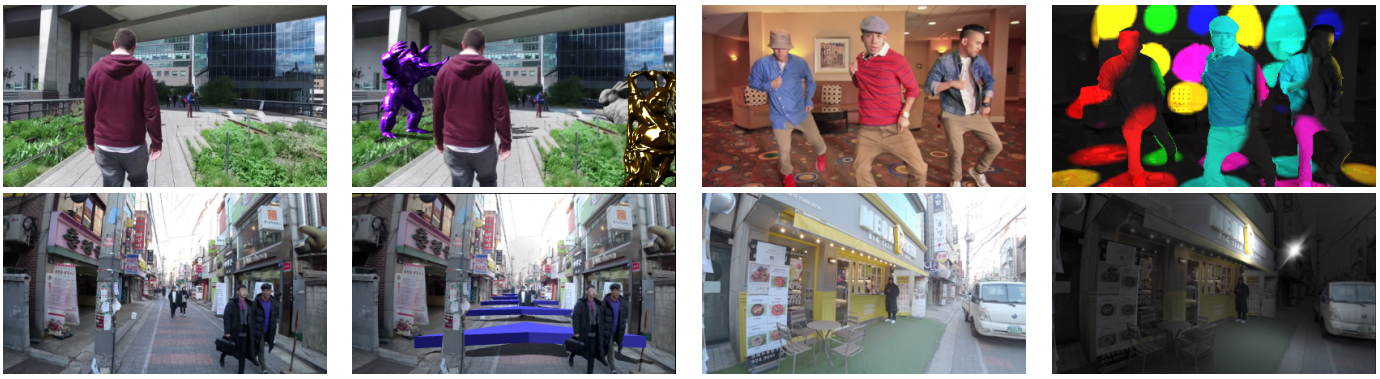

Fig. 9. **Object and light insertion.** Because of their temporal coherence, the depth buffers produced by our method may be used to insert objects (top) or position-dependent lighting effects (bottom) without distracting flickering. Please see supplemental material for a video comparison with single-frame depth prediction.

travels the scene, or segmenting the video via tracking of simple proxy geometries in 3D. In all these applications, the geometrical and temporal consistency of the depth maps play a critical role in producing realistic effects in which depth ordering of moving objects is preserved and the result is consistent over time.

Object and Light Insertion. Fig. 9 shows the results of inserting synthetic objects and lights into real-world videos. The 3D scene is created by unprojecting our depth maps per-frame using the per-frame input camera, without further 3D processing. To slightly improve the sharpness of depth edges, we identify edges between moving and static regions using the depth map and projected scene flow, then separately apply morphological erosion followed by dilation to the moving and static regions in order to replace values along depth edges with depth values from interior regions. Lights and objects were placed manually using NUKE [Ltd 2018].

Temporal coherence is critical for creating effective insertion results: while the human eye is often not sensitive to small inaccuracies in shadow or lighting placement, it is extremely sensitive to the jitter or flashing that can occur when the depth predictions near an inserted object are unstable. The depth maps produced by our method are stable enough to avoid distracting artifacts compared with single-frame depth prediction, even with large moving objects (see supplementary video for comparison).

3D Segmentation. The stability of our depth maps allows for effective manual segmentation of videos using 3D information. The user may place simple proxy geometry around the objects in 3D, then color the points by the proxy geometry they fall inside (Fig. 10). The proxies may be animated to track moving objects such as the dog and walking person (Fig. 10, left). Loose proxy geometry usually suffices to separate the objects in 3D: for the "puppy" sequence, each segment was represented by either a box (dog, person) or a plane (ground, wall, background). The position of the bounding boxes was manually adjusted over time. Only 8 keyframes for the 121 frame sequence were required. This 3D setup can be achieved in a few minutes using a 3D video editing package such as NUKE [Ltd 2018].

Unlike data-drive segmentation models such as Mask R-CNN [He et al. 2017], this approach allows the user to segment objects that

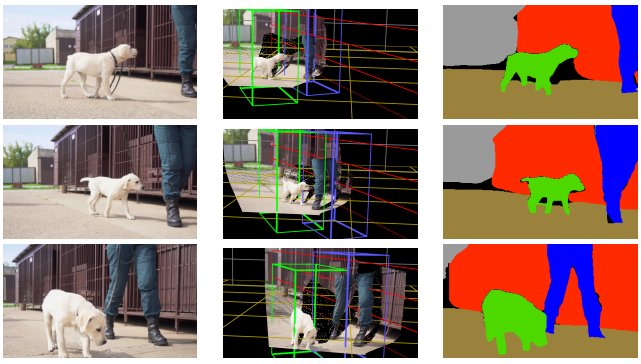

Fig. 10. **Image segmentation using 3D proxy geometry.** An input image (left) may be segmented by using simple proxy geometry in 3D (center). Points that lie within boxes (dog, person) or behind planes (ground, wall, background) are colored by the respective proxies' color to produce the segmentation (right). Black indicates areas outside of a proxy or inside multiple proxies. The green and blue boxes track the dog and person using manual keyframes. Because of the temporal coherence of the depth maps, keyframes were needed on only 8 of 121 frames for this sequence.

may not correspond to a specific label in the segmentation training set (e.g., the red wall in Fig. 10, right). To obtain this result using learned, image-based segmentation, a dataset of ground-truth segmentations with labels corresponding to geometric features would typically be required.

6 DISCUSSION AND LIMITATIONS

Our method generates high-quality, temporally consistent depth maps for arbitrary moving objects in video, suitable for effects such as object insertion, relighting, and 3D segmentation. There are two main areas for improvement, however: failures due to inaccuracy of optical flow, and accuracy of occlusion boundaries.

Our performance is affected by the accuracy of the optical flow estimates. While state-of-the art optical flow methods are getting better and better, they still have errors in some cases. Our method

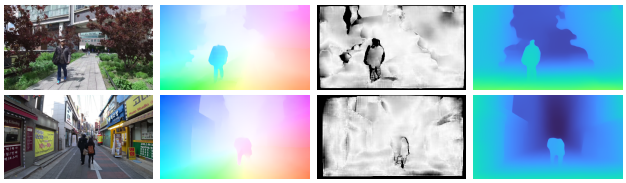

Fig. 11. **Effect of optical flow failure.** Even when evaluated with wide baselines, RAFT optical flow [Teed and Deng 2020] may be inaccurate. Where our mask $MO_{i \to i+k}$ marks flow as inaccurate, our depth estimates are robust to flow errors (top: note head of person is dark in consistency map). If, however, flow is self-consistent but wrong, our method produces incorrect results (bottom, note heads are marked consistent).

can handle optical flow errors to some extent as long as the erroneous regions can be identified using a left-right consistency check and removed from our objective (as described in Sec. 3.3). We observed that in some cases, severe errors such as removal of heads and limbs can still be left-right consistent (with less than a pixel error). In such cases, the optical flow errors translate into errors in our depth prediction (Fig. 11). This problem can be potentially alleviated by developing more intelligent confidence measurements, or by exploring the use of photometric reconstruction losses.

The occlusion boundaries produced by our method may be inaccurate by a few pixels, and so are not yet suitable for demanding effects such as novel view synthesis. Blurring of depth boundaries is a common side effect of CNN depth prediction, and semi-transparent effects such as hair are not solvable using a single depth map. Edge-aware filtering of depth maps [Barron and Poole 2016] may improve sharpness, but may reduce temporal consistency. In future work, we hope to integrate a matting or layering approach (e.g. [Lu et al. 2020]) to better handle difficult depth boundaries.

REFERENCES

Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 2020. 4d visualization of dynamic events from unconstrained multi-view videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5366–5375.

Jonathan T Barron and Ben Poole. 2016. The fast bilateral solver. In *European Conference on Computer Vision*. Springer, 617–632.

Tali Basha, Shai Avidan, Alexander Hornung, and Wojciech Matusik. 2012. Structure and motion from scene registration. In *IEEE Conf. Comput. Vis. Pattern Recog.* IEEE.

Tali Basha, Yael Moses, and Nahum Kiryati. 2013. Multi-view scene flow estimation: A view centered variational approach. *Int. J. Comput. Vis.* 1 (2013).

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. 2012. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV) (Part IV, LNCS 7577)*, A. Fitzgibbon et al. (Eds.) (Ed.). Springer-Verlag, 611–625.

Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. 2019a. Unsupervised learning of depth and ego-motion: A structured approach. In *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, Vol. 2. 7.

Vincent Michael Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. 2019b. Depth Prediction Without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos. In *AAAI*.

Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. 2016. Single-image depth perception in the wild. *arXiv preprint arXiv:1604.03901* (2016).

Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. 2019. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Int. Conf. Comput. Vis.* 7063–7072.

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip L. Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. 2016. Fusion4D: real-time performance capture of challenging scenes. *ACM Trans. Graph.* 35 (2016).

David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth map prediction from a single image using a multi-scale deep network. *Neural Information Processing Systems* (2014).

Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. 2018. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2002–2011.

Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. 2017. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 270–279.

Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. 2019. Digging into Self-Supervised Monocular Depth Prediction. (October 2019).

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

Matthias Innmann, Michael Zollhöfer, Matthias Niessner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *Eur. Conf. Comput. Vis.*

Sebastian Hoppe Nesgaard Jensen, Mads Emil Brix Doest, Henrik Aanæs, and Alessio Del Bue. 2020. A benchmark and evaluation of non-rigid structure from motion. *International Journal of Computer Vision* (2020), 1–18.

Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. 2020. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *European Conference on Computer Vision*. 582–600.

Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. 2020. Robust Consistent Video Depth Estimation. *arXiv preprint arXiv:2012.05901* (2020).

Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. 2019. Learning the depths of moving people by watching frozen people. In *IEEE Conf. Comput. Vis. Pattern Recog.*

Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. 2020a. MannequinChallenge: Learning the Depths of Moving People by Watching Frozen People. *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2020b. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *arXiv preprint arXiv:2011.13084* (2020).

Zhengqi Li and Noah Snavely. 2018. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2041–2050.

The Foundry Visionmongers Ltd. 2018. *NUKE*. https://www.foundry.com/products/nuke

Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. 2020. Layered Neural Rendering for Retiming People in Video. *ACM Trans. Graph.* 39, 6, Article 256 (Nov. 2020), 14 pages.

Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. 2020. Consistent Video Depth Estimation. (2020).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. 2015. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conf. Comput. Vis. Pattern Recog.*

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *International Conference on Computer Vision (ICCV)*.

Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Sheikh. 2010a. 3D reconstruction of a moving point from a series of 2D projections. In *European conference on computer vision*. Springer, 158–171.

Hyun Soo Park, Takaaki Shiratori, Iain A. Matthews, and Yaser Sheikh. 2010b. 3D Reconstruction of a Moving Point from a Series of 2D Projections. In *Eur. Conf. Comput. Vis.*

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. 2020. Deformable Neural Radiance Fields. *arXiv preprint arXiv:2011.12948* (2020).

Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. 2020. Don't forget the past: Recurrent depth estimation from monocular video. *IEEE Robotics and Automation Letters* 5, 4 (2020), 6813–6820.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2020. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun. 2016. Dense monocular depth estimation in complex dynamic scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*

Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz. 2018. Soccer on Your Tabletop. In *IEEE Conf. Comput. Vis. Pattern Recog.*

Christian Richardt, Hyeongwoo Kim, Levi Valgaerts, and Christian Theobalt. 2016. Dense wide-baseline scene flow from two handheld video cameras. In *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 276–285.

Chris Russell, Rui Yu, and Lourdes Agapito. 2014. Video pop-up: Monocular 3d reconstruction of dynamic scenes. In *Eur. Conf. Comput. Vis.* 583–598.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, Vol. 1. IEEE, 519–528.

Tomas Simon, Jack Valmadre, Iain A. Matthews, and Yaser Sheikh. 2017. Kronecker-Markov Prior for Dynamic 3D Reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (2017), 2201–2214.

Tatsunori Taniai, Sudipta N. Sinha, and Yoichi Sato. 2017. Fast Multi-frame Stereo Scene Flow with Motion Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6891–6900.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*. Springer, 402–419.

Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. 2008. Nonrigid Structure-from-Motion: Estimating Shape and Motion with Hierarchical Priors. *IEEE transactions on pattern analysis and machine intelligence* 30 (06 2008), 878–92. https://doi.org/10.1109/TPAMI.2007.70752

Minh Vo, Srinivasa G Narasimhan, and Yaser Sheikh. 2016. Spatiotemporal bundle adjustment for dynamic 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1710–1718.

Chaoyang Wang, Simon Lucey, Federico Perazzi, and Oliver Wang. 2019. Web stereo video supervision for depth prediction from dynamic scenes. In *2019 International Conference on 3D Vision (3DV)*. IEEE, 348–357.

Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. 2011. Stereoscopic scene flow computation for 3D motion understanding. *Int. J. Comput. Vis.* (2011).

Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. 2018. Monocular relative depth perception with web stereo data supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 311–320.

Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. 2018. Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 0–0.

Zhichao Yin and Jianping Shi. 2018. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1983–1992.

Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. 2020. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5336–5345.

Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. 2017. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1851–1858.

A APPENDIX

A.1 Instability of Analytic Scene Flow

Scene flow computed directly from depth and optical flow is unstable when the angle between associated camera rays is small, as can be shown with the following derivation. Please see [Park et al. 2010a] for further explanation of the multi-view geometry. Under analytical sceneflow, we can rewrite $\mathcal{L}_i^{prior}(x)$ as:

$$\begin{aligned}\mathcal{L}_i^{prior}(x) &= |X_{i+1}(x_1) - X_i(x) - X_{i+2}(x_2) + X_{i+1}(x_1)|_2 \\ &= |X_i(x) - 2X_{i+1}(x_1) + X_{i+2}(x_2)|_2,\end{aligned} \tag{14}$$

where $x_1 = x + v_{i\to i+1}(x)$, $x_2 = x_1 + v_{i+1\to i+2}(x_1)$. We can rewrite $X_i(x)$ as $r_i(x)d_i^r(x) + t_i$, where $r_i(x)$ is the camera ray direction vector at x, $d_i^r(x)$ the ray depth at x and t_i the camera translation vector. To simplify notation, we write $r(x)$ as r_0, $r_{i+1}(x_1)$ as r_1 and $r_{i+2}(x_2)$ as r_2. With similar rewriting for d_i^r and t_i, we can rewrite $\mathcal{L}_i^{prior}(x)$ as :

$$\mathcal{L}_i^{prior}(x) = |r_0 d_0^r + t_0 - 2r_1 d_1^r - 2t_1 + r_2 d_2^r + t_2|_2 \tag{15}$$

Optimizing $\mathcal{L}_i^{prior}(x)$ then becomes solving $Rd = t$ under least squares, where r_1,r_2 and r_3 are columns of R, $d = [d_0^r, -2d_1^r, d_2^r]$ and $t = 2t_1 - t_2 - t_0$. This problem is ill-posed when R is poorly conditioned, which happens when the angles between r_1, r_2 and r_3 are small, for example when the object's and camera's motion align over the span of 3 frames.