DeepV2D: Video to Depth with Differentiable Structure from Motion

Zachary Teed[1] Jia Deng[1,2]
[1]Princeton University [2]University of Michigan
{zteed,jiadeng}@cs.princeton.edu

Abstract

We propose DeepV2D, an end-to-end differentiable deep learning architecture for predicting depth from a video sequence. We incorporate elements of classical Structure from Motion into an end-to-end trainable pipeline by designing a set of differentiable geometric modules. Our full system alternates between predicting depth and refining camera pose. We estimate depth by building a cost volume over learned features and apply a multi-scale 3D convolutional network for stereo matching. The predicted depth is then sent to the motion module which performs iterative pose updates by mapping optical flow to a camera motion update. We evaluate our proposed system on NYU, KITTI, and SUN3D datasets and show improved results over monocular baselines and deep and classical stereo reconstruction.

1. Introduction

In Video to Depth we are interested in estimating the depth of a given frame in a video sequence. This task has many important applications including autonomous navigation and video understanding in general.

3D reconstruction from video has traditionally been approached from the classical Structure from Motion (SfM) pipeline. SfM uses correspondence between nearby viewpoints to derive 3D structure, simultaneously building a 3D point cloud while estimating camera parameters [33]. This pipeline has shown impressive results on a number of tasks [1, 15] and is often paired with Multi-View Stereo to build a more complete 3D representation [8, 9]. However, this pipeline is limited. Final reconstruction is only as good as correspondence, which is often inaccurate or difficult to obtain, resulting in structurally unlikely artifacts and incomplete reconstruction.

An alternative to the traditional pipeline is deep learning. Given ground truth depth, a deep network can be trained to directly predict depth from either a single image [4, 3, 23] or multiple frames [49, 41] using generic architectures consisting of standard operations such as convolution and pooling. One advantage of deep networks is that they can use monocular cues such as texture gradients and shading, as evidenced by their strong performance on depth estimation from a single image [4, 3, 23]. On the other hand, standard deep networks have been found to have trouble utilizing inter-frame correspondence—simply stacking multiple frames fails to outperform single image depth [49, 41].

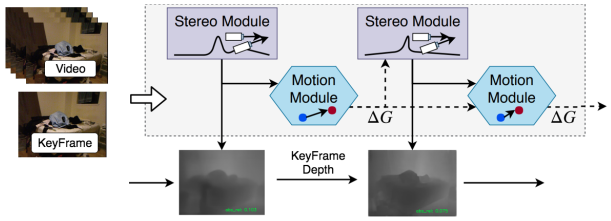

Figure 1. DeepV2D estimates the depth of a keyframe in a video sequence. It is fully differentiable and alternates between depth reconstruction and motion refinement. We perform stereo matching over learned features to reconstruct depth. Our motion module uses the predicted depth to map flow to a motion update ΔG.

In this work we propose a new approach, DeepV2D, that takes the best of both worlds by incorporating algorithmic elements of classic SfM into a deep network. We design a set of differentiable geometric modules based on classical SfM operations. Using these modules along with standard convolutional layers, we compose a video-to-depth system that is fully differentiable and end-to-end trainable.

DeepV2D is advantageous over a generic deep network because with multiview geometry hardcoded into the system, the network only needs to search for the remaining pieces. This smaller search space reduces overfitting and improves generalization. Our approach is advantageous over classic SfM because it uses learned features for correspondence and context to guide reconstruction.

Our main contribution is an end-to-end differentiable architecture that fully incorporates multiview geometry in SfM—we "differentiablize" known algorithms to recover camera motion and 3D structure from correspondence. With such algorithms embedded, the network only needs to search for correspondence. This is different from prior deep-learning based approaches [49, 42, 41, 48], which incorporate geometry to a substantially lesser extent by re-

quiring the network to learn some known geometric operations, e.g. motion estimation from correspondence.

In particular, we introduce a novel differentiable operator, the $\mathbb{SE}3$ transform layer, which uses geometry to map flow and depth into a camera pose update. In addition, we introduce a stereo network estimates depth from a collection of frames and is, for the first time, fully differentiable with respect to all inputs, including camera pose.

Our work is closely related to DeepTAM [48] which estimates depth and motion from video. DeepTAM is the first work to completely replace the components of SLAM—mapping and tracking—with fully learned networks. They produce high quality depth maps and can estimate camera trajectory in challenging environments. But there are several key differences. First, we incorporate geometry to a greater extent in our motion module. We introduce the $\mathbb{SE}3$ transform layer which maps from image motion to camera motion directly, while DeepTAM requires a network to learn this mapping. Second, our work is end-to-end differentiable and trained jointly whereas DeepTAM trains their modules in isolation and is not fully differentiable. Furthermore, we leverage learned features for stereo reconstruction by building a cost volume over learned features, while DeepTAM uses a fixed similiarity metric.

We evaluate DeepV2D on three separate datasets. On NYU depth [35] we substantially improve depth predictions over our monocular baseline and classical SfM results. On KITTI we outperform all other stereo based approaches and monocular depth estimation networks not trained with additional data. We evaluate against DeepTAM on the SUN3D dataset and show improved results.

2. Related Work

Structure from Motion: Beginning with early systems designed for small image collections [24, 27], Structure from Motion (SfM) has improved dramatically in regards to robustness, accuracy, and scalability. Advances have come from improved features [25, 14], optimization techniques [36], and more scalable data structures and representations [34, 11], culminating in a number of robust systems capable of large-scale reconstruction task [34, 37, 45]. However, SfM is limited by the accuracy and availability of correspondence. In low texture regions, occlusions, or lighting changes SfM can produce noisy or missing reconstructions or fail to converge entirely.

Simultaneous Localization and Sapping (SLAM) jointly estimates camera motion and 3D structure from a video sequence [6, 28, 29, 30]. LSD-SLAM [6] is unique in that it relies on a featureless approach to 3D reconstruction, directly estimating depth maps and camera pose by minimizing photometric error.

We replace both the motion estimation and mapping components of SfM and SLAM with neural network modules. SfM and SLAM are limited by their use of handcrafted features. Our trainable modules are able learn features suited for the tasks of motion and depth estimation, while retaining the geometric principles of SfM. Unlike SfM and SLAM, we can learn priors over 3D structure from large RGB-D datasets. Because our network is fully differentiable, it can readily be used in conjunction with standard deep learning components. We avoid the need for hand-engineered iterative re-weighting schemes to deal with occlusions or moving objects and instead let the network decide which image regions are important.

Single Image Depth Estimation: There has been a lot of recent interest in estimating 3D properties such as depth and surface normals from a single image. Eigen et al. [4] first showed that deep convolutional networks could be trained directly on raw pixels to estimate depth from a single image. This network was able to use monocular features alone to recover depth. Later deeper network architectures further improved performance [3, 23].

In our case, we are interested in estimating depth from a video sequence. Single-image depth networks can be readily applied to this task, but in doing so, they are not able to use motion to guide reconstruction. Our approach retains the advantages of single image depth networks, while also being able to use of the motion parallax present in videos.

Geometry and Deep Learning: Geometric principles have been a guiding force for many deep learning architectures. Convolutional networks have been particularly successful at stereo matching [14, 26, 21]. Kendall et al. [21] built a 3D cost volume over 2D feature maps by sampling from a range of disparities. Kat et al. introduced LSM [18] and showed that similar ideas could be applied to reconstruct objects from multiple viewpoints. DeepV2D retains the geometric principles of these works, but is able to reconstruct scenes from video without known camera pose. Furthermore, unlike LSM which is limited to objects due to its choice of a Euclidean reconstruction grid, we parameterize reconstructon by camera frustum coordinates, enabling us to reconstruct challenging indoor and outdoor scenes.

Camera motion estimation with deep neural networks has generated a lot of recent interest. Kendall et. al [20, 19] focused on the problem of camera localization, while other work [49, 42, 44, 48, 41] aim to estimate camera motion between a pair of frames. These approaches all treat camera pose as a regression problem by training a network to output the parameters of the camera motion matrix. Most related to our work is DeepTAM [48] which is unique in that it estimates camera motion iteratively, where each new motion estimate use used to render the target frame onto the keyframe. This greatly improves tracking performance and generalization. Although, like previous work on deep motion estimation, DeepTAM requires a neural network to predict camera motion. These networks must learn to map

image motion into a camera motion update. We propose the $\mathbb{SE}3$ transform layer which translates motion estimation into a correspondence problem. Unlike prior work, our motion module only needs to learn optical flow.

Furthermore, unlike DeepTAM whose components are not fully differentiable and trained in isolation, our system is end-to-end differentiable since we construct the cost volume over learned features using differentiable bilinear sampling. This allows our DeepV2D in its entirety to be jointly trained end-to-end. By using learned features for reconstruction, our stereo module can learn a robust feature representation along with contextual information to facilitate matching and reconstruction.

Several geometric optimization problems have recently been formulated as differentiable network modules. Wang et al. [43] proposed a differentiable network operator which estimates camera motion by minimizing photometric error. BA-Net [39] applies a differentiable implementation of the Levenberg-Marquet(LM) algorithm to solve for camera pose and depth jointly by minimizing reprojection error in a learned feature space. These works require optimization to be performed by photometric alignment which is often highly non-convex [5]. Our $\mathbb{SE}3$ transform layer predicts the residual term directly which results in a simplified optimization problem.

We additionally decompose reconstruction into stereo matching and motion estimation. While our final depth estimate is a product of stereo matching, BA-Net estimates depth as weighted combination of basis depth estimates produced by a single image depth network.

3. Approach

DeepV2D predicts depth from a video sequence. We take a collection of frames, plus a given keyframe, and predict a dense depth map. While DeepV2D predicts the depth for just a single frame from a video, it an easily be extended to output the depth for any collection of frames.

We decompose depth estimation into two separate subproblems which we solve using neural network modules. First, given our image sequence and camera motion estimates, we can reconstruct depth with stereo reconstruction. Our *Stereo Module* performs stereo reconstruction from a collection of images with camera motion estimates. The Stereo Module requires camera motion as an input. To this end, we estimate camera motion using our *Motion Module* which takes the keyframe depth as input. In the forward pass, we alternate between the stereo and motion modules as we show in 1. In this work, we assume known camera intrinsics (i.e. the camera is calibrated).

3.1. Camera Geometry and View Synthesis

As a prelimary, we define some of the operations used within the stereo and motion modules. We represent camera motion using 3D rigid body transformations. A rigid body transformation $G \in \mathbb{SE}(3)$ describes rotation and translation in 3D:

$$G = \begin{pmatrix} \mathbf{R} & t \\ 0 & 1 \end{pmatrix} \text{ where } \mathbf{R} \in SO(3) \text{ and } t \in \mathbb{R}^3 \quad (1)$$

Furthermore, a rigid body transform can act to transform a 3d point $\mathbf{X} = (X, Y, Z, 1)$: $\mathbf{X}' = G \cdot \mathbf{X}$.

The camera operator $\pi : \mathbb{R}^4 \mapsto \mathbb{R}^2$ projects a 3D point $\mathbf{X} = (X, Y, Z, 1)$ to a pixel $\mathbf{x} = (x, y)$:

$$\pi(\mathbf{X}) = (f_x \frac{X}{Z} + c_x, f_y \frac{Y}{Z} + c_y) \quad (2)$$

where (f_x, f_y, c_x, c_y) are the camera intrinsics. Likewise, given depth z we can recover a 3D point in homogeneous coordinates using backprojection $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \mapsto \mathbb{R}^4$:

$$\pi^{-1}(\mathbf{x}, z) = (z\frac{x - c_x}{f_x}, z\frac{y - c_y}{f_y}, z, 1)^T \quad (3)$$

With π and π^{-1} we can define the projective warping function $w : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{SE}(3) \mapsto \mathbb{R}^2$ which maps a point p with depth z to a camera transformed by G:

$$w(\mathbf{x}, z, G) = \pi(G \cdot \pi^{-1}(\mathbf{x}, z)) \quad (4)$$

All the equations defined are fully differentiable with respect to all inputs and can readily be used in conjunction with standard network layers.

As a final note, we can apply Equation 4 to render an entire image from a synthetic viewpoint provided the camera transform matrix G and depth map D. Letting $\tilde{I}_{ij}$ be (i, j) pixel in the rendered frame, we can compute its value by using Equation 4 to compute its location in the reference frame, and sample from the projected coordinate:

$$\tilde{I}_{ij} = I(w(\mathbf{p}, D_{ij}, G)), \quad \text{where } \mathbf{p} = (i, j) \quad (5)$$

Here $I(\cdot)$ denotes the sampling operation (note that the projected points are continuous values). We choose to use differentiable bilinear sampling which was proposed in *spatial transformer networks* [17]. Differentiable bilinear sampling computes the value of a point by interpolating from its 4-pixel neighbors with weights determined by proximity. By choosing differentiable bilinear sampling we can backpropogate the gradient through the entire rendering process with respect to all inputs (i.e. depth, pose, reference image).

3.2. Stereo Module

Given a set of T frames $\mathbf{I} = \{I_1, I_2, ..., I_T\}$ and their respective estimated poses $\mathbf{G} = \{G_1, G_2, ..., G_T\}$ our stereo module predicts a dense depth map for keyframe image I^* which we will denote D^*. Each pose, $G_t \in \mathbb{SE}3$ represents the transformation from the keyframe camera to the camera at frame t. Hence, a point in the keyframe p with depth z can be mapped to its location in frame t using the projective warping function defined in Equation 4: $p_t = w(p, z, G_t)$.

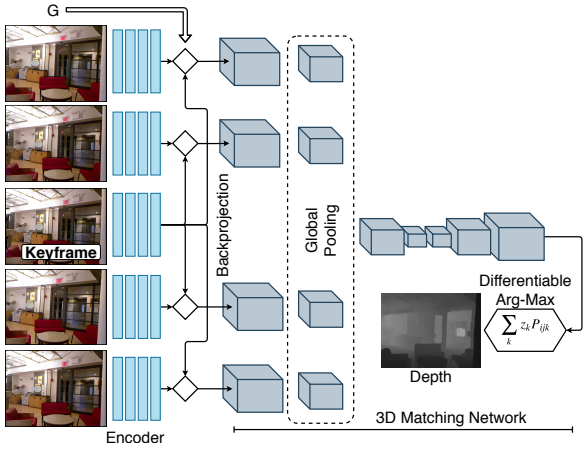

Figure 2. The *Stereo Module* estimates keyframe depth. First, each image is fed through a network to generate a learned 2d feature representation. We then form a 3d cost volume for each reference frame by backprojecting 2d features over a set of depth hypothesis. After several 3d convolutional layers, we perform global pooling over the set of reference frames. Finally, we apply a 3d hourglass network to produce a final cost volume which is mapped to the final depth map with differentiable arg-max.

Two-View Reconstruction: We first consider the case of two view stereo reconstruction between the image pair (I^*, I_t) before showing how we can generalize to an arbitrary number of frames. Given the keyframe image I^* and the reference image I_t we start by feeding each image through a convolutional neural network to generate a dense unary feature maps F^* and F_t. We call this network the encoder, and the weights of the encoder are shared for each image. The purpose of the encoder is to learn a dense feature representation which will provide context and facilitate stereo matching.

The stereo module constructs a cost volume from the generated feature images. The cost volume is a stack of feature maps, each rendered from the viewpoint of the keyframe camera. We enumerate over a set of hypothesis depths $\mathbf{Z} = \{z_1, z_2, ..., z_D\}$ which span the ranges observed in the dataset. For each depth z_k, we render the feature map F_t from the keyframe camera using Equation 5, assuming a planar scene of depth z_k, generating the warped feature map $\tilde{F}_t$. We concatenate $[F^*, \tilde{F}_t]_k$ along the channel axis to form the k^{th} entry in the cost volume. The final cost volume is formed by stacking the D rendered viewpoints into a single 4D tensor C_t. Hence, if F^* has dimensions (H, W, C), then the dimension of the fully constructed cost volume will be $(H, W, D, 2C)$.

The cost volume is a powerful representation for stereo reconstruction and converts depth estimation into a matching problem. We perform matching and refinment with a 3D convolutional neural network. Our network consists of an encoder composed of 3x3x3 convolutional layers which subsample the spatial resolution, and a decoder which upsamples the spatial resolution. The encoder and decoder are connected with skip connections by performing elementwise addition of the feature maps. The overall architecture of the 3D matching network is similiar to the hourglass network [32] with 2D layers replaced with 3D convolutions.

Decoder: The output of the matching network is a volume of dimension $P \in \mathbb{R}^{H \times W \times D}$. The elements of the volume represent the likelihood of a surface. We first perform softmax over the depth dimension to convert surface liklihood to a probability of depths. In other words, P_{ijk} represent the probability of pixel (i, j) having depth z_k.

We convert the probability volume into a single depth estimate using the differentiable argmax function [21]. A pointwise depth estimate is found by finding the expected depth—computed by taking the sum of each depth multiplied by its corresponding probability:

$$D^*_{ij} = \sum_k z_k P_{ijk} \tag{6}$$

Multiview Reconstruction: Our reconstruction pipeline can easily be extended to more than one keyframe image pair to improve performance. For each keyframe image pair, (I^*, I_t) for $t = 1, ..., T$, we compute the cost volume C_t. Each volume is first processed by 4 3x3x3 convolutional layers with shared weights before a global pooling step as shown in 2. The global pooling step aggregates information across viewpoints by averaging the feature maps of the T volumes.

Our stereo network shares many similarities with classical multiview-stereo pipelines which employ a cost volume to reconstruction depth. Although, our approach has two key advantages over classical techniques. First, we begin by processing the features maps through 2D CNNs to generate a dense features. Instead of using hand-crafted features, the 2D network can learn feature representations which are more robust and easier to match. Furthermore, our 3D matching network is able to learn a similarity metric between feature vectors while using contextual information to refine the reconstruction. Classical work relies on much simpler priors like smoothness assumptions.

3.3. Motion Module

The input to our motion module is the keyframe image/depth pair (I^*, D^*) and the video frames $\mathbf{I} = \{I_1, I_2, ..., I_T\}$. The motion module estimates the motion between each keyframe image pair, (I^*, I_t) for $t = 1, ..., T$ as we show in Figure 3. The final output of the motion module is the set of rigid body transformations $\mathbf{G} = \{G_1, G_2, ..., G_T\}$.

The motion module considers each keyframe image pair independently, and operates in parallel for each of the T pairs. For the remainder of this section, we describe the operation for the pair (I^*, I_t) but keep in mind that it works the same way for each of the other pairs.

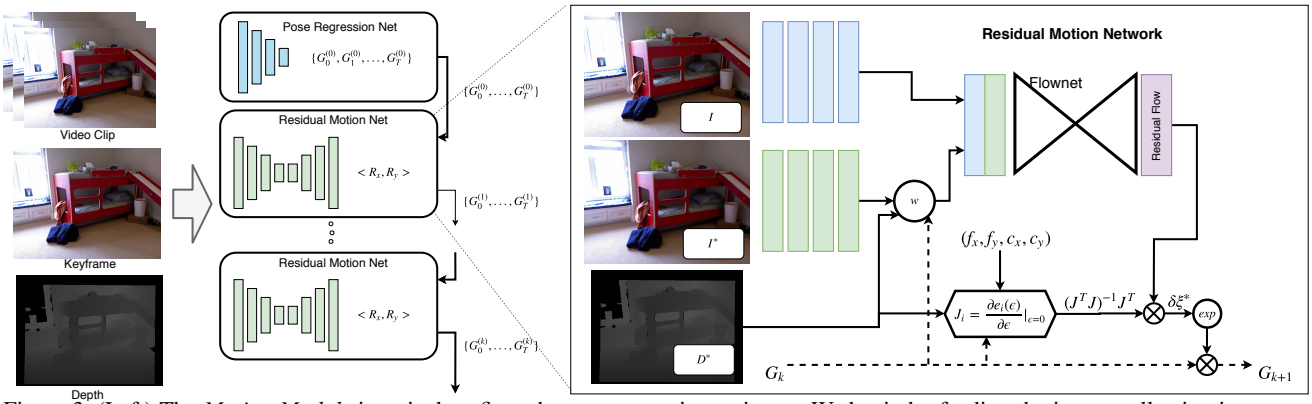

Figure 3. (Left) The *Motion Module* iteratively refines the camera motion estimate. We begin by feeding the image collection into a pose regression network to generate an initial motion estimate between the target image and each reference image. (Right) We then predict the residual motion between the warped feature images which is mapped to a pose update in the residual motion unit. The *Motion Module* proceeds iteratively, improving motion estimates with each iteration.

Initialization: To generate an initial motion estimate, we simply stack the frames and train a network to predict the transformation parameters $(r_x, r_y, r_z, t_x, t_y, t_z)$. This is in line with previous work and what we refer to as the pose regression network. The results of the pose regression network are coarse and far from accurate enough for stereo reconstruction. Regardless, we found the pose regression network to be good starting point to initialize our system.

Iterative Refinement: The initial motion estimate, $G_t^{(0)}$, is used as the starting point for further refinement. Like the stereo module, we begin by extracting a learned feature representation for the keyframe image pair to produce the feature images F^* and F_t. Again the weights of the feature extractor are shared across all images.

Given the keyframe depth estimate D^* and the current motion estimate $G_t^{(k)}$, we can render F_t from the estimated viewpoint of the keyframe camera using Equation 5 (applying the projective warping function $w(p, D^*, G_t^{(k)})$) to produce the warped feature map $\tilde{F}_t$.

The objective of the motion module is to find a transformation which aligns $\tilde{F}_t$ and F^*. Each iteration of the motion module takes the rendered feature image pair $(\tilde{F}_t, F^*)$ and produces a transformation which updates the current motion estimate. As the motion estimate improves, the feature images become more and more aligned, resulting in smaller incremental updates. Instead of relying on an initial motion estimate, we can test how each motion estimate agrees with the inverse projection and propose updates to correct the error.

We use lie-algebra elements $\xi = (\mathbf{v}, \mathbf{w}) \in se(3)$ to parameterize camera motion updates. A element ξ can be mapped to $\mathbb{SE}3$ with the exponential mapping $exp_{\mathbb{SE}3} : se(3) \mapsto \mathbb{SE}3$ [38]. The $\mathbb{SE}3$ group operator is matrix multiplication. Given the vector $\delta\xi$ we can perform a camera motion update on $G_t^{(k)}$:

$$G_t^{(k+1)} = e^{\delta\xi} \cdot G_t^{(k)} \tag{7}$$

Residual Motion Network: We can now describe in greater detail how the motion updates are computed. Starting with keyframe depth D^*, feature images (F^*, F_t), and the motion estimate $G_t^{(k)}$, our residual motion network generates a motion update element $\delta\xi$. As shown in Figure 3, the residual motion network is applied iteratively.

In the iteration step, we estimate the optical flow between the rendered image $\tilde{F}_t$ and F^* which we term the residual flow—denoted R. To estimate flow, we concatenate the feature maps $[\tilde{F}_t, F^*]$ use an encoder-decoder network with skip connections modeled after FlowNetS [2]. This network is *not* directly supervised on flow, but instead the flow is used as an intermediate to produce the motion update.

The residual flow tells us the 2D motion between the rendered feature images. Our residual motion network produces a update to correct this motion. We propose the $\mathbb{SE}3$ transform layer which is differentiable and maps the 2D motion into a 3D rigid body update. More formally, the input to this layer is the residual flow R, the keyframe depth D^*, and the camera parameters. The output is a lie-algebra element $\delta\xi \in se(3)$ which we use to perform a motion update following Equation 7.

For a given keyframe point p_i, we can find its location in reference frame $p_i' = w(p_i, D^*(p_i), G_t^{(k)})$. We define $p_i''(\delta\xi)$ to be the reprojected point under a camera transformation: $p_i''(\delta\xi) = w(p_i, D^*(p_i), e^{\delta\xi} \cdot G_t^{(k)})$. We find a motion update such that the distance between $p_i''(\delta\xi)$ and p_i' matches the residual flow. We formalize this as the following objective function:

$$\begin{aligned} E(\delta\xi) &= \sum\nolimits_i ||e_i(\delta\xi)||_2^2, \text{ where} \\ e_i(\delta\xi) &= (p_i''(\delta\xi) - p_i') - R_i \end{aligned} \tag{8}$$

Since this is a sum of squares, an update can be proposed with a Gauss-Newton iteration:

$$\delta\xi = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T R \qquad J_i = \frac{\partial e_i(\epsilon)}{\partial \epsilon}|_{\epsilon=0} = \frac{\partial p_i''(\epsilon)}{\partial \epsilon}|_{\epsilon=0} \tag{9}$$

where $\mathbf{J} = [J_1, J_2, ..., J_n]$ is the stack of derivatives.

Here $\mathbf{J}$ depends only on the depth, $G_t^{(k)}$, and the camera intrinsics; thus, the term $(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$ is independent of the residual flow. Taking this term to be constant, the motion update is simply a linear mapping of the residual flow, encoding geometry and allowing the gradient to be easily backpropogated through the single Gauss-Newton iteration. This property means we do *not* need any intermediate supervision on flow.

In practice, we don't need dense flow to compute camera motion. We train the network to output 2 additional channels $\mathbf{W} = (W_x, W_y)$ constrained to $[0, 1]$ with the sigmoid activation. W_x, W_y weight the the residual in the respective x,y directions. We update Equation 8 accordingly

$$E(\delta\xi) = \sum_i e_i \text{ diag}(W_i)\ e_i^T \tag{10}$$

Again, we provide no intermediate supervision. The network learns on its own to weight different image regions.

Deep learning has presented a new alternative for estimating camera motion. By predicting motion in the image space, our network doesn't need to learn the geometric relationships between depth, camera motion, and optical flow. We can apply geometric knowledge by directly mapping depth and optical flow to a camera motion estimate. Furthermore, our formulation is more general than methods which predict camera parameters directly. Our minimization step takes as input the camera intrinsics, meaning that as long as the optical flow is accurate, our method can generalize to situations with different image dimensions and camera parameters.

3.4. Supervision

Depth Loss: We supervise depth on the mean absolute error between the predicted and ground truth depth over the set of pixels with valid ground truth depths denoted $V(D_{gt})$.

$$L_{depth} = \frac{1}{|V(D_{gt})|} \sum_{p \in V(D)} |\hat{D}(p) - D_{gt}(p)| \tag{11}$$

We also include a small L1 smoothness term on the pixels where depth is missing

$$L_{smooth} = \sum_{p \notin V(D_{gt})} |\partial_x \hat{D}(p)| + |\partial_y \hat{D}(p)| \tag{12}$$

giving the loss: $L_{structure} = L_{depth} + \lambda_s L_{smooth}$.

Motion Loss: We craft a loss function which avoids the need to consider rotation and translation independently and instead penalize the network on reprojection error directly. During the forward pass, our network outputs a sequence of pose estimates $\{G_t^{(1)}, ..., G_t^{(K)}\}$ for each of the T images. We define the loss between two poses to be the mean huber-distance of the reprojected points with $\delta = 1$:

$$d(G_1, G_2|D) = \frac{1}{|V(D)|} \sum_{p \in V(D)} ||e_p||_\delta$$
$$\text{where } e_p = w(p, D, G_1) - w(p, D, G_2) \tag{13}$$

The reprojection error is taken to be the sum of errors between the predicted and ground truth pose over each sequence and image:

$$L_{rep} = \sum_{t=1}^{T} \sum_{k=1}^{K} d(\hat{G}_t^{(k)}, G_t^{(gt)}|D_{gt}) \tag{14}$$

Additionally, we want the predicted residual flow to match the motion update. We add a regularization penalty on the residual flow following each Gauss-Newton update by penalizing on the weighted squared error term $E(\xi)$ in Equation 10. Finally, we want to avoid the degenerate case where all flow weights become 0. We take the logloss of the top K weights. The final motion loss is the combination:

$$L_{motion} = L_{rep} + \lambda_E E(\xi) + \lambda_w L_{weight} \tag{15}$$

4. Experiments

We test our approach on NYUv2 [35] and the KITTI [10] datasets and compare to both classical SfM and monocular depth estimation approaches. We apply the following 2-stage training approach. We perform data augmentation by adjusting brightness, gamma, and performing random scaling of the image channels.

Stage I: We train the Motion Module using the L_{motion} loss with RMSProp [40]. For the input depth, we use the ground truth depth with missing values filled in with nearest neighbor interpolation.

Stage II: In stage II, we jointly train the Motion and Stereo modules end-to-end on the combination of motion and depth loss terms $L_{motion} + L_{structure}$ with RMSProp. Again, DeepV2D requires an additional depth estimate. For each training instance, we choose between two options: (1) use the ground truth depth or (2) use the depth predicted last time this training instance was encountered. During training we decay the probability of ground truth initialization.

Timing Information: On the NYU dataset our system operates at 340ms per iteration for a 5 frame video with 480x640 input resolution. For 5 192x1088 frames on KITTI, DeepV2D runs at 230ms/iteration.

NYU Training: We experiment on the NYU depth dataset [35] using the standard Eigen train/test split [4]. NYU provides a challenging benchmark to test our approach. Unlike other datasets such as KITTI where camera motion is mostly planar, NYU exhibits more complex motion which span all degrees of freedom.

We train Stage I for 50k iterations with a batch size of 4, and train Stage II for 160k iterations with a batch size of 1. We set the number of residual iterations in the motion

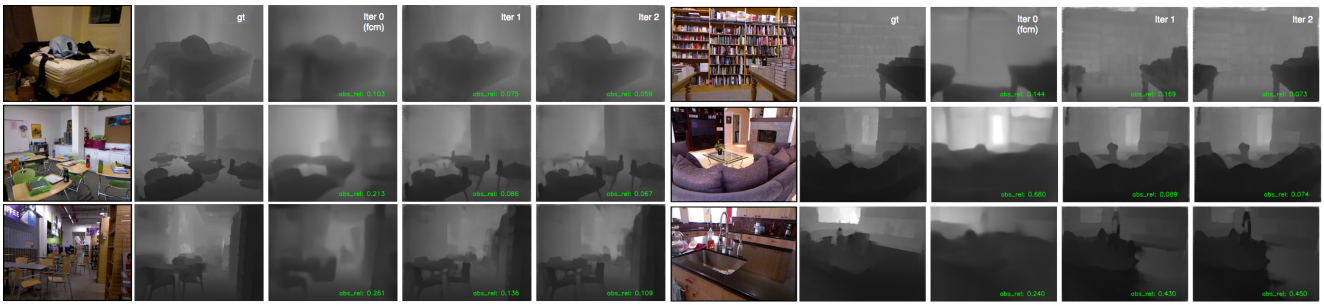

Figure 4. DeepV2D is initialized with a single-image depth prediction from [23] which is refined with each iteration. Here we show the successive refinement with each iteration with the scale matched abs rel metric. DeepV2D is often able to make substantial corrections to the initial depth estimate and add a significant level of detail. In the final example we show a case where we fail to recover the depth of a specular surface. Overall, DeepV2D produces accurate and detailed depth reconstructions and substantially improves the initial single-image prediction.

	Lower is better				Higher is better		
NYUv2	rel	RMSE	RMSE log	sc-inv	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al [4]	0.215	0.907	0.285		0.611	0.887	0.971
Eigen and Fergus [3]	0.158	0.641	0.214	0.148	0.769	0.950	0.988
Laina et al [23]	0.127	0.573	0.195	-	0.811	0.953	0.988
DeMon et al [41]	-	-	-	0.180	-	-	-
Ours	**0.118**	**0.537**	**0.173**	**0.119**	**0.858**	**0.976**	**0.994**

Table 1. Comparison of proposed approach to monocular depth estimation networks on NYU. We outperform the baseline monocular network [23] which we use to initialize our method.

	iters	views	rel ↓	$\delta < 1.25$↑	$\delta < 1.25^2$↑	$\delta < 1.25^3$↑
colmap [34]	7		0.404	0.549	0.700	0.775
DfUSMC [13]	7		0.448	0.487	0.697	0.814
(fcrn)	0	-	0.125	0.843	0.963	0.991
ours	2	2	0.115	0.869	0.970	0.991
ours	2	3	0.097	0.905	0.981	0.994
ours	2	5	0.089	0.920	**0.984**	**0.996**
ours	2	7	**0.088**	**0.922**	**0.984**	**0.996**

Table 2. We evaluate DeepV2D against classical SfM where global scale ambiguity is resolved with median matching. Additionally, we test the perfomance of our system as we increase the number of views. We compare favorably to classical SfM and see improved results by increasing the number of frames.

module to again be 3. During training, we sample a set of target frames uniformly from the raw distribution. For each target frame, we sample 6 neighboring frames spaced approximately 0.25s apart. At each training iteration, we randomly sample 3 of the 6 frames. We use the full 480x640 images. NYU does not have ground truth camera pose data, but we are able to generate good estimates applying RGB-D SLAM [29]. At test time, we initialize the depth estimate with a single-image depth network [23].

NYU Results: We show some example NYU results of DeepV2D in Figure 4. We are able to add a significant level of detail over the baseline monocular network and often make large corrections. Like classical SfM, reflective surfaces are difficult to recover. Overall, DeepV2D produces accurate and detailed depth reconstructions.

In Table 1 we compare to single-image depth estimation networks including the baseline single-image initialization FCRN [23]. For reference, we also include the networks from Eigen and Fergus [3], and DeMon [41] which uses

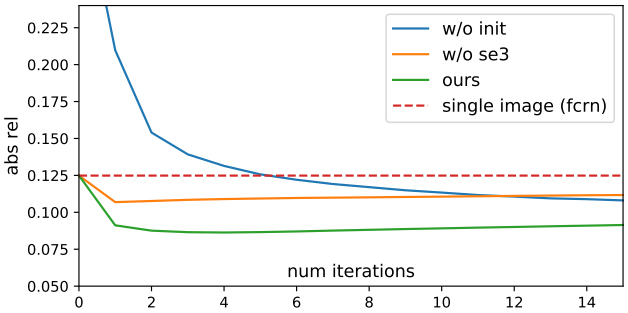

Figure 5. We test the convergence properties of DeepV2D under different conditions. Our baseline system is initialized with a single-image depth estimate from [23]. (no init) tests the convergence when DeepV2D is instead initialized with a flat depth estimate of 5m. (w/o se3) replaces the SE3 transform layer with a network head which predicts the motion update directly. Although (w/o init) takes longer to converge, it can still produce accurate depth estimates even with a simple initialization strategy. (w/o se3) indicates that our SE3 transform layer is a critical component.

two frames to reconstruct depth. However, DeMoN was not trained directly on NYU due to insufficient supervision. We outperform the baseline network improving the challenging $\delta < 1.25$ metric from 0.811 to 0.858.

For a direct comparison with classical SfM, we perform median matching as done in [49] to resolve global scale ambiguity (Table 1). We gather classical SfM results with the publicly available colmap [34] and DfUSMC [13], fixing camera intrinsics to the calibrated values. Both [34] and [13] are able to generate accurate and highly detailed reconstructions on many of the test images; however, they struggle to recover low texture scenes, producing large final errors. By using learned features and structural priors, DeepV2D can circumvent many of these failure cases.

NYU Ablations: DeepV2D introduces an iterative method for depth estimation. In Figure 5 we look at the convergence properties of our proposed system by plotting the scale matched absolute relative error (abs rel) as a function of the number of motion/stereo module iterations. The baseline model is initialized with a single image depth estimate

	KITTI Raw	Stereo	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	Abs Rel $\downarrow$	Sq Rel $\downarrow$	RMSE $\downarrow$	RMSE log $\downarrow$
	Mean		0.593	0.776	0.878	0.403	5.530	8.709	0.403
Extra Data	Eigen et al. [4] Fine		0.702	0.890	0.958	0.203	1.548	6.307	0.282
	Goddard et al. [12] (+City Scapes)		0.861	0.949	0.976	0.114	0.898	4.935	0.206
	Kuznietzov et al. [22]		0.862	0.960	0.986	0.113	0.741	4.621	0.189
	DORN (vgg) [7]		0.915	0.980	0.993	0.081	0.376	3.056	0.132
	DORN (resnet101) [7]		**0.932**	**0.984**	**0.994**	**0.072**	**0.307**	**2.727**	**0.120**
From Scratch	Goddard et al. [12]		0.803	0.922	0.964	0.148	1.344	5.927	0.247
	Yang et al [47]		0.888	0.958	0.980	0.097	0.734	4.442	0.187
	DfUSMC [13]	Y	0.617	0.796	0.874	0.346	5.984	8.879	0.454
	Ours (no stereo)		0.831	0.942	0.977	0.135	0.949	4.932	0.210
	Ours	Y	**0.923**	**0.970**	**0.987**	**0.091**	**0.582**	**3.644**	**0.154**

Table 3. Comparison of proposed approach on the KITTI Raw Dataset using the testing split proposed by [4]. We outperform classical SfM using DfUSMC [13] and all methods not trained using external data. We are competitive with DORN [7] on the more robust δ metrics even though DORN is initialized with a pretrained resnet model. We also train DeepV2D with stereo cues disabled showing that while our model has the capacity to learn single image depth, stereopis adds a large boost in performance.

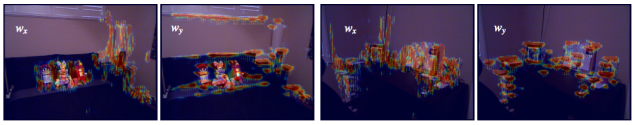

Figure 6. We overlay a heat map of the residual flow weights on example images. The w_x represents the confidence in the x-direction and learns to up-weight vertical edges and salient image regions.

from [23]. We test a version where we instead initialize with a flat depth estimate of 5m (labeled w/o init). While convergence is slower, we can still recover good depth estimates with enough iterations. Additionally, we test a version of our system where we replace the $\mathbb{SE}3$ transform layer with a network head which predicts the camera motion update directly, keeping all other components the same. Results show that this hurts performance, indicating that our $\mathbb{SE}3$ layer is beneficial for accurate depth estimation.

We can also visualize the regions which the motion module attends to in Figure 6. The motion module predicts two residual flow weights w_x and w_y which reflect the confidence of the flow vector in the respective directions and learns to up-weight edges and salient image regions.

KITTI: For completeness, we evaluate different variants of our proposed approach on the KITTI driving benchmark [10] and compare to single-image and stereo based approaches. For testing we follow the Eigen train/test split proposed in [4]. The KITTI dataset contains many dynamically moving objects presenting a challenging scenario for stereopis.

We use an input sequence of 5 images for our network formed by taking the two closest frames before and after the target image. We generate ground truth depth by reprojecting 3D velodyne points onto the left color camera and resize images to 292x1088 pixels and crop the top 100 pixels for an input size of 192x1088, and ground truth motion from the gps/imu files.

We train Stage I for 20k iterations with a batch size of 4, and train Stage II for 200k iterations with a batch size of 1. We set the number of residual iterations in the motion module to be 3. At test time we bootstrap the reconstruction process by using the output from the pose regression network, which is trained with all 5 frames as input.

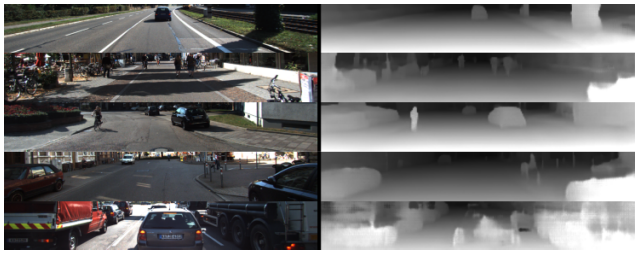

Figure 7. Visualization of our estimated depth maps on KITTI. Like classical methods, we often fail to reconstruct dynamic scenes as shown in the final example. Overall, we are able to recover a lot of detail in the depth maps and preserve thin structures such as poles and trees.

We provide quantitative results in Table 3. Goddard et. al [12] Yang et al [47] trains single-image depth networks supervised on photo-consistency measure between stereo pairs. Kuznietzov et al. [22] combines photo-consistency with velodyne data. DORN [7] is a single-image network which is initialized with a pretrained resnet model. We outperform our classical baseline and all methods not using external data. We are competitive with DORN on the outlier robust $\delta < 1.25$ metric.

We test a baseline model where we disable stereo cues by training the network on videos containing only the keyframe. This result shows that while our network has the capacity to estimate depth using no stereo information, stereopis greatly improves results, reducing the $\delta < 1.25$ error by more than 50%.

In Figure 7 we provide some visualizations of DeepV2D depth predictions. We are able to recover a significant level of detail, including thin structures such as poles and trees.

SUN3D, DeepTAM Comparison: We compare the performance of our method to DeepTAM [46] on the SUN3D dataset. We follow the setup in DeepTAM where depth estimation is tested in isolation, using the camera motion provided in the dataset. For DeMon, SGM, and DTAM we report the results as provided by [48]. As shown in Table 4

	L1-inv	L1-rel	sc-inv
DTAM	0.210	0.423	0.374
SGM	0.197	0.412	0.340
DeMoN	-	-	0.146
DeepTAM	0.064	0.111	0.130
Ours	**0.056**	**0.105**	**0.124**

Table 4. Isolated mapping perfomance on the SUN3D dataset. We outperform classical mapping systems like SGM and DTAM. We additionally perform better than DeepTAM.

show, we outperform classical mapping systems like Semi-Global mapping [16] and DTAM [31]. DeepTAM uses 3 networks to recover depth including a single-image refinement network. We outperform DeepTAM despite using only a single network and applying no refinement.

5. Conclusion

We propose DeepV2D, an end-to-end differentiable system which applies both monocular and geometric cues to predict 3D structure from a video sequence. DeepV2D is built from a set of geometric modules based on classical SfM operations and applies stereopis to recover depth.

Acknowledgements We would like to thank Zhaoheng Zheng for helping with baseline experiments. This work was partially funded by the Toyota Research Institute.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[3] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[4] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.

[6] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[7] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[8] Y. Furukawa, C. Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[9] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[11] R. Gherardi, M. Farenzena, and A. Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1594–1600. IEEE, 2010.

[12] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[13] H. Ha, S. Im, J. Park, H.-G. Jeon, and I. So Kweon. High-quality depth from uncalibrated small motion clip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5413–5421, 2016.

[14] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3279–3286. IEEE, 2015.

[15] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3287–3295, 2015.

[16] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[18] A. Kar, J. Malik, and C. Häne. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems*, pages 364–375, 2017.

[19] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proc. CVPR*, volume 3, page 8, 2017.

[20] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2938–2946. IEEE, 2015.

[21] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. *arXiv preprint arXiv:1703.04309*, 2017.

[22] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017.

[23] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[24] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[26] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. 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*, pages 4040–4048, 2016.

[27] R. Mohr, L. Quan, and F. Veillon. Relative 3d reconstruction using multiple uncalibrated images. *The International Journal of Robotics Research*, 14(6):619–632, 1995.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[29] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[30] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[31] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[32] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[33] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[34] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012.

[36] K. N. Snavely. Scene reconstruction and visualization from internet photo collections. 2009.

[37] N. Snavely. Scene reconstruction and visualization from internet photo collections: A survey. *IPSJ Transactions on Computer Vision and Applications*, 3:44–66, 2011.

[38] H. Strasdat, J. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010.

[39] C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

[40] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[41] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. *arXiv preprint arXiv:1612.02401*, 2016.

[42] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[43] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.

[44] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.

[45] C. Wu et al. Visualsfm: A visual structure from motion system. 2011.

[46] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[47] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. *arXiv preprint arXiv:1807.02570*, 2018.

[48] H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. *arXiv preprint arXiv:1808.01900*, 2018.

[49] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. *arXiv preprint arXiv:1704.07813*, 2017.