LONDON'S GLOBAL UNIVERSITY
Department of Computer Science



Multi-View Stereo from Keyframe Depth and Optical Flow

Faculty of Engineering

Department of Computer Science

University College London

A Thesis Presented in Partial Fulfillment of the Degree

MSc Computer Graphics, Vision and Imaging

September 2025

Abstract

3D tasks like SfM and MVS are some of the classical problems in computer vision. For long, they have been solved by carefully crafted pipelined stages that are both time- and compute-intensive. More recently, the advent of deep learning has introduced novel methods that leverage large-scale datasets and hardware to directly solve these problems in a single forward pass. However, the substantial data and GPU requirements remain a barrier for many. In this report, we propose a training strategy that does not rely on RGB images, instead taking depth and optical flow estimates as input, and show that this helps our model generalise to scenarios outside the training distribution. We also introduce a novel method to estimate camera poses in a single forward pass, eliminating the need for post-processing optimization. We modify DUSt3R to accept depth and optical flow estimates as input and predict camera ray bundles along with point maps. We show that our model is competitive with state-of-the-art results in dense 3D prediction and camera pose estimation while using $\sim$1% of the training data and reducing inference time due to our method not requiring post-processing optimization.

General methods that leverage computation are ultimately the most effective.

— Richard Sutton

Table of Contents

List of Figures iii

List of Tables iv

1 Introduction and Background 1

2 Related Work 3

2.1 Structure From Motion . . . 3

2.2 Multi-View Stereo . . . 3

2.3 Ray-Based Camera Parametrization . . . 4

3 Methodology 5

3.1 Problem Setup and Notation . . . 5

3.2 Recovering Camera Pose . . . 7

3.3 Architecture . . . 8

3.4 Training . . . 11

4 Results and Analysis 14

4.1 Evaluation Method and Metrics . . . 14

4.2 Results . . . 15

4.3 Limitations . . . 16

4.4 Visual Results . . . 17

5 Conclusion 21

References 22

Appendix A Source Code 29

List of Figures

3.1 Our model's architecture. We use separate encoders to account for the different input modalities and the camera head takes tokens from both decoders as input to predict the relative pose of the second image. 10

4.1 Inference results of our model on a pair of images from the BlendedMVS dataset. Here, we show the predicted point cloud without any meshing. . 17

4.2 Inference results of our model on a set of 16 images (5 shown) from the BlendedMVS dataset with meshing. On the left we visualise an example of what the output might see after processing with keyframe depth and optical flow estimates with the middle frame chosen as the keyframe. . . 18

4.3 Inference results of our model on a set of 16 images (5 shown) from the 7 scenes dataset with meshing. On the left we visualise the actual RGB inputs before the depth and optical flow estimates. 19

4.4 A failure case on a pair of images from the 7 scenes dataset. The large disparity between the two images causes the flow estimate to fail leading to incorrect predictions 20

List of Tables

4.1 Comparison of our method with baselines on the 7-scenes dataset. Lower scores are better for all categories. Best results in each category highlighted in bold. 15

List of Abbreviations

SO(3): Special Orthogonal Group (all orthogonal 3×3 matrices of determinant 1)

BA: Bundle Adjustment

CNN: Convolutional Neural Network

DUSt3R: Dense and Unconstrained Stereo 3D Reconstruction

GPU: Graphics Processing Unit

MLP: Multi Layer Perceptron

MVS: Multi-view Stereo

PnP: Perspective-n-Point

RANSAC: RANdom SAmple Consensus

RGB: Red Green Blue

SIFT: Scale Invariant Feature Transform

SLAM: Simultaneous Localization And Mapping

SfM: Structure from Motion

VGGSfM: Visual Geometry Grounded deep SfM

VGGT: Visual Geometry Grounded Transformer

1 | Introduction and Background

For long, geometry-based methods have been the dominant approach for image-based 3D reconstruction tasks, employing iterative optimization to improve results from initial estimates. Deep learning has been used to enhance parts of the pipeline that do not rely on geometry, such as keypoint matching, although it was not until VGGSfM [1] that deep learning was used for each stage of the pipeline, resulting in a fully differentiable SfM solver.

In the recent past, there have been attempts to replace the entire pipeline, and solve 3D tasks like SfM and MVS, with a single neural network. DUSt3r [2] showed encouraging results in this direction and its extension MASt3R [3] further improved results. Subsequently, there were many attempts to build on these models—using camera and scene priors to improve results [4], extending to dynamic scenes [5, 6, 7, 8], and improving performance for larger collections of images through multiview generalization [9, 10].

Recently, VGGT [11] showed that it was possible for a single model to solve multiple 3D tasks while matching or even beating previous state-of-the-art methods that depend on slow and computationally expensive bundle adjustment. Surprisingly, it achieved this without strong inductive biases, instead leveraging large-scale data and compute to build robust priors. However, the data and compute requirements of such approaches are quite expensive and can be a barrier to many people.

To solve these issues, we propose a novel approach to training for 3D reconstruction tasks. By training our model on depth estimates of keyframes and optical flow estimates for the other frames, we enable the learning of data priors that do not depend on RGB and thus generalize better to unseen data. Further, we introduce a new method to directly predict camera poses in a single forward pass, eliminating the need for optimization-based post-processing such as PnP with RANSAC [12]. We extend the DUSt3R model with an

additional prediction head that estimates a bundle of rays passing through each image pixel. The camera pose is then recovered from this ray bundle. Choosing the DUSt3R model also allows us to maintain low compute requirements as inference is still done pairwise which lifts GPU resource constraints, and the user can trade-off between speed and compute requirements by adjusting batch size.

We train our model in a fully-supervised manner on BlendedMVS [13], a relatively small public dataset of synthetic outdoor scenes. This dataset was also used by both DUSt3R and VGGT in their training, but it accounted for $\sim$1% of their total training data. This achieves the goal of adopting a fully data-driven training strategy, without any geometric constraints, while significantly reducing the amount of data required to achieve similar results.

To summarize, our contributions are as follows:

1. We introduce a novel method to directly predict relative camera poses in 3D reconstruction tasks.
2. We present a new training strategy to reduce the data and hardware costs associated with training feedforward models for 3D tasks. Despite being trained on only a small fraction of the data as other methods, we achieve comparable performance on unseen data, establishing that our approach generalizes to other settings by training only on synthetic outdoor scenes and testing on a collection of real indoor scenes.
3. We demonstrate that, for the same number of images, our model is faster end-to-end as we directly estimate camera pose parameters without need for further optimization.

2 | Related Work

2.1 Structure From Motion

Sparse 3D reconstruction, combined with simultaneous camera pose estimation, from image sets has been a longstanding challenge in computer vision. Most traditional pipelines [14, 15, 16] typically consist of multiple stages, including keypoint detection and matching, triangulation, and joint bundle adjustment to optimize 3D coordinates and camera parameters. COLMAP [17] is the most prominent of these pipelines. With the advent of deep learning, there were many attempts to improve individual stages of the pipeline such as feature detection [18, 19, 20, 21] and matching [22, 23, 24, 25]. Recently, methods [26, 27, 28, 29] have attempted end-to-end differentiable SfM, with VGGSfM [1] outperforming traditional algorithms in specific scenarios.

2.2 Multi-View Stereo

MVS takes as input images along with their camera parameters estimated with SfM and densely reconstructs the visible geometry of the scene. Traditional solutions to this problem can be grouped into three categories: completely "handcrafted" [30, 31, 32], optimization-based [33, 34, 35, 36], and learning-based [37, 38, 39, 40, 41]. However, all of these methods require explicit knowledge of camera parameters, and their performance is highly dependent on the quality of SfM estimates. Works like DUSt3R [2], MASt3R [3], and VGGT [11] have focused on performing MVS without any explicit knowledge of the camera parameters, instead relying on large data priors to directly predict scene geometry.

2.3 Ray-Based Camera Parametrization

Earlier works [42, 43] used ray-based representations for fish-eyed lenses where the standard pinhole model was not a good approximation. These model each pixel by its own ray, however, even better algorithms struggled to accurately calibrate these cameras due to the large number of parameters. Some methods [44, 45] have also attempted to use ray parametrizations to estimate camera poses from videos and sparse images.

Ray parametrizations are also popular in novel view synthesis techniques [46, 47, 48], where pixel-wise appearance conditioning is employed instead of explicitly rendering full images.

3 | Methodology

We begin by introducing the problem setup and some notation used in our method in section 3.1. Next, in section 3.2, we explain our method for recovering camera pose from the model's predictions, followed by section 3.3, where we outline how we build on the DUSt3R model to predict our new objective. Lastly, in section 3.4, we describe the training formulation and the dataset.

3.1 Problem Setup and Notation

Consider a short video V, taken from a camera with known intrinsics K, of a static scene that we wish to reconstruct in 3D. In order to achieve this goal, we can predict point maps, along with the corresponding position for the camera, at each video frame. Point maps relate each pixel u to a point p in 3D space, that is, $p = P(u) \in \mathbb{R}^3$.

In order to predict the corresponding positions for each camera, we represent the camera at each frame as a ray bundle B that contains rays b passing through each pixel in the frame. We represent these rays using Plücker coordinates.

The Plücker coordinates of a line ℓ are defined as $\langle d, m \rangle$, where d is its direction vector and m (its momentum vector) $= p \times d$ where p is any point on the line. The representation is independent of the choice of p as for any other point p' [49]

$$\begin{aligned} p' \times d &= [p + (p' - p)] \times d \\ &= p \times d + (p' - p) \times d \\ &= m + 0 \qquad\qquad \text{As } p' - p \text{ also lies on } d \\ &= m \end{aligned}$$

The Plücker coordinates of a line can thus be defined as a 6-tuple, therefore, our model

must also predict a "ray map" $b = B(u) \in \mathbb{R}^6$ for each pixel in the second image (camera 1 is chosen to be the world coordinate frame, therefore its pose is always identity). We elaborate on recovering camera pose from this representation in the next section.

The DUSt3R model Φ, which our model is adapted from, takes as input two images I_1 and I_2 and predicts point maps P_1 and P_2 in the coordinate frame of the first camera c_1, that is

$$\Phi(I_1, I_2) = (P_1^{c_1}, P_2^{c_1}). \tag{3.1}$$

On the other hand, our model takes as input a depth estimate for the first image and an optical flow estimate for the second image, and learns to predict point maps $P_1^{c_1}$ and $P_2^{c_1}$ as well as the ray map $B_2^{c_1}$ as described above, i.e.

$$\Phi'\big(\text{depth}(I_1), \text{flow}(I_2)\big) = (P_1^{c_1}, P_2^{c_1}, B_2^{c_1}). \tag{3.2}$$

Assume that V has n frames $[I_1, I_2, ..., I_n]$ with an equal number of camera positions $[c_1, c_2, ..., c_n]$ and point maps $[P_1, P_2, ..., P_n]$ corresponding to each frame. In order to use our model's pairwise predictions for video sequences, we select a keyframe I_k from the video as the keyframe and predict point maps and camera ray bundles for all other frames with respect to this key frame. Thus for a sequence of images, we perform the following:

$$\Phi'(\text{depth}(I_1), \text{flow}(I_2) = (P_k^{c_k}, P_i^{c_k}, B_i^{c_k}), \quad \forall i \neq k.$$

During inference, the model performs best when this keyframe is close to the middle of the frame, and we must transform all predictions back to the coordinate frame of the first camera, which conventionally serves as the world reference frame. For longer sequences of frames, we can select multiple keyframes (with some overlap) and chain pose

transformations to orient all predictions back to the world reference frame (first camera). Thus, the pairwise inference time of our model grows linearly with the number of images in the input.

3.2 Recovering Camera Pose

As described in the previous section, the model predicts a ray bundle B for the second image's camera in the coordinate frame of the first camera. This section describes how to recover the camera extrinsics matrix from this ray bundle representation.

The camera extrinsics matrix consists of two components—Rotation R, a 3×3 matrix, and translation t, a 3-vector—and is constructed as $\left[\begin{array}{c|c} R & t \\ 0 & 1 \end{array}\right]$ to form a 4×4 matrix. This extrinsics matrix can represent a transformation from world coordinates to camera coordinates, and its inverse, the camera pose matrix, represents a transformation from camera coordinates to world coordinates.

On the other hand, the intrinsics matrix is a 3×3 matrix which contains information about the focal lengths f_x and f_y (though usually, $f_x = f_y$) and principal point (x_0, y_0) in pixel coordinates (usually the centre of the image). The intrinsics matrix is formed as $\begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$.

In order to supervise training, we need ground truth ray bundles represented as Plücker coordinates. To obtain these, it is necessary to have the ground truth intrinsics K and camera pose R, t (camera-to-world transformation) for all images. Since all camera rays pass through the camera centre, we select it as the p for our moment computation for all rays. Thus, we can compute the plucker coordinates of ray passing through a pixel u as follows [50]:

$$d = R^T K^{-1} u, \quad m = (-R^T t) \times d.$$

Now, one can recover a camera's pose from its ray bundle representation as a separate

rotation and translation. In the coordinate frame of the first camera, the translation parameter for the second camera's pose will be equal to the location of its camera centre o in 3D space. This can be modelled as:

$$\begin{aligned} m &= o \times d \\ &= [d]_{\times}^{T} o \end{aligned}$$

where $[d]_{\times}$ is the skew-symmetric cross product matrix of d. This equation follows the form $Ax = b$ as m and d are known (predicted by our model) and o can be solved for by finding the best fit solution for $\langle d, m \rangle$pairs for all pixels. In our experiments, we found that solving this for each pixel can be quite expensive in the backward pass, therefore, we sample some of the pixels at random and only use those to compute the camera origin.

To recover the rotation matrix, we use the Kabsch Algorithm [51] to find the rotation matrix that best aligns the ground truth direction vectors of the first camera with those predicted by the model. This rotation corresponds to the rotation of the first camera to the second. Therefore, given ground truth direction vectors of the first camera d_1 and predicted direction vectors of the second camera d_2, the algorithm aims to solve the following equation:

$$\underset{R}{\operatorname{argmin}} ||d_2 - d_1 R||_F$$

such that $R \in \mathrm{SO}(3)$ (R is a valid rotation matrix).

3.3 Architecture

Our model is based on the DUSt3R architecture, with some changes to adapt to the different input modalities and training goals of our approach. The original DUSt3R architecture is inspired by CroCo [52] and thus benefits greatly from CroCo pretraining

[53], which also serves as the starting point for our training. The original model first encodes input images in a Siamese manner using a common image encoder, obtaining two token representations E_1 and E_2

$$E_1 = \text{Encoder}(I_1) \text{ and } E_2 = \text{Encoder}(I_2).$$

Next the tokens are fed into the decoder where they are sequentially input into decoder blocks consisting of alternating self-attention (each token of an image attends to all tokens of the same image) followed by cross-attention (each token of an image attends to all tokens of the other image) and finally input to an MLP. Thus, information is constantly shared between the two branches in the decoder.

$$D_1^i = \text{DecoderBlock}_1^i(D_1^{i-1}, D_2^{i-1}),$$
$$D_2^i = \text{DecoderBlock}_2^i(D_2^{i-1}, D_1^{i-1}),$$

for $i = 1, ..., b$ for a decoder with b blocks. The initial inputs are the tokens yielded by the encoder $D_1^0 = E_1$ and $D_2^0 = E_2$. DecoderBlock_u^v denotes the v^{th} block in the u^{th} branch ($u \in \{1, 2\}$). Finally the tokens from each branch are fed into a regression head that outputs a point map and an associated confidence map

$$P_1^{c_1}, C_1 = Head_1(D_1^0, ...D_1^b)$$
$$P_2^{c_1}, C_2 = Head_2(D_2^0, ...D_2^b).$$

The output point maps are predicted up to a one-parameter ambiguity (scale is unkown).

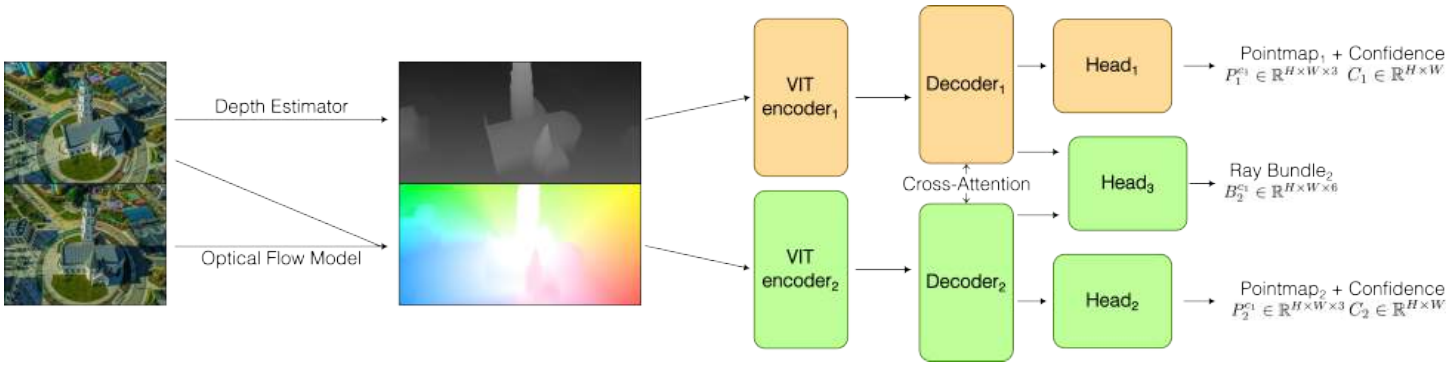


Figure 3.1: Our model's architecture. We use separate encoders to account for the different input modalities and the camera head takes tokens from both decoders as input to predict the relative pose of the second image.

It is important to note that this architecture has no explicit geometric inductive biases, and thus the predicted point maps may not correspond to any realistic camera model. The network uses large amounts of data to learn priors for depth, pose, intrinsics etc.

While the original DUSt3R architecture uses a common encoder for both input RGB images, we find that this is not enough when both images belong to different modalities, i.e. relative depth and optical flow. Thus, we train separate encoders for each modality.

$$E_1 = \text{Encoder}_1(I_1) \text{ and } E_2 = \text{Encoder}_2(I_2).$$

Furthermore, the original model only has 2 downstream heads to predict the point maps for each view, therefore we add another DPT head [54] in order to predict the Plücker Coordinates for the rays passing through each pixel. Since we predict the relative pose between the cameras, we provide tokens from both decoders to the camera head:

$$B_2^{c_1} = Head_3(D_1^0, D_2^0, ...D_1^b, D_2^b).$$

We use Depth Anything V2 [55] and DPFlow [56] to precompute the depth and flow estimates to avoid recomputing these at runtime. We also noticed that flow estimates in the forward direction, that is from the first frame to the second frame, gave significantly poorer results as compared to the reverse. This is because the reverse direction includes information about the second image including for those pixels that are not present in the first image.

3.4 Training

As mentioned in the introduction, we train only on the BlendedMVS dataset which is a static and synthetic collection of mostly outdoor scenes. We use the same preprocessing as DUSt3R to form images pairs for training, however, due to the nature of optical flow models, we cannot have pairs with extremely wide baselines, thus we filter out pairs with extremely low similarity scores, giving us roughly 100,000 image pairs for training.

To keep training costs minimal, we follow a similar sequential training regime as DUSt3R—training on smaller 224×224 images and then on larger 512×512 images. These steps use the same confidence-aware 3D regression loss as the original model which is a confidence-weighted 3D regression loss over all valid pixels.

We perform an additional training step in order to train the camera head. Here, we freeze the encoders, decoders, and other camera heads as these already generate enough information for dense 3D reconstruction and this is sufficient to learn camera poses as well. To supervise learning, we use the following losses:

1. **RRE and RTE**: from the predicted Plücker Coordinates, we first compute the camera pose as outlined above in section 3.2. Once we have the camera pose matrix, we compute the Relative Rotation Error (RRE) and Relative Translation Error (RTE) between the ground truth and predicted camera pose and try to minimize these through training. We use these as they are invariant to the scale ambiguity

of the coordinate frame. To compute the RRE, we must first convert the rotation matrices to quaternions and then compute the following:

$$\ell_{\text{RRE}}(q_{\text{gt}}, q_{\text{pred}}) = \cos^{-1}(2\langle q_{\text{gt}}, q_{\text{pred}}\rangle^2 - 1).$$

where $\langle q_{\text{gt}}, q_{\text{pred}}\rangle$is the dot product of predicted and ground truth quarternions.

The RTE computes the alignment between two given translation vectors as:

$$\ell_{\text{RTE}}(t_{\text{gt}}, t_{\text{pred}}) = \cos^{-1}(\frac{t_{\text{gt}} \cdot t_{\text{pred}}}{||t_{\text{gt}}||\ ||t_{\text{pred}}||}).$$

We compute both the RRE and RTE in degrees.

2. **MSE Loss**: In practice we noticed that the Plücker Coordinates predicted by the camera head would become very large after a few epochs as they tried to minimize RTE and RRE, leading to invalid (infinite) values in the pose recovery computation. To combat this, we add an MSE loss term to ensure that the predicted values don't stray too far from the target values. This loss doesn't greatly affect gradient calculations in the early stages of training as the RRE and RTE losses are orders of magnitude larger. A similar reconstruction loss is used in [45] when estimating camera ray bundles in Plücker Coordinates through diffusion.

$$\ell_{\text{MSE}}(B_{\text{gt}}, B_{\text{pred}}) = \frac{\sum\limits_{u,v}(B_{\text{gt}}^{(u,v)} - B_{\text{pred}}^{(u,v)})^2}{u \times v}.$$

Since RTE and RRE are both in degrees, and we want the MSE loss, which in our experiments was significantly smaller than RTE and RRE, to act more as a regulariser (a norm penalty), we choose not to scale the loss terms relative to each other and instead

add them directly to obtain our final loss:

$$\mathcal{L} = \ell_{\text{RRE}} + \ell_{\text{RTE}} + \ell_{\text{MSE}}.$$

4 | Results and Analysis

4.1 Evaluation Method and Metrics

To measure our model's accuracy in predicting camera poses, we use the Relative Rotational Error and Relative Translational Error metrics as described in the previous chapter. We also calculate the mean Euclidean distance between the predicted and ground truth 3D positions for each pixel to measure MVS performance.

$$\text{Mean Euclidean Distance}(P_{\text{gt}}^{c_1}, P_{\text{pred}}^{c_1}) = \frac{\sum_{i,j} \sqrt{\sum_{x,y,z} (P_{\text{gt}}^{c_1} - P_{\text{pred}}^{c_1})^2}}{i \times j}$$

The predicted and ground truth 3D points are normalized and the predicted point cloud is aligned to the ground truth using the Umeyama algorithm [57].

We evaluate our model on the 7-scenes dataset [58] which is a collection of 46 sequences of seven real indoor scenes. The dataset also provides ground truth depth maps and we obtain the RGB and depth camera calibration parameters from [59] and use these to reproject the depth maps with kapture [60] to obtain ground truth 3D points.

To test the models, we randomly choose a starting frame from each of the sequences and then select every 4^{th} frame in the sequence till we have a sample of 16 frames. This approach is chosen rather than a completely random sample to prevent degenerate optical flow estimates. We keep the seed fixed to ensure that the same frames are chosen for all three models.

We choose DUSt3R (with and without bundle adjustment) and VGGT as our baselines as these are examples of approaches that focus on leveraging data and provide state-of-the-art results in multiple 3D tasks. We infer all images against the middle frame

for reference for both our method and DUSt3r and then transform results back to the coordinate frame of the first camera to evaluate results. All three models are tested with a batch size of 16 and on a single Nvidia RTX 3090 GPU with 24 GB VRAM.

4.2 Results

7-Scenes Dataset	RRE (deg.) ↓	RTE (deg.) ↓	Dist. ↓	Time
DUSt3R (wo. BA)	3.02	27.67	0.47	∼2.89s
DUSt3R (with BA)	2.14	26.06	0.46	∼11.67s
VGGT	**1.54**	**10.20**	**0.44**	∼1.36s
Ours	7.71	34.34	0.54	**∼0.86s**

Table 4.1: Comparison of our method with baselines on the 7-scenes dataset. Lower scores are better for all categories. Best results in each category highlighted in bold.

We show the results of our evaluation on the 7-scenes dataset in Table 4.1. While we are unable to surpass the baselines on the chosen metrics, we are competitive across the board. VGGT is the strongest when it comes to translation because it does not suffer from the same accumulation of errors as pairwise methods like ours and DUSt3R.

It is also important to note that our model was never specifically trained on real indoor scenes, our competitive result in the distance metric shows that our training method allows our model to generalise well to data from other scenarios. Visual results of the same can be seen in Figure 4.3.

Furthermore, we also show that our camera pose prediction method approaches the performance of other pairwise models while taking roughly a third of the time by avoiding costly optimization.

4.3 Limitations

One of the major limitations of our approach is its dependence on the initial depth and optical flow estimates. If these estimators miss any detail, such as a piece on a chessboard, our model has no way of inferring its presence, leading to an incorrect prediction. Moreover, the optical flow assumption of brightness consistency [61] enforces a narrow baseline condition on the input images, and thus, our model fails for sparse sets of views.

Furthermore, we do not apply any constraints on the predicted ray bundles. For a standard camera model with the principal point at the center of the image, we can enforce direction symmetricity constraints on the outputs, which would lead to better Plücker Coordinate approximations and ultimately better pose predictions. Another extension of this idea would be to predict only one ray per image token. This would lead to faster convergence as all rays could be used in the camera origin calculation and would also reduce the number of parameters needed by the model.

Additionally, as with any other pairwise approach, there is the problem of long input sequences. While our model is able to predict point maps and relative camera poses for a pair of images with only minute errors, these inaccuracies accumulate in longer sequences and we see that predictions of frames further down the sequence become increasingly inaccurate.

Lastly, our training currently only consists of 1 dataset. While our approach is not as data hungry as other methods, our model would benefit from stronger priors with scenes from more diverse scenarios.

4.4 Visual Results

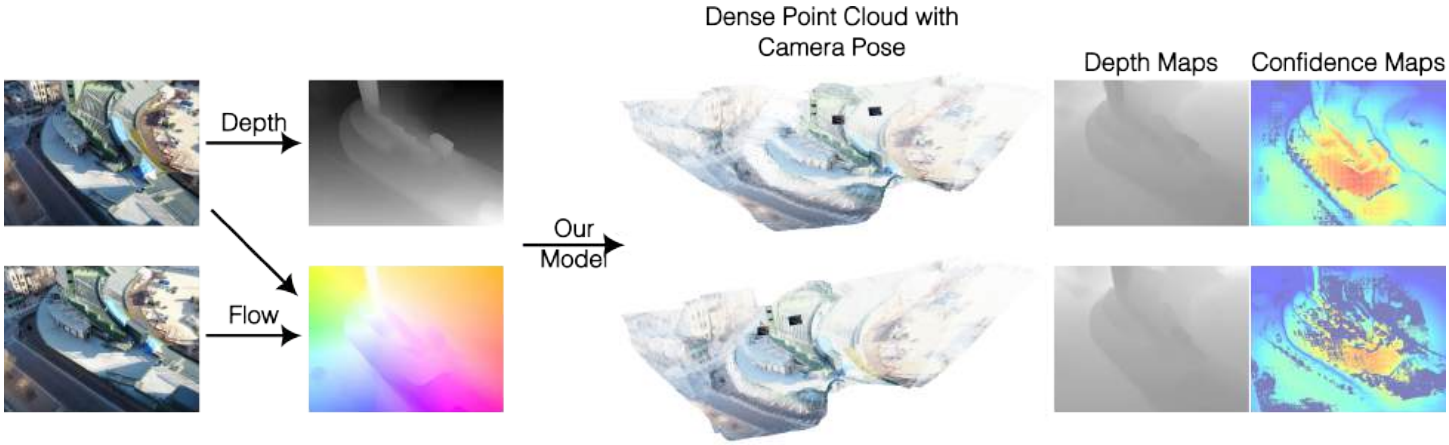


Figure 4.1: Inference results of our model on a pair of images from the BlendedMVS dataset. Here, we show the predicted point cloud without any meshing.

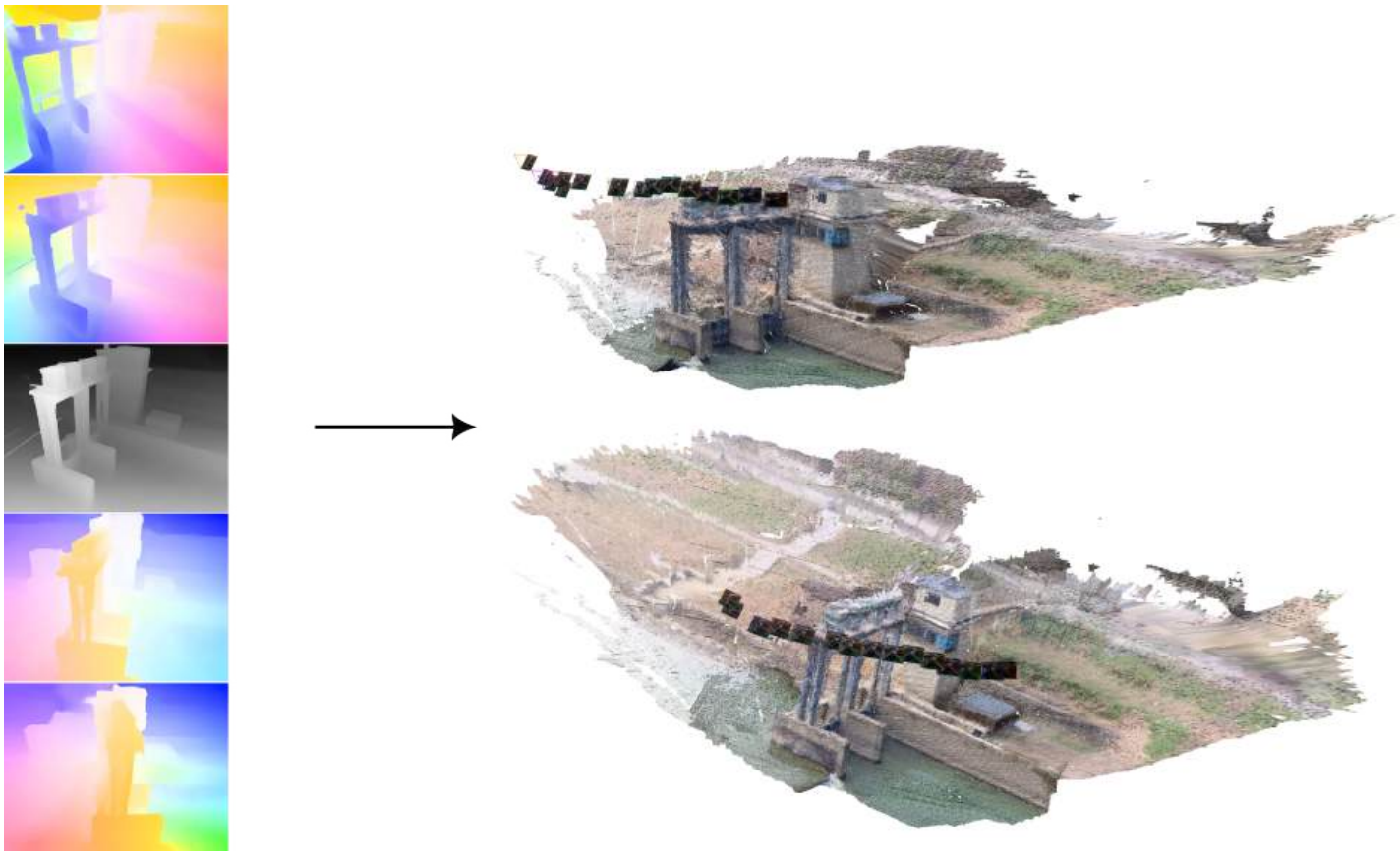


Figure 4.2: Inference results of our model on a set of 16 images (5 shown) from the BlendedMVS dataset with meshing. On the left we visualise an example of what the output might see after processing with keyframe depth and optical flow estimates with the middle frame chosen as the keyframe.

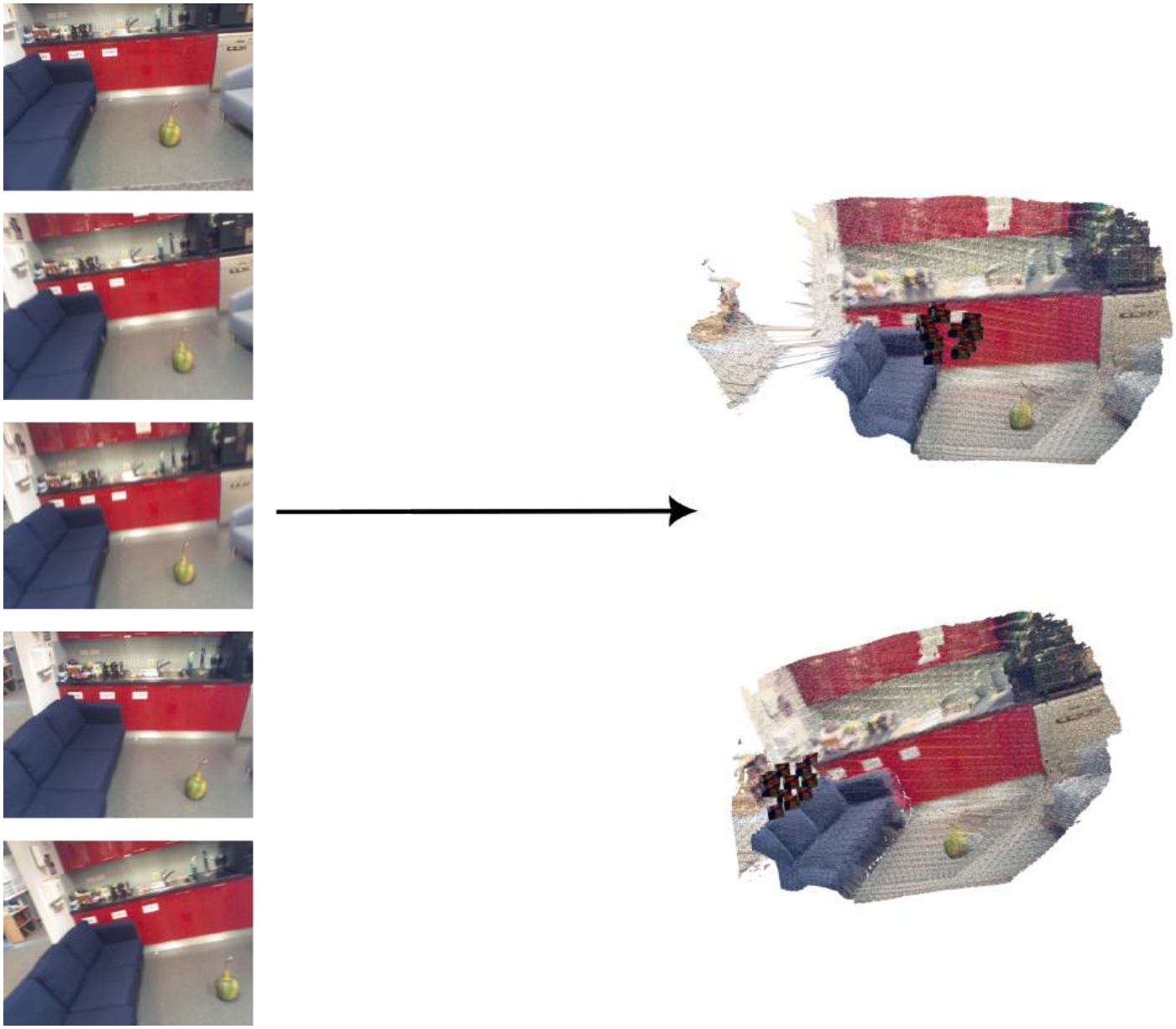


Figure 4.3: Inference results of our model on a set of 16 images (5 shown) from the 7 scenes dataset with meshing. On the left we visualise the actual RGB inputs before the depth and optical flow estimates.

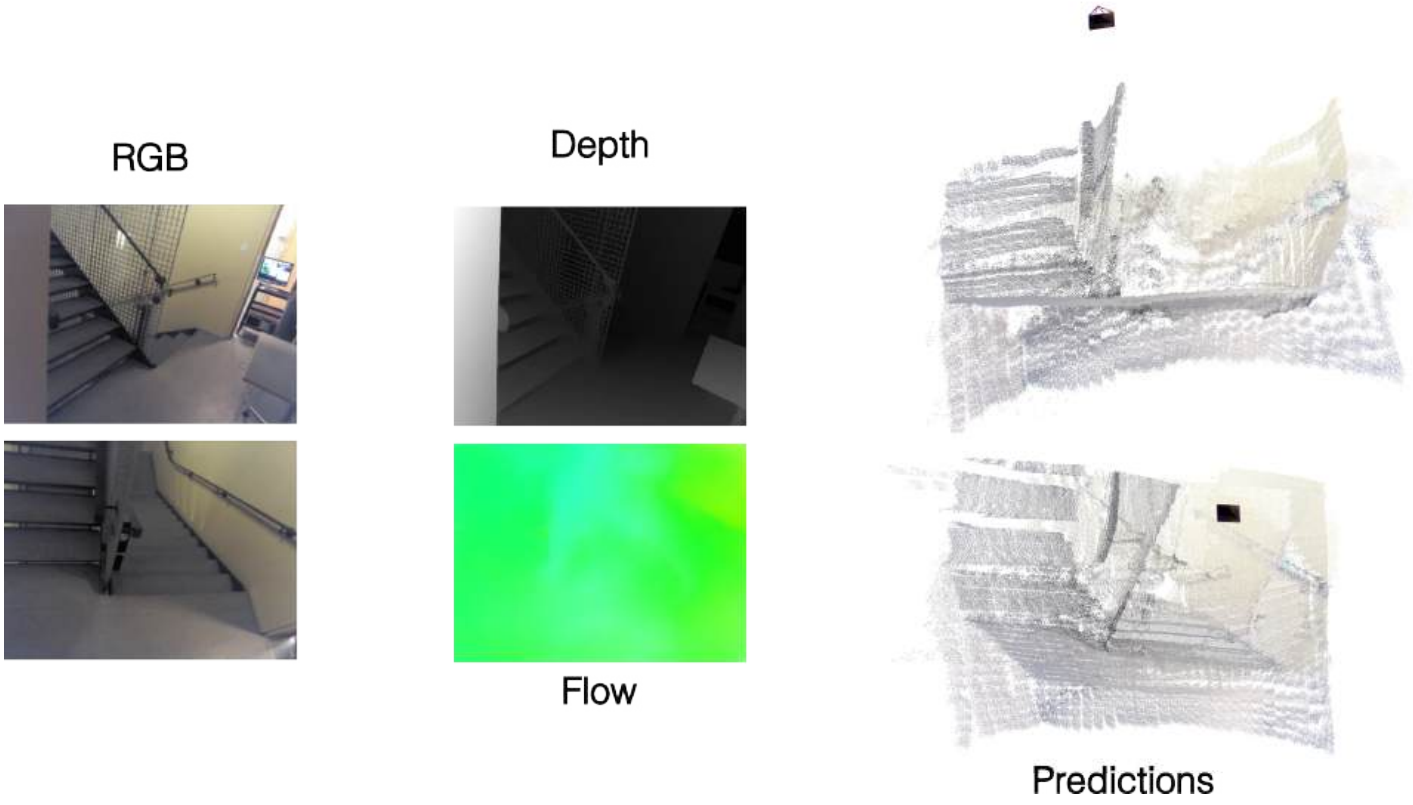


Figure 4.4: A failure case on a pair of images from the 7 scenes dataset. The large disparity between the two images causes the flow estimate to fail leading to incorrect predictions

5 | Conclusion

We present a novel training strategy for data-driven 3D models alongside a new method for predicting camera poses. Our model achieves results competitive with current state-of-the-art approaches in camera pose estimation and multi-view stereo, while requiring only a fraction of the training data. We also show that our training approach produces a model that generalizes effectively to entirely different data domains. Furthermore, our method removes the need for post-processing optimization to recover camera poses, thereby reducing the time required for inference.

In the future, to reduce the errors from inaccurate depth and flow estimates, we can also provide the model with the original RGB images so that it may see information that the estimates missed. We would also like to extend our training strategy to a multi-view setting so that we may regress point maps and poses for multiple views at once, reducing the accumulation of errors in long input sequences.

References

[1] J. Wang, N. Karaev, C. Rupprecht, and D. Novotny, "Vggsfm: Visual geometry grounded deep structure from motion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21686–21697, 2024.

[2] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *CVPR*, 2024.

[3] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3d with mast3r," 2024.

[4] W. Jang, P. Weinzaepfel, V. Leroy, L. Agapito, and J. Revaud, "Pow3r: Empowering unconstrained 3d reconstruction with camera and scene priors," 2025.

[5] E. Sucar, Z. Lai, E. Insafutdinov, and A. Vedaldi, "Dynamic point maps: A versatile representation for dynamic 3d reconstruction," 2025.

[6] J. Zhang, C. Herrmann, J. Hur, V. Jampani, T. Darrell, F. Cole, D. Sun, and M.-H. Yang, "Monst3r: A simple approach for estimating geometry in the presence of motion," *arXiv preprint arxiv:2410.03825*, 2024.

[7] H. Feng, J. Zhang, Q. Wang, Y. Ye, P. Yu, M. J. Black, T. Darrell, and A. Kanazawa, "St4rtrack: Simultaneous 4d reconstruction and tracking in the world," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[8] X. Fei, W. Zheng, Y. Duan, W. Zhan, M. Tomizuka, K. Keutzer, and J. Lu, "Driv3r: Learning dense 4d reconstruction for autonomous driving," *arXiv preprint arXiv:2412.06777*, 2024.

[9] Y. Cabon, L. Stoffl, L. Antsfeld, G. Csurka, B. Chidlovskii, J. Revaud, and V. Leroy, "Must3r: Multi-view network for stereo 3d reconstruction," in *CVPR*, 2025.

[10] J. Yang, A. Sax, K. J. Liang, M. Henaff, H. Tang, A. Cao, J. Chai, F. Meier, and M. Feiszli, "Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025.

[11] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[13] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," *Commun. ACM*, vol. 54, p. 105â112, Oct. 2011.

[15] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building rome on a cloudless day," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 368–381, Springer Berlin Heidelberg, 2010.

[16] S. Liu, Y. Gao, T. Zhang, R. Pautrat, J. L. Schönberger, V. Larsson, and M. Pollefeys, "Robust incremental structure-from-motion withnbsp;hybrid features," in *Computer Vision â ECCV 2024: 18th European Conference, Milan, Italy, September 29âOctober 4, 2024, Proceedings, Part XXXVI*, (Berlin, Heidelberg), p. 249â269, Springer-Verlag, 2024.

[17] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[19] M. Tyszkiewicz, P. Fua, and E. Trulls, "Disk: Learning local features with policy gradient," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] S. Suri, M. Walmer, K. Gupta, and A. Shrivastava, "Lift: A surprisingly simple lightweight feature transform for dense vit descriptors," 2024.

[21] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," 2018.

[22] H. Chen, Z. Luo, J. Zhang, L. Zhou, X. Bai, Z. Hu, C.-L. Tai, and L. Quan, "Learning to match features with seeded graph matching network," 2021.

[23] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *CVPR*, 2020.

[24] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "LightGlue: Local Feature Matching at Light Speed," in *ICCV*, 2023.

[25] Y. Shi, J.-X. Cai, Y. Shavit, T.-J. Mu, W. Feng, and K. Zhang, "Clustergnn: Cluster-based coarse-to-fine graph neural network for efficient feature matching," 2023.

[26] C. Smith, D. Charatan, A. Tewari, and V. Sitzmann, "Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent," 2024.

[27] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in neural information processing systems*, 2021.

[28] Z. Teed and J. Deng, "Deepv2d: Video to depth with differentiable structure from motion," 2020.

[29] X. Wei, Y. Zhang, Z. Li, Y. Fu, and X. Xue, "Deepsfm: Structure from motion via deep bundle adjustment," in *ECCV*, 2020.

[30] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[31] Y. Furukawa and C. HernÃ¡ndez, *Multi-View Stereo: A Tutorial*. Now Publishers, 2015.

[32] S. Galliani, K. Lasinger, and K. Schindler, "Massively parallel multiview stereopsis by surface normal diffusion," June 2015.

[33] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5590–5599, 2021.

[34] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[35] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[36] Q. Fu, Q. Xu, Y.-S. Ong, and W. Tao, "Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction," 2022.

[37] Z. Zhang, R. Peng, Y. Hu, and R. Wang, "Geomvsnet: Learning multi-view stereo with geometry perception," in *Proceedings of the IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition, pp. 21508–21518, 2023.

[38] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan, "Cascade cost volume for high-resolution multi-view stereo and stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2495–2504, 2020.

[39] Z. Ma, Z. Teed, and J. Deng, "Multiview stereo with cascaded epipolar raft," in *Proceedings of the European conference on computer vision (ECCV)*, 2022.

[40] R. Peng, R. Wang, Z. Wang, Y. Lai, and R. Wang, "Rethinking depth estimation for multi-view stereo: A unified representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[41] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," *European Conference on Computer Vision (ECCV)*, 2018.

[42] J. Kannala and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," 2006.

[43] M. Grossberg and S. Nayar, "A general imaging model and a method for finding its parameters," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 108–115 vol.2, 2001.

[44] I. Vasiljevic, V. Guizilini, R. Ambrus, S. Pillai, W. Burgard, G. Shakhnarovich, and A. Gaidon, "Neural ray surfaces for self-supervised learning of depth and ego-motion," 2020.

[45] J. Y. Zhang, A. Lin, M. Kumar, T.-H. Yang, D. Ramanan, and S. Tulsiani, "Cameras as rays: Pose estimation via ray diffusion," in *International Conference on Learning Representations (ICLR)*, 2024.

[46] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand, "Light field networks: Neural scene representations with single-evaluation rendering," in *Proc. NeurIPS*, 2021.

[47] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[48] D. Watson, W. Chan, R. Martin-Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi, "Novel view synthesis with diffusion models," 2022.

[49] Y.-B. Jia, "Plücker coordinates for lines in the space," 2024.

[50] A. K. Subramaniam, "Camera parameterization using plücker coordinates," 2025.

[51] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, pp. 922–923, Sep 1976.

[52] Weinzaepfel, Philippe and Leroy, Vincent and Lucas, Thomas and Brégier, Romain and Cabon, Yohann and Arora, Vaibhav and Antsfeld, Leonid and Chidlovskii, Boris and Csurka, Gabriela and Revaud Jérôme, "CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion," in *NeurIPS*, 2022.

[53] P. Weinzaepfel, T. Lucas, V. Leroy, Y. Cabon, V. Arora, R. Brégier, G. Csurka, L. Antsfeld, B. Chidlovskii, and J. Revaud, "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow," in *ICCV*, 2023.

[54] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *ArXiv preprint*, 2021.

[55] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," *arXiv:2406.09414*, 2024.

[56] H. Morimitsu, X. Zhu, R. M. C. Jr., X. Ji, and X.-C. Yin, "Dpflow: Adaptive optical flow estimation with a dual-pyramid framework," 2025.

[57] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[58] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time rgb-d camera relocalization," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, October 2013.

[59] M. Zins and K. Feingold, "7-scenes color/depth calibration." https://github.com/zinsmatt/7-Scenes-Calibration, 2024.

[60] M. Humenberger, Y. Cabon, N. Guerin, J. Morat, V. Leroy, J. Revaud, P. Rerole, N. Pion, C. de Souza, and G. Csurka, "Robust image retrieval-based visual localization using kapture," 2022.

[61] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: A survey," *Computer Vision and Image Understanding*, vol. 134, 02 2015.