DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

AnyCam-3D: Extending AnyCam for Dense 3D Geometry Reconstruction in Dynamic, Uncalibrated Videos

Nil Biescas

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

AnyCam-3D: Extending AnyCam for Dense 3D Geometry Reconstruction in Dynamic, Uncalibrated Videos

Erweiterung von AnyCam zur dichten 3D-Geometrierekonstruktion in dynamischen, nicht kalibrierten Videos

Author:	Nil Biescas
Supervisor:	Daniel Cremers
Advisor:	Felix Wimbauer and Dominik Muhle
Submission Date:	02/09/2025

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Felix Wimbauer, for their guidance, support, and valuable feedback throughout this thesis. I am also thankful to Dominik Muhle for their insightful discussions and encouragement during this work.

Abstract

Estimating dense 3D geometry from casual videos is a challenging task in computer vision, particularly in the presence of dynamic objects and uncalibrated cameras. Traditional SfM and SLAM pipelines rely on bundle adjustment and perform well in static scenes but often fail under these conditions, while recent deep learning methods like DUSt3R have reframed point map estimation in a data-driven way but remain limited to static scenes and often require supervision or calibration. This thesis introduces AnyCam-3D, an extension of the self-supervised AnyCam framework, which was designed for robust camera motion and intrinsics estimation in dynamic videos. We augment AnyCam with a Dense Prediction Transformer (DPT) head to predict dense 3D point maps and propose three complementary losses for training: confidence-aware regression, temporal consistency, and depth supervision. Experiments on the TUM-RGBD dynamic sequences demonstrate that AnyCam-3D can predict temporally consistent, metric-scale 3D geometry while maintaining strong pose estimation performance. This work highlights the feasibility of self-supervised, feed-forward 3D reconstruction in dynamic settings and provides a foundation for future methods leveraging uncalibrated, in-the-wild videos for large-scale scene geometry learning.

Zusammenfassung

Die Schätzung dichter 3D-Geometrie aus ungezwungenen Videodaten stellt eine anspruchsvolle Aufgabe in der Computer Vision dar, insbesondere bei Szenen mit dynamischen Objekten und nicht kalibrierten Kameras. Traditionelle SfM- und SLAM-Pipelines basieren auf Bundle Adjustment und liefern in statischen Szenen gute Ergebnisse, versagen jedoch häufig unter diesen Bedingungen. Neuere Deep-Learning-Ansätze wie DUSt3R haben die Schätzung von Punktkarten datengetrieben neu formuliert, sind jedoch weiterhin auf statische Szenen beschränkt und erfordern oft Überwachung oder Kalibrierung. Diese Arbeit stellt AnyCam-3D vor, eine Erweiterung des selbstüberwachten AnyCam-Frameworks, das für eine robuste Schätzung von Kamerabewegungen und Kameraintrinsiken in dynamischen Videos entwickelt wurde. Wir erweitern AnyCam um einen Dense Prediction Transformer (DPT)-Kopf, der dichte 3D-Punktkarten vorhersagt, und schlagen drei komplementäre Verlustfunktionen für das Training vor: eine konfidenzbewusste Regression, eine zeitliche Konsistenz und eine Tiefenüberwachung. Experimente auf den dynamischen Sequenzen des TUM-RGBD-Datensatzes zeigen, dass AnyCam-3D in der Lage ist, zeitlich konsistente, metrisch skalierte 3D-Geometrien vorherzusagen und gleichzeitig eine starke Leistung bei der Schätzung der Kameraposen zu bewahren. Diese Arbeit unterstreicht die Machbarkeit einer selbstüberwachten, vorwärtsgerichteten 3D-Rekonstruktion in dynamischen Szenarien und bildet eine Grundlage für zukünftige Methoden, die unkalibrierte Videos aus realen Umgebungen zur großskaligen Geometrieerfassung von Szenen nutzen.

Contents

1 Introduction

Estimating dense 3D point maps that capture the geometry of a scene from dynamic videos is a long-standing challenge in computer vision. A robust solution to this problem would provide accurate scene structure for robotics, enabling improved navigation, planning, and perception in complex real-world environments. In addition, being able to extract large-scale 3D data from casual, uncalibrated video sources (e.g., online videos) could help overcome one of the main bottlenecks in robotics research: the scarcity of annotated 3D data for training agents that operate outside simulation.

Traditional pipelines such as Structure-from-Motion (SfM) and Simultaneous Localization and Mapping (SLAM) have been highly successful for static scenes, but they often struggle in dynamic settings where independently moving objects and scene changes violate their geometric assumptions. Recent research has shifted toward deep learning methods that predict dense 3D geometry directly from images. A key milestone is DUSt3R [**dust3r**], which introduced a unified framework to predict dense 3D point maps from image pairs, followed by alignment-based recovery of camera intrinsics and poses. Although DUSt3R is designed for static scenes, its simplicity, accuracy, and availability have made it a popular foundation for subsequent research, and many recent approaches build upon its formulation. MonST3R [**monst3r**] extends this framework to videos, introducing temporal consistency to handle sequential data.

In parallel, optimization-based pipelines such as MegaSAM [**megasam**] integrate SLAM-inspired differentiable optimization with learned depth, pose, and motion predictions. However, these methods are generally trained in a supervised manner and assume calibrated intrinsics, limiting their applicability to in-the-wild videos where such calibration data is rarely available.

In this work, we introduce **AnyCam-3D**, an extension of AnyCam [**anycam**], which was developed for robust self-supervised estimation of camera intrinsics and motion in dynamic, uncalibrated video sequences. We augment AnyCam with a Dense Prediction Transformer (DPT) head to predict dense 3D point maps directly, leveraging its temporally consistent camera predictions. To ensure robust geometry, we design three complementary loss functions: a confidence-aware regression loss, a temporal consistency loss, and a depth supervision loss. AnyCam-3D demonstrates that accurate, metric-scale 3D structure can be learned from casual dynamic videos without ground-truth intrinsics or poses, bridging the gap between learning-based reconstruction and optimization-heavy SLAM systems.

1.1 Research Objectives

The core objective of this research is to extend the AnyCam model to perform dense 3D point estimation. This study specifically aims to: (1) evaluate whether incorporating 3D geometry

reconstruction improves the base model's performance on its primary tasks of camera pose estimation and intrinsics recovery in dynamic scenes, (2) identify effective strategies for training models to predict accurate 3D point maps in dynamic, uncalibrated video settings, and (3) achieve temporally consistent 3D point maps across frames to ensure coherent scene geometry over time.

2 Back Ground

2.1 Structure from Motion

Structure from Motion is a classical problem in computer Vision [1]. Given a set of N images of a scene taken with a camera, SfM (Structure from Motion) consist on reconstructing the intrinsics matrix of the camera as well as to reconstruct the scene using sparse 3D point maps to have a 3D model of the scene.

Traditionally SfM Structure-from-Motion (SfM) estimates both camera parameters and 3D structure from a collection of images. The relationship between a 3D point in the scene and its image projection is described by the *camera projection matrix*:

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}], \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \lambda = Z_c,$$

with

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad [\mathbf{R} \mid \mathbf{t}] = \begin{bmatrix} & & t_x \\ & \mathbf{R} & t_y \\ & & t_z \end{bmatrix}.$$

Here, $\mathbf{K}$ is the intrinsic calibration matrix, $\mathbf{R}$ is the 3×3 rotation matrix, $\mathbf{t} = (t_x, t_y, t_z)^\top$ is the translation vector, and Z_c is the depth of the point in the camera frame.

This formulation underpins epipolar geometry, as correspondences across views can constrain both camera poses and scene structure. This mathematical constraint is used along others to solve the Sfm problem.

A widely used tool for SfM is `COLMAP` [2]. It combines several components: image matching with feature descriptors, outlier rejection via RANSAC to ensure geometric consistency, and the construction of a *scene graph* with images as nodes and verified matches as edges. This graph encodes which views observe overlapping parts of the scene, serving as the foundation for subsequent pose estimation and 3D reconstruction.

The next step is choosing an initial image pair, which is critical: a poor initialization can make the entire reconstruction fail. From there, the reconstruction proceeds through an iterative process called *incremental reconstruction* as showned in Figure 2.1. In each iteration, a new image is registered to the model by triangulating new 3D points with already registered images. Bundle adjustment then refines camera intrinsics, extrinsics, and 3D point locations jointly, while outlier filtering removes unreliable correspondences. These steps—image registration, triangulation, bundle adjustment, and filtering—are repeated until the entire scene is reconstructed. The result is a sparse 3D point cloud together with calibrated cameras.

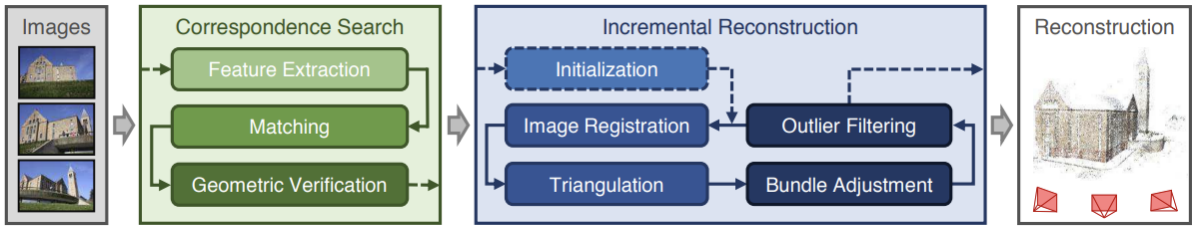

Figure 2.1: Incremental Structure-from-Motion (SfM) pipeline. Given a set of N images of a scene, `COLMAP` first performs correspondence search by extracting and matching feature descriptors across images, followed by geometric verification. The system then incrementally reconstructs the scene by registering images, triangulating 3D points, and refining estimates through bundle adjustment and outlier filtering, leveraging epipolar geometry constraints at each step. Figure from [2].

Revisiting SfM More recent work has explored incorporating learning-based techniques into the COLMAP pipeline, motivated by its main limitation: the lack of end-to-end differentiability. Approaches such as `FlowMap` [3] rely on training a regressor network at test time. FlowMap directly minimizes the difference between the optical flow induced by a moving camera in a static 3D scene and precomputed correspondences in the form of point tracks and off-the-shelf optical flow.

Other methods follow a two-stage design: first, a neural network pretrained on large quantities of 3D data predicts dense point maps; second, these point maps are optimized with bundle adjustment to recover camera intrinsics and extrinsics. An example of this paradigm is `DUSt3r` [4], a foundation model for 3D vision. DUSt3r decomposes SfM into two steps: (i) local reconstruction of every image pair into point maps, and (ii) global alignment of all point maps into a consistent world coordinate frame.

2.2 SLAM

Structure from Motion and SLAM are understood as fundamental in Computer Vision and in industrial applications ranging from robotics to Virtual Reality. Simultaneous Localization and Mapping is one of the fundamental challenges in robotics navigation, dealing with the necessity of building a map of the environment while simultaneously determining the location of the robot within this map.

Traditionally SLAM Early SLAM methods were based on probabilistic formulations, modeling the problem as a joint estimation of robot trajectory and map. Techniques such as the Extended Kalman Filter (EKF-SLAM) and Particle Filters (FastSLAM) [5] dominated the classical literature, relying on incremental sensor updates to estimate both the robot pose and the position of landmarks in the environment. These methods were effective in small-scale, low-dimensional settings but struggled with scalability, drift, and nonlinearities in real-world scenarios.

The introduction of graph-based SLAM [6] increased the robustness of SLAM methods. Here, the problem is formulated as a factor graph [7] where nodes represent robot poses or

landmarks, and edges encode relative constraints obtained from odometry or sensor observations. This graph formulation allows for global optimization using nonlinear least squares, reducing accumulated drift and providing more consistent maps. Outlier rejection and loop closure detection play a critical role, as they allow the system to correct long-term drift by recognizing previously visited places.

Revisiting SLAM Modern SLAM systems have incorporated advanced optimization techniques, robust perception modules, and learning-based components. Visual SLAM pipelines such as ORB-SLAM [8] have become standard, using feature detection, tracking, and bundle adjustment to achieve high accuracy from camera data alone. Dense SLAM systems, such as KinectFusion [9], expanded the focus from sparse landmarks to dense surface reconstruction, enabling detailed 3D mapping for augmented and virtual reality.

Recently, deep learning has been increasingly integrated into SLAM pipelines [10]. Learning-based approaches replace or augment traditional components such as feature detection, data association, and depth estimation. Neural implicit representations and differentiable mapping techniques push SLAM towards end-to-end differentiable pipelines, making it possible to jointly optimize perception, mapping, and localization. At the same time, hybrid approaches combine classical geometry with learned priors, aiming to retain robustness while leveraging the adaptability of deep models.

3 Related Work

3.1 3D Point Maps as foundation for 3D Tasks

A robot, understood as an autonomous agent operating in three-dimensional space, is composed of multiple subsystems [11, 12]. Among these, navigation control plays a central role, as it provides the information required for decision-making based on the robot's position and perception of the environment.

In computer vision, a fundamental premise is that understanding the 3D world entails the ability to reconstruct it [13]. A reliable 3D reconstruction enables downstream tasks such as motion planning—where optimal paths can be computed to avoid obstacles—as well as object detection and localization, for instance in search-and-rescue scenarios. More broadly, reconstruction provides access to the geometry of objects in the scene, their visual appearance, spatial relationships, and interactions with illumination. While modern rendering pipelines can produce realistic visual effects when geometry is available, obtaining accurate geometry remains a central challenge. Classical approaches rely on multi-stage pipelines involving camera calibration, depth estimation, pixel correspondence matching, camera pose estimation, and dense 3D reconstruction [2].

Recovering the intrinsic and extrinsic parameters of a camera typically requires formulating nonlinear optimization problems [14, 1]. These problems are often sensitive to initialization and assumptions, and the conditions for convergence are not always satisfied. As a result, obtaining accurate solutions that generalize across diverse settings often demands extensive refinement and careful calibration procedures.

In natural language processing (NLP), certain tasks—such as machine translation or question answering—have historically played a role similar to that of camera pose estimation in computer vision, serving as classical benchmarks. Traditionally, each of these tasks required specialized systems tailored to their specific problem formulation. However, it is reasonable to assume that solving such tasks, whether in NLP or computer vision, relies on underlying knowledge that is not task-specific but rather shared across domains [15]. This suggests that data inherently contains structural regularities that a single model could exploit to address multiple tasks simultaneously.

The T5 model explored this hypothesis through transfer learning with a unified text-to-text framework [16]. Their approach employed an encoder–decoder Transformer architecture to perform a wide range of NLP tasks by casting them into a common text-to-text format. Under this formulation, the same model was able to handle question answering, machine translation, and text summarization, among others, within a unified paradigm.

Following the success of unified models in NLP, and motivated by the idea of addressing multiple tasks within a single framework, DUST3R [4] extended this paradigm to 3D computer vision by introducing a shared output representation capable of solving a variety of geometric tasks with one model.

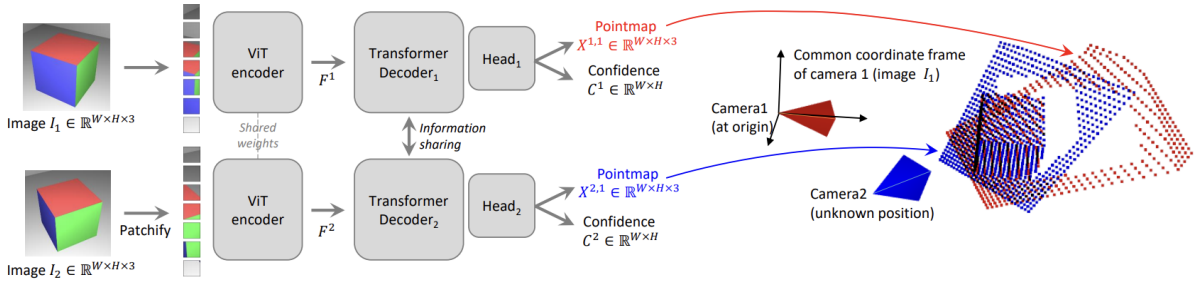

Figure 3.1: Two input views of a scene, I^1 and I^2, are processed by a shared Vision Transformer (ViT) encoder in a Siamese setup, producing token features F^1 and F^2. These features are fed into two transformer decoders that communicate through cross-attention. Each decoder outputs a pointmap and its corresponding confidence map, with both pointmaps expressed in the coordinate frame of the first image I^1. Figure from [4].

Dust3r proposed to make Geometric 3D Vision easy. It tackles the problem of Multi-view stereo reconstruction (MVS) in the wild, where the task is to reconstruct the 3D object capture from different points of view. Traditionally one would need to first estimate the camera intrinsic and extrinsic parameters. These are usually tedious and cumbersome to obtain, yet they are necessary to triangulate corresponding pixels in 3D space, which is at the core of all best performing MVS algorithms. Concretely, Dust3r tackles the problem of dense and Unconstrained Stereo 3D Reconstruction of Arbitrary Image Collections which is essentially a specialized extension of the general Multi-View Stereo (MVS) problem.

At the core of DUST3R is the idea of directly learning a mapping from input images to the desired output representation, namely point maps.

Pointmap. A *pointmap* is a dense 2D grid of 3D points, represented by $X \in \mathbb{R}^{W\times H\times 3}$. Each pointmap corresponds to an RGB image I of size $W \times H$, establishing a one-to-one relation between image pixels and 3D scene points. In other words, for every pixel coordinate $(i, j) \in \{1, \ldots, W\} \times \{1, \ldots, H\}$, the pixel value $I_{i,j}$ is associated with a 3D point $X_{i,j}$.

Network architecture Dust3r is inspired by CroCo [17], which introduces a self-supervised pretraining strategy for dense correspondence learning, and the model directly benefits from CroCo's pretrained features. As shown in figure [**dust3r_figure**], the input of Dust3r are two images, which are transformed in a siamese manner by the same weight-sharing ViT encoder yielding two token representations F^1 and F^2:

$$F^1 = Encoder(I^1), F^2 = Encoder(I^2) \tag{3.1}$$

Then, two decoder heads take the output tokens of the encoder for each image and performs self attention and cross attention and finally feed tokens to an MLP and outputs a pointmap and an associated confidence map

$$X^{1,1}, C^{1,1} = \text{Head}^1\left(G_0^1, \ldots, G_B^1\right), \tag{3.2}$$

$$X^{2,1}, C^{2,1} = \text{Head}^2\left(G_0^2, \ldots, G_B^2\right). \tag{3.3}$$

3D Regression loss. DUSt3R supervises predicted pointmaps $X^{v,1}$ against ground-truth $\bar{X}^{v,1}$ on valid pixels $\mathcal{D}^v$. The scale factors z and $\bar{z}$ are defined as the mean norms of $X^{v,1}$ and $\bar{X}^{v,1}$:

$$z = \frac{1}{|\mathcal{D}^1|} \sum_{i \in \mathcal{D}^1} \|X_i^1\|, \quad \bar{z} = \frac{1}{|\mathcal{D}^2|} \sum_{i \in \mathcal{D}^2} \|\bar{X}_i^2\|.$$

The regression loss is then

$$\ell_{\text{regr}}(v,i) = \left\| \tfrac{1}{z} X_i^{v,1} - \tfrac{1}{\bar{z}} \bar{X}_i^{v,1} \right\|,$$

ensuring that both predicted and ground-truth 3D points are *scale-normalized* before measuring error.

Confidence-aware loss. Not all pixels correspond to reliable 3D points (e.g., sky, textureless or occluded regions). DUSt3R learns a confidence weight $C_i^{v,1}$ to down-weight uncertain areas:

$$\mathcal{L}_{\text{conf}} = \sum_v \sum_{i \in \mathcal{D}^v} \left(C_i^{v,1} \, \ell_{\text{regr}}(v,i) - \alpha \log C_i^{v,1} \right), \tag{3.4}$$

with $C_i^{v,1} = 1 + \exp(\tilde{C}_i^{v,1}) > 1$ ensuring positivity. This balances learning by emphasizing confident pixels while regularizing noisy predictions.

3.2 Data-Driven Models for Static Scenes

At the core of DUST3R is the idea of directly learning a mapping from input images to the desired output representation, namely point maps. Crucially, in contrast to classical pipelines where learning is often used as a supporting component, in DUST3R learning constitutes the foundation of the algorithm itself. The model employs a Transformer encoder–decoder architecture trained to predict point maps from image collections, leveraging established computer vision datasets such as ScanNet++ [**yeshwanth2023scannet**] and Habitat [18].

Several methods have recently been proposed as extensions of DUST3R, among which MAST3R [19] is a direct follow-up. A key characteristic of DUST3R is that the predicted point maps are defined only up to an unknown global scale factor. Consequently, they cannot be directly employed in applications that require metric distances, such as autonomous driving, where accurate measurements (e.g., the distance between a vehicle and a pedestrian) are critical.

This limitation arises from the normalization procedure used during training. Since the training data originates from diverse cameras with different intrinsic parameters, the corresponding ground-truth point maps naturally appear at varying scales. To stabilize optimization and avoid divergence, DUST3R normalizes both the predicted point maps $X_i^{v,1}$ and the ground-truth point maps $\bar{X}_i^{v,1}$ before comparing them. This is formalized in the regression loss:

$$\ell_{\text{regr}}(v,i) = \left\| \frac{1}{z} X_i^{v,1} - \frac{1}{\bar{z}} \bar{X}_i^{v,1} \right\|, \tag{3.5}$$

where z and $\bar{z}$ denote the normalization factors applied to the predicted and ground-truth point maps, respectively, defined as the mean distance of all valid 3D points to the origin.

While this normalization stabilizes training, it inherently removes absolute scale information, leading to non-metric predictions.

MAST3R modifies the regression loss to ignore normalization for the predicted pointmaps when the ground-truth pointmaps are known to be metric. MAST3R also proposed to improve the downstream task of pixel correspondances by adding a matching head to DUST3R that outputs two dense feature maps D^1 and $D^2 \in \mathbb{R}^{H \times W \times d}$ of dimensional d:

$$D^1 = \text{Head}^1_{\text{desc}}\left([H^1, H'^1]\right), \tag{3.6}$$

$$D^2 = \text{Head}^2_{\text{desc}}\left([H^2, H'^2]\right). \tag{3.7}$$

trained using the infoNCE loss [20] to encourage each local descriptor from one image to match with at most a single descriptor from the other image that represents the same 3D point in the scene.

While DUST3R introduced a novel paradigm in geometric computer vision by unifying multiple 3D vision tasks, and MAST3R extended this framework with additional heads for feature matching and metric reconstruction, both methods operate on image pairs. Their outputs consist of local 3D reconstructions that subsequently need to be aligned within a global coordinate system. However, the number of image pairs grows quadratically with the size of the image collection, which introduces significant challenges for global alignment, particularly in terms of robustness and computational efficiency for large-scale datasets.

To address this limitation, MUST3R [21] proposes a multi-view network for stereo 3D reconstruction. The architecture modifies DUST3R by enforcing symmetry and extending the formulation to jointly process multiple views, thereby directly predicting a consistent 3D structure for all images in a shared coordinate frame.

Similar to how MAST3R builds upon the ideas of DUST3R by improving a specific downstream task, and how MUST3R further extends the framework beyond image pairs, these developments collectively highlight the effectiveness of 3D point maps as a powerful output representation.

3.3 DUSt3R-like models for Dynamic sequences

An important limitation of DUST3R is that it was trained exclusively on static scenes. As a result, the model does not generalize well to dynamic environments where objects may deform or change geometry over time. This restricts its applicability in real-time understanding tasks, since the resulting 3D reconstructions become inaccurate when the same object appears across frames with varying shapes or poses.

MonST3R [22] was one of the first approaches to tackle the problem of novel view synthesis (NVS) in dynamic scenes, using DUST3R as its backbone. The authors argued that the limitations of DUST3R in dynamic settings did not stem from the architecture itself, but rather from its training data, which was designed exclusively for static scenes. By training on data better suited for dynamic environments, the model could be extended to handle temporal changes effectively.

The key distinction from DUST3R is that in MonST3R each point map corresponds to a single point in time. A refinement stage is then introduced to align these temporally

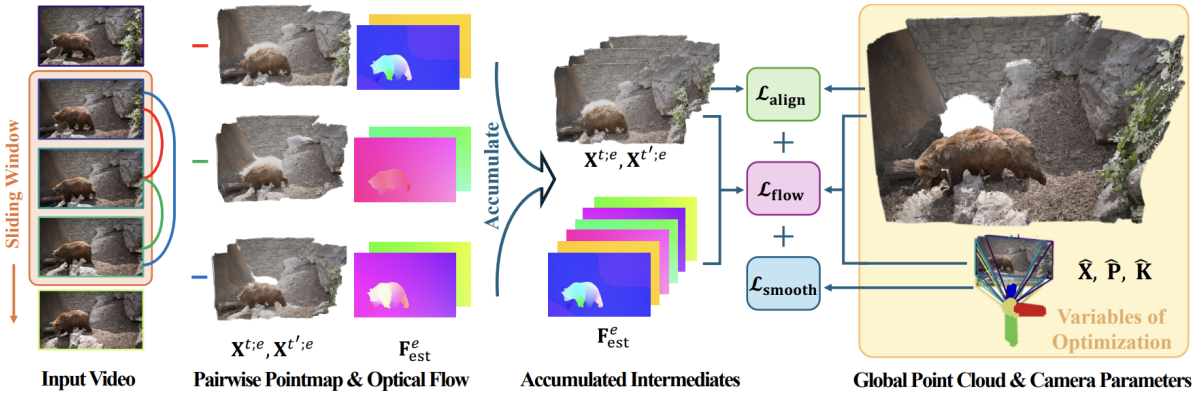

Figure 3.2: **Dynamic global point cloud and camera pose estimation.** Using a fixed-length temporal window, MonST3R computes pairwise pointmaps for each frame pair, complemented with optical flow from an external method. These pairwise estimates are combined to jointly optimize a global point cloud and per-frame camera poses, from which dense video depth can be directly obtained. Figure from [22].

local point maps into a consistent global representation, while simultaneously estimating the associated camera poses and intrinsics.

The model cannot directly produce globally aligned point maps, which makes an additional optimization step necessary to account for the influence of dynamic objects in the predictions. This optimization jointly estimates a global dynamic point cloud and the corresponding camera poses by exploiting both the pairwise predictions of the model and the inherent temporal structure of video sequences.

In DUST3R, global alignment is achieved through a connectivity graph constructed over all pairwise frame combinations. However, this strategy becomes computationally prohibitive for long video sequences. MonST3R addresses this limitation by introducing a sliding temporal window as showed in figure 3.2, thereby reducing the number of frame combinations considered at once, and by employing strided sampling to further improve runtime efficiency.

To ensure consistency between frames, it is common to introduce a motion signal that helps the model recognize when two pixels, observed at different positions in separate frames, correspond to the same 3D point. The apparent displacement in image space is then attributed to camera motion. However, static-object assumptions break down when objects in the scene move independently, since pixel displacements can arise from either camera motion or object motion. This ambiguity makes the optimization ill-posed.

State-of-the-art methods address this by learning a motion mask in an uncertainty formulation, down-weighting pixels that do not follow the expected camera-induced flow, thereby reducing confusion. MonST3R uses a flow projection loss to encourage the global pointmaps and camera poses to be consistent with the estimated flow for the confident, static regions of the actual frames. More precisely, given two frames t, t', using their global pointmaps, camera extrinsics and intrinsics, they compute the flow fields from taking the global pointmap $\mathbf{X}^t$, assuming the scene is static, and then moving the camera from t to t'. Then they can

encourage this to be close to the estimated flow, $\mathbf{F}_{\text{est}}^{t\to t'}$, in the regions which are confidently static $\mathbf{S}^{\text{global};t\to t'}$ according to the global parameters:

$$\mathcal{L}_{\text{flow}}(\mathbf{X}) = \sum_{W^i \in W} \sum_{t\to t' \in W^i} \left\| \mathbf{S}^{\text{global};t\to t'} \cdot \left(\mathbf{F}_{\text{cam}}^{\text{global};t\to t'} - \mathbf{F}_{\text{est}}^{t\to t'} \right) \right\|_1, \tag{6}$$

where $\mathbf{F}_{\text{cam}}^{\text{global};t\to t'}$ denotes the flow induced by the predicted geometry and camera motion, and $\mathbf{F}_{\text{est}}^{t\to t'}$ is the optical flow estimated using an off-the-shelf flow network. To identify confident static regions, MonST3R compares these two flows at the pixel level. Specifically, it computes the per-pixel difference $\|\mathbf{F}_{\text{cam}}^{\text{global};t\to t'} - \mathbf{F}_{\text{est}}^{t\to t'}\|_2$ and applies a threshold τ. Pixels where the difference falls below τ are considered confidently static, while pixels above this threshold are marked as non-static. The confident static mask is therefore defined as:

$$\mathbf{S}^{\text{global};t\to t'} = \left[\|\mathbf{F}_{\text{cam}}^{\text{global};t\to t'} - \mathbf{F}_{\text{off}}^{t\to t'}\|_2 < \tau \right],$$

where the bracket notation represents the indicator function, returning 1 when the condition is satisfied and 0 otherwise. Under this formulation, if the predicted flow and the observed flow are equal, their difference is zero, which is less than τ, and the pixel is thus labeled as confidently static.

In addition to the flow constraint described above, MonST3R introduces an alignment constraint and a smoothness constraint.

Alignment constraint First, they adopt the alignment term from DUST3R, which seeks a single rigid transformation $\mathbf{P}^{t:e}$ that aligns each pairwise estimation with the world-coordinate point maps. Since both $\mathbf{X}^{t:t'+e'}$ and $\mathbf{X}^{t':t'+e'}$ are expressed in the same camera coordinate frame, this alignment ensures consistency across pairwise reconstructions.

Smoothness constraint Second, they incorporate a camera trajectory smoothness loss to encourage temporally coherent camera motion. This loss penalizes abrupt changes in camera rotation and translation between consecutive frames:

$$\mathcal{L}_{\text{smooth}}(\mathbf{X}) = \sum_{t=0}^{N} \left(\left\| \mathbf{R}_t^\top \mathbf{R}_{t+1} - \mathbf{I} \right\|_F + \left\| \mathbf{T}^{t+1} - \mathbf{T}^t \right\|_2 \right), \tag{5}$$

where the Frobenius norm $\|\cdot\|_F$ measures the deviation in rotation, the L2 norm $\|\cdot\|_2$ measures the difference in translation, and $\mathbf{I}$ denotes the identity matrix.

Combining these terms, the complete optimization objective for estimating the dynamic global point cloud and camera poses is:

$$\hat{\mathbf{X}} = \underset{\mathbf{X},\mathbf{P}_W,\sigma}{\arg\min}\ \mathcal{L}_{\text{align}}(\mathbf{X},\sigma,\mathbf{P}_W) + w_{\text{smooth}}\mathcal{L}_{\text{smooth}}(\mathbf{X}) + w_{\text{flow}}\mathcal{L}_{\text{flow}}(\mathbf{X}), \tag{7}$$

where w_{smooth} and w_{flow} are scalar weights that balance the contributions of the smoothness and flow losses, respectively.

MegaSAM [23], a concurrent work to MonST3R, adopts a deep visual SLAM formulation with a differentiable bundle adjustment (BA) layer to jointly optimize camera poses, focal length, and disparity maps from monocular video, as it can be seen in figure 3.3. Given a frame pair (I_i, I_j), rigid-motion correspondences are computed as:

$$u_{ij} = \pi\big(G_{ij} \circ \pi^{-1}(p_i, d_i, K^{-1}), K\big),$$

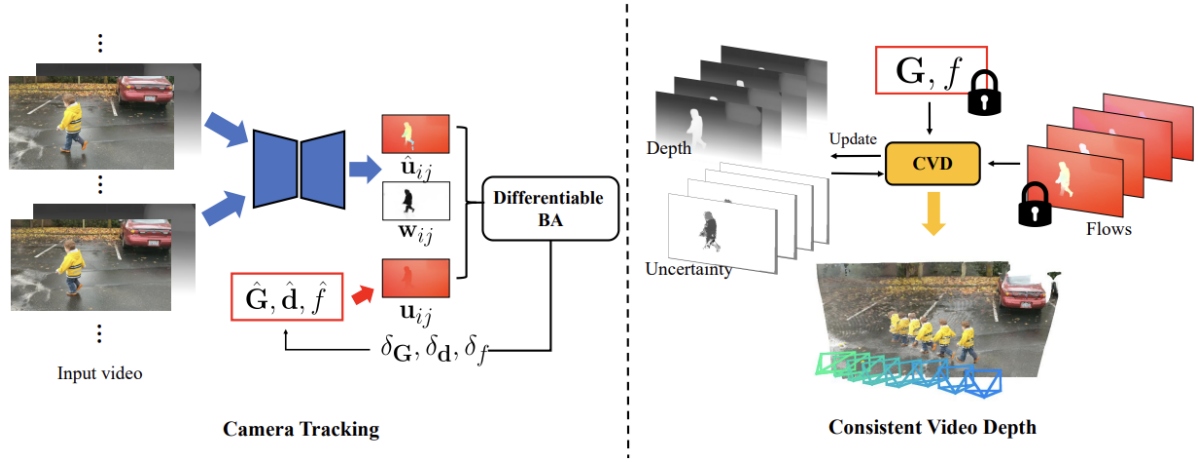

Figure 3.3: The system operates in two stages. *Camera tracking (left):* A differentiable bundle adjustment (BA) layer iteratively updates camera poses $\hat{G}$, focal length $\hat{f}$, and disparities $\hat{d}$ by minimizing reprojection error between predicted optical flow $\hat{u}_{ij}$ and motion-induced flow u_{ij}, guided by confidence w_{ij} and motion probability maps. *Consistent video depth estimation (right):* With fixed camera parameters, a first-order optimization refines per-frame depth and uncertainty maps using flow, temporal consistency, and prior losses, producing dense, consistent depth maps for the entire video. Figure from [23]

where G_{ij} is the relative pose between frames, d_i is disparity, and K is the intrinsic matrix. Optimization minimizes the weighted reprojection loss:

$$C(G, d, f) = \sum_{(i,j)\in P} \|\hat{u}_{ij} - u_{ij}\|^2_{\Sigma_{ij}},$$

MegaSAM integrates monocular depth priors and motion probability maps to down-weight dynamic regions, improving robustness in videos with low parallax or moving objects. A second stage refines video depth through optimization of flow and depth consistency:

$$C_{\text{cvd}} = w_{\text{flow}} C_{\text{flow}} + w_{\text{temp}} C_{\text{temp}} + w_{\text{prior}} C_{\text{prior}},$$

where C_{flow} enforces reprojection consistency, C_{temp} encourages temporal depth stability, and C_{prior} regularizes disparity with monocular predictions.

Both MonST3R and MegaSAM rely on optimization stages rather than using raw network predictions, but MegaSAM demonstrates superior performance by tightly coupling learned features with differentiable optimization.

3.4 Feed-Forward Models

In contrast, a parallel line of research has emerged that focuses on models capable of directly predicting camera poses, 3D point maps, and other 3D vision outputs in a single forward pass, without requiring an explicit optimization stage.

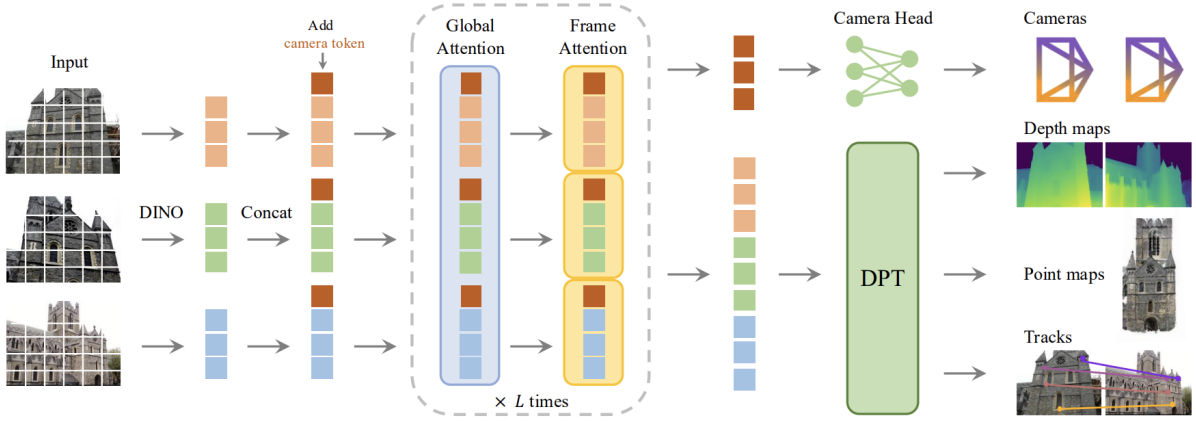

Figure 3.4: **Architecture overview.** The model encodes input images into patch tokens using DINO features and augments them with camera tokens for pose prediction. It alternates between frame-wise and global self-attention layers to capture both local and scene-level context. A dedicated camera head predicts camera intrinsics and extrinsics, while a DPT [27] head produces dense outputs. Figure from [26]

The *Bitter Lesson*, as articulated by Sutton [24], argues that models which fully exploit large-scale data and leverage scalable, parallelized training will ultimately outperform those that rely heavily on human-designed inductive biases. In other words, approaches that depend on handcrafted rules inspired by human understanding of a task are eventually surpassed by data-driven methods capable of learning directly from massive datasets.

Early approaches to 3D computer vision often relied on handcrafted representations such as edge detection, generalized cylinders, or local descriptors like SIFT features [25]. Predicting camera poses, globally aligned point maps, and other 3D quantities was traditionally treated as a divide-and-conquer problem: one technique would be developed for feature extraction, another for point matching, and these components would then be integrated into a multi-stage pipeline to produce the final reconstruction.

Although widely adopted in both academia and industry, such pipelines suffered from notable drawbacks. Errors accumulated across stages, leading to degraded performance, and the information produced at one stage could not be seamlessly reused in another, even when it might have been beneficial. This lack of end-to-end integration limited the overall robustness and adaptability of these systems.

3.4.1 VGGT

The Visual Geometry Grounded Transformer (VGGT) [26] is a feed-forward Transformer architecture designed to overcome these limitations. In a single forward pass, VGGT is capable of jointly predicting camera pose estimates, globally aligned point maps, depth maps, and point tracks, thereby unifying multiple geometric vision tasks within one network.

Trained over 16 datasets it shows promising results on several key benchmarks of key 3D computer vision tasks.

3.4.2 Tokenization and Attention Mechanism

Given a set of N input RGB images $\{I_i\}_{i=1}^N$, VGGT first encodes each image $I_i \in \mathbb{R}^{3\times H\times W}$ into a set of K patch tokens $\{t_{i,k}\}_{k=1}^K$ using a DINOv2 encoder. For each frame, an additional camera token t_i^g and four register tokens $t_{i,j}^R$ are appended, forming the token set

$$T_i = \{t_{i,1}, \ldots, t_{i,K},\ t_i^g,\ t_{i,1}^R, \ldots, t_{i,4}^R\}.$$

The union of all token sets $T = \bigcup_{i=1}^N T_i$ is then processed by the transformer backbone, which alternates between *frame-wise self-attention* and *global self-attention*. Frame-wise self-attention operates within each T_i independently, while global self-attention attends jointly across all tokens in T, thereby enforcing cross-frame coherence, as illustrated in figure 3.4.

Unlike typical transformers, VGGT avoids explicit frame index embeddings, preserving *permutation equivariance*: the predictions remain invariant to the order of the input images. This also allows for flexible input length, enabling the model to handle varying numbers of views. The model can still implicitly cluster tokens by frame through the learned camera and register tokens, which ensures per-frame grouping without requiring explicit positional encoding.

The alternating-attention transformer produces features that are used for four different tasks, each parameterized by a dedicated prediction head. For the dense prediction tasks—depth maps D_i, point maps P_i, and tracking features T_i—VGGT first converts the output tokens into dense feature maps F_i using a DPT layer [27]. These feature maps are then mapped to depth and point maps through convolutional layers, while the tracking features are processed by a CoTracker2 [28] based head to perform point tracking. The remaining prediction head uses the camera tokens to regress the camera parameters g_i for each frame.

4 AnyCam

4.1 Self-supervised Learning for 3D Vision

The Bitter Lesson. Among the different paradigms for training deep learning models, self-supervised learning stands out as the approach most consistent with the *Bitter Lesson*. The Bitter Lesson states that progress in AI arises primarily from methods that scale with data and compute, rather than from approaches that rely heavily on handcrafted domain knowledge. Self-supervised learning embodies this principle by eliminating the need for costly human annotations, instead deriving supervisory signals directly from raw data through pretext tasks such as masked token prediction, contrastive alignment, or temporal ordering.

This makes self-supervised learning fundamentally different from traditional supervised learning, which is limited by the availability and potential bias of labeled datasets. It also differs from classical unsupervised learning, which typically emphasizes clustering or density estimation without explicitly shaping representations for transfer. In contrast, self-supervised learning leverages the abundance of unlabeled data while still producing strong and generalizable representations, which can be effectively adapted to a wide range of downstream tasks.

VGGT [26] follows simple design choices that allow the network to decide how to represent the information learned from the training datasets. Although it uses a transformer [29] as its core component, a key difference from most current foundation models lies in its supervised training regime. The minimalist design, with few inductive biases, enables the model to learn strong representations from data. However, because it cannot exploit the vast amount of unlabeled images and videos available on the internet, it becomes more susceptible to the biases inherent in the labeled datasets it relies on. While one might argue that the training data is sufficient, scaling laws consistently show that advances come from scaling both compute and data. If the model depends only on supervised datasets, this reliance becomes a bottleneck that prevents the development of stronger models.

VGGT therefore integrates several important aspects that move it toward being a foundation model for 3D vision, as its representations are versatile enough to support multiple 3D tasks. Yet, its supervised training setup limits scalability to internet-scale data, which remains largely unlabeled.

In this regard, state-of-the-art approaches that leverage unannotated datasets, such as AnyCam [30], highlight an alternative path forward.

AnyCam tackles the problem of relative camera pose estimation as well as intrinsics estimation given a video with dynamic objects in a feed forward manner. The model tries to tackle the inability of current methods like DUSt3R to model dynamic objects and the heavily reliance on annotated data. As an alternative AnyCam is a fast transformer model that directly estimates camera poses and intrinsics from a dynamic video sequence in feed-forward fashion. The intuition is that such network can exploit the relation between the optical flow

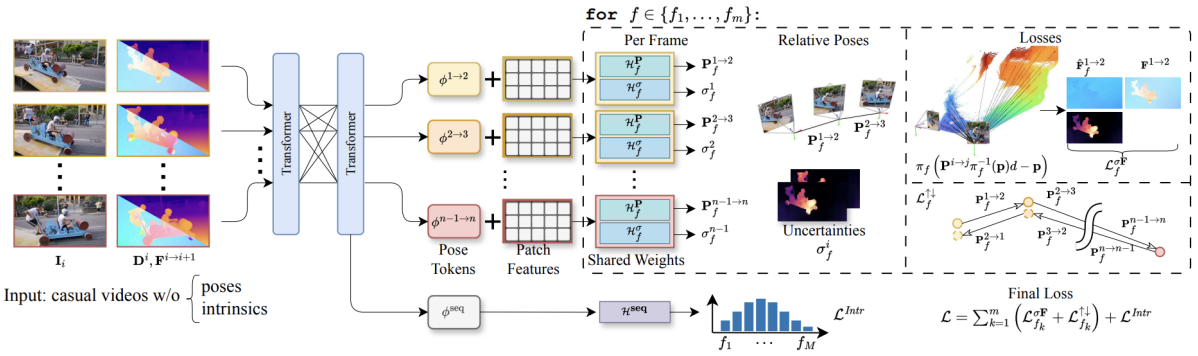

Figure 4.1: For each frame, AnyCam uses depth and optical flow estimated by off-the-shelf networks. Self-attention produces pose tokens encoding motion between frames $n-1$ and n. For every pose token and focal length f, the model predicts a relative camera pose and an uncertainty map, with intrinsics $f \in \{f_1, \ldots, f_{32}\}$.
Figure from [30]

and the movement of the camera across frames to learn strong priors over realistic camera poses.

4.2 AnyCam Model Architecture

In the following, we briefly recap the AnyCam model as we extend its architecture.

AnyCam assumes a simplified pinhole model that is constant for the duration of a sequence. Therefore, camera intrinsics for a video are modeled via a single focal length $f \in \mathbb{R}_+$. AnyCam takes as input a sequence of n video frames $I^i \in ([0,1]^3)^\Omega$, $i \in N$, where $N = \{1, \ldots, n\}$ denotes the set of frame indices and $\Omega = \{1, \ldots, H\} \times \{1, \ldots, W\}$ denotes the pixel lattice. Using off-the-shelf depth and optical flow predictors, it further obtain dense depth maps $D^i \in \mathbb{R}_+^{1 \times H \times W}$, as well as dense optical flows $F^{i \to j} \in \mathbb{R}^{2 \times H \times W}$ from any frame i to j.

AnyCam is based on a transformer architecture, which as input receives frames I^i, depth predictions D^i, and flow predictions $F^{i \to i+1}$ for a short sequence, as can be seen in figure 4.1.

First, a backbone extracts features for each timestep separately. Stacked self-attention layers allow the model to then exchange information between the frames, producing $n-1$ pose tokens $\phi^{i \to i+1}$, $i \in \{1, \ldots, n-1\}$ and updated feature maps. Additionally, a single sequence token ϕ^{seq} is predicted.

To decode the features, a frame prediction head $\mathcal{H} = (\mathcal{H}^P, \mathcal{H}^\sigma)$ outputs the 6DOF camera poses $\mathbf{P}^{i \to i+1}$ and pixel-wise uncertainty maps $\sigma^i \in \mathbb{R}_+^\Omega$ for every timestep. A sequence head $\mathcal{H}^{\text{seq}}$ additionally determines a single set of camera intrinsics for the video.

Because of the difficulty of predicting directly the intrinsics parameterized by the focal length, AnyCam uses 32 different heads each encoding a specific focal length.

Objectives of the loss function. AnyCam is trained on large, unlabelled datasets. The objectives of the loss functions are two fold. Leveraging multi-view information to recover

local camera motion between adjacent frames and using the context of longer sequences to learn realistic long-range camera motion patterns.

Uncertainty-aware flow loss. Using the predicted relative pose $\mathbf{P}_f^{i\to j}$ between two frames i and j, and the corresponding depth map $\mathbf{D}^i$, one can project all pixel locations p_{uv} from frame i into frame j. Subtracting the original pixel locations from the projected pixel locations $p'_{f,uv}$ yields the induced optical flow $\hat{\mathbf{F}}_f^{i\to j}$:

$$\hat{\mathbf{F}}_{f,uv}^{i\to j} = p'_{f,uv} - p_{uv} = \pi_f\left(\mathbf{P}_f^{i\to j}\pi_f^{-1}(p_{uv}, d_{uv})\right) - p_{uv}, \tag{4.1}$$

where π_f and π_f^{-1} denote the projection and back-projection functions, respectively.

In a static world, the induced flow formulation would drive the predicted camera pose to the real quantity in order to match the reference optical flow $\mathbf{F}_f^{i\to j}$. However, dynamic objects lead to inconsistencies in the reference optical flow, which would deteriorate gradients during optimization. AnyCam proposes to model the inconsistencies produced by dynamic objects using aleatoric uncertainty using the predicted uncertainty map $\mathbf{U}^i$.

Formally, for a pair of frames i and j, AnyCam loss function is:

$$\ell_{uv}^{\mathbf{F}_f^{i\to j}} = \left\|\hat{\mathbf{F}}_{f,uv}^{i\to j} - \mathbf{F}_{f,uv}^{i\to j}\right\|_1, \tag{4.2}$$

$$\mathcal{L}_f^{\mathbf{F}_f^{i\to j}} = -\frac{1}{|\Omega|}\sum_{uv\in\Omega}\ln\frac{1}{\sqrt{2}\sigma_{f,uv}^i}\exp\left(-\frac{\sqrt{2}\,\ell_{f,uv}^{\mathbf{F}_f^{i\to j}}}{\sigma_{f,uv}^i}\right), \tag{4.3}$$

which is then summed up over every pair of neighboring frames within the sequence:

$$\mathcal{L}_f^{c\mathbf{F}} = \sum_{i=1}^{n-1}\mathcal{L}_f^{\mathbf{F}_f^{i\to i+1}}. \tag{4.4}$$

Intuitively, the model reduces the influence of regions in the input frame that are expected to have high loss, such as dynamic objects. As a result, pose supervision is primarily guided by regions that are reliably represented through the induced flow.

Learning camera motion patterns. The AnyCam uncertainty-aware flow loss trains the network for camera pose estimation but does not explicitly encourage the use of information from neighboring frames. Optical flow predicted by off-the-shelf networks may be inaccurate, and static scene regions can be occluded by dynamic objects. To promote sequence-level reasoning, AnyCam introduces a forward–backward consistency loss and a dropout scheme.

The consistency loss enforces agreement between forward and backward pose predictions. When reversing the frame order, the relative pose $\mathbf{P}_f^{j\to i}$ should equal the inverse of the original pose $(\mathbf{P}_f^{i\to j})^{-1}$. This is implemented with an ℓ_1 loss:

$$\mathcal{L}_f^{\uparrow\downarrow} = \sum_{i=1}^{n-1}\left\|\left(\mathbf{P}_f^{i\to i+1}\right)^{-1}\mathbf{P}_f^{i+1\to i} - \mathbf{I}_4\right\|_{1,1}, \tag{4.5}$$

where $\mathbf{I}_4$ is the 4×4 identity matrix and $\|\cdot\|_{1,1}$ denotes the element-wise ℓ_1 norm. This constraint improves robustness by encouraging geometric consistency between predictions.

To further strengthen temporal reasoning, a dropout scheme is applied to the pose tokens during training. Each token element is set to zero with probability p_{drop}, forcing the network to exchange information across frames through the self-attention layers. This regularization helps the model learn realistic motion patterns and reduces overfitting to noise in the input, while leaving uncertainty maps unaffected.

Finding the best candidate To find the best candidate focal length, AnyCam uses the magnitude of $\mathcal{L}_f^{\sigma\mathbf{F}}$.

After the model has converged, the magnitude of $\mathcal{L}_f^{\sigma\mathbf{F}}$ will expresses how well f is to the true but unknown actual focal length. AnyCam trains H^{seq} to predict which of the focal candidates will yield the lowest flow loss and it trains it using the Kullback diverege loss:

$$\mathcal{L}^{\text{Intr}} = \mathbf{KL}_{\text{div}}\left(\mathcal{P}, \text{softmax}\left(-\mathcal{L}_{f_1}^{\sigma\mathbf{F}}, \ldots, -\mathcal{L}_{f_m}^{\sigma\mathbf{F}}\right)\right) \tag{4.6}$$

where $\mathcal{P} = (p_{fi}, ..., p_{fm})$ is the output of H^{seq}:

Final loss. The final AnyCam loss to work with unlabelled video sequences is defined by:

$$\mathcal{L} = \sum_{k=1}^{m}\left(\lambda_{\sigma\mathbf{F}}\mathcal{L}_{f_k}^{\sigma\mathbf{F}} + \lambda_{\uparrow\downarrow}\mathcal{L}_{f_k}^{\uparrow\downarrow}\right) + \lambda_{\text{Intr}}\mathcal{L}^{\text{Intr}} \tag{4.7}$$

Towards a foundational model for 3D vision. Using videos from platforms such as YouTube is not straightforward, since the geometry of the scene depends on the intrinsic parameters of the recording camera, which are typically unknown and therefore complicate the use of this data. Furthermore, modeling dynamic objects introduces an additional challenge, as it requires accounting for geometry that changes over time. VGGT does not encounter these issues, as it is restricted to static scenes. By contrast, AnyCam explicitly incorporates uncertainty modeling into its formulation and automatically filters out dynamic objects. This design enables effective training on large collections of videos sourced from YouTube and other casual recordings, ultimately leading to strong generalization capabilities. Importantly, this also addresses a key requirement for building foundation models in 3D computer vision: AnyCam's formulation allows training directly on unannotated internet-scale video data, such as those available on YouTube.

5 Method

5.1 Problem Definition

In this work, the focus is on 3D point estimation. Given an RGB image I defined by the pixel lattice $\Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ representing a dense 2D grid, the goal is to predict a corresponding 3D point for every image pixel, establishing a one-to-one correspondence between pixels and scene points. For each pixel coordinate $(u, v) \in \Omega$, the task is to predict the 3D point X_{uv} associated with the pixel value I_{ij}.

5.2 3D Point Maps

AnyCam does not directly predict 3D point maps for dynamic scenes. Instead, they can be recovered from the estimated camera intrinsics together with per-frame depth predictions, as follows: For a given frame $i \in N$, let $\Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ denote the pixel lattice and $p = (u, v) \in \Omega$ a pixel coordinate. Define the homogeneous image coordinate

$$\tilde{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix},$$

and let $K_f \in \mathbb{R}^{3\times3}$ denote the intrinsics matrix parameterized by focal length f. Given the depth value $z_i(p)$ at pixel p, the corresponding 3D point in camera coordinates is obtained via unprojection:

$$X_f^i(p) = z_i(p)\, K_f^{-1}\, \tilde{p}. \tag{5.1}$$

The full point map of frame i is then defined as

$$M_f^i = \{X_f^i(p) \mid p \in \Omega\}, \quad M_f^i \in \mathbb{R}^{3\times H\times W}.$$

Repeating this process for all frames $i \in N$ yields a sequence of point maps $\{M_f^i\}_{i\in N}$, each expressed in the corresponding camera coordinate frame. These maps provide dense 3D structure estimates that can be used to supervise or evaluate the predictions of our extended AnyCam model.

DPT for dense predictions. AnyCam does not directly output point maps, but relies on intrinsics and depth (Eq. 5.1). We modify AnyCam to also predict 3D pointmaps and refer to this model as AnyCam-3D. To equip the model with this capability, we add a Dense Prediction Transformer (DPT) head.

Figure 5.1 illustrates the AnyCam-3D model. AnyCam-3D receives as input frames I_i, depth predictions D_i, and flow predictions $F_{i\to i+1}$ for a short sequence. A DinoV2 backbone extracts feature maps for each timestep separately. Let $\{F_\ell^i\}_{\ell\in\{1,4,7,12\}}$ be the feature maps from the DinoV2 backbone at layers ℓ for frame $i \in N$. The DPT aggregates these multi-scale features into a dense representation $\hat{F}^i \in \mathbb{R}^{C\times H\times W}$, which is then processed by convolutional layers to predict a point map:

$$\hat{M}_i \in \mathbb{R}^{3\times H\times W}, \quad i \in N.$$

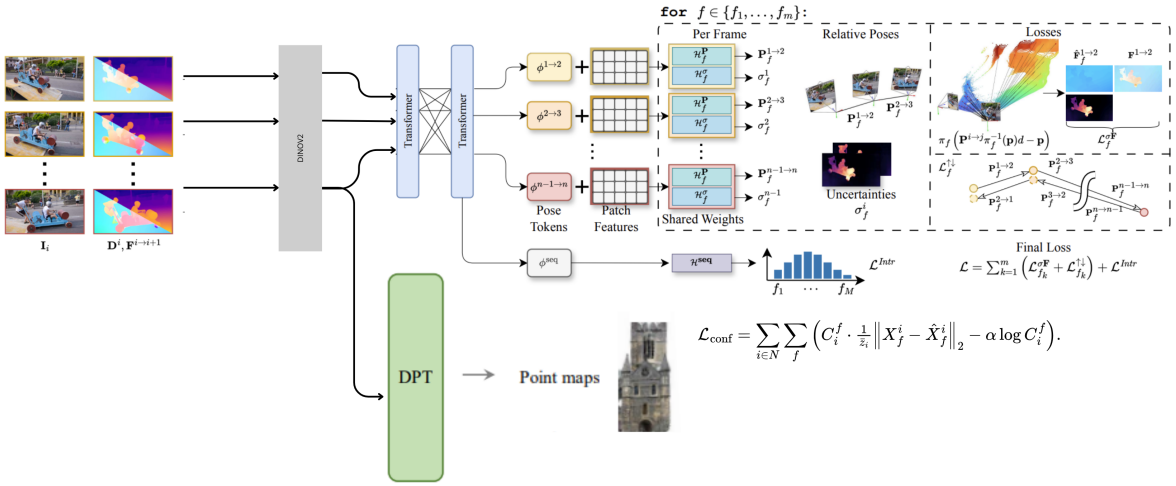

Figure 5.1: **AnyCam-3D architecture.** AnyCam-3D extends AnyCam to directly predict dense 3D point maps. The model takes as input a short image sequence (I_i), predicted depth maps (D_i), and optical flow $(F_{i\to i+1})$. A DinoV2 backbone extracts multi-scale features (F_ℓ^i) for each frame, which are aggregated by a Dense Prediction Transformer (DPT) head to produce a dense feature map $\hat{F}^i \in \mathbb{R}^{C\times H\times W}$. From this representation, convolutional layers output a point map $\hat{M}_i \in \mathbb{R}^{3\times H\times W}$, with optional confidence estimates.

5.2.1 Point Map Prediction

From the dense features $\hat{F}^i$ produced by the DPT head, we explore two parameterizations for the predicted 3D point maps:

(1) Direct regression. We predict four output channels of size $H \times W$: three channels for the 3D coordinates (x, y, z) and one for a per-pixel confidence map σ^i. These are obtained via a 3×3 convolution on $\hat{F}^i$.

(2) Multi-focal regression. The original AnyCam models intrinsics using 32 focal length candidates $\{f_1, \ldots, f_{32}\}$. Inspired by this, we predict 66 channels: the first 64 encode (x, y) coordinates for each focal candidate (32×2), channel 65 encodes the shared depth z, and channel 66 encodes the per-pixel confidence σ^i.

Global Attention. We also experiment with adding a global attention mechanism before the DPT head. Specifically, we apply self-attention across all tokens extracted from the DinoV2 layers, allowing information exchange between them prior to aggregation in the DPT module.

In Section 6.3, we report results both with and without this global attention step, highlighting its effect on point map prediction quality.

5.2.2 Loss Formulation

We evaluate three formulations for supervising point map prediction: a 3D regression loss (with a confidence-aware extension), a consistency loss and a depth loss.

3D Regression Loss

Following MASt3r, we regress predicted point maps against ground truth 3D points, normalized by the mean depth to preserve metric scale. For each frame $i \in N$ and focal length candidate $f \in \{f_1, \ldots, f_{32}\}$, let X_f^i be the ground truth point map from Eq. 5.1, and $\hat{X}_f^i$ the prediction. The loss is defined as

$$\mathcal{L}_{\text{regr}} = \sum_{i \in N} \sum_{f} \frac{1}{\bar{z}_i} \left\| X_f^i - \hat{X}_f^i \right\|_2, \tag{5.2}$$

where $\bar{z}_i$ is the mean norm in frame i for X_f^i.

Extending this formulation as in DUSt3R, we weight the MASt3r loss with predicted confidence maps C_i and add a regularization term:

$$\mathcal{L}_{\text{conf}} = \sum_{i \in N} \sum_{f} \left(C_i^f \cdot \tfrac{1}{\bar{z}_i} \left\| X_f^i - \hat{X}_f^i \right\|_2 - \alpha \log C_i^f \right). \tag{5.3}$$

This loss keeps the MASt3r definition of metric depth through normalization by $\bar{z}_i$, while introducing confidence weighting to downweight uncertain regions and prevent trivial solutions.

Consistency Loss

The static regions of the predicted point maps should remain consistent across time when expressed in the coordinate frame of the next view. To enforce this, we introduce a 3D consistency loss defined per focal length candidate.

For each frame $i \in N - 1$ and focal candidate $f \in \{f_1, \ldots, f_{32}\}$, let $\hat{X}_f^i \in \mathbb{R}^{3 \times H \times W}$ denote the predicted point map for frame i assuming that comes from focal length f. AnyCam also predicts a relative camera pose

$$P_f^{i \to i+1} \in SE(3),$$

for each focal candidate f. This allows us to transform the point map from frame i into the coordinate system of frame $i + 1$:

$$\hat{X}_f^{i \to i+1} = P_f^{i \to i+1} \, \hat{X}_f^i.$$

To compare this with the prediction in frame $i+1$, we use the optical flow $F^{i\to i+1}$ to warp the point map X_f^{i+1}:

$$X_f^{i+1} = \text{Warp}\Big(X_f^{i+1}, F^{i\to i+1}\Big).$$

The per-frame, per-focal 3D consistency loss is then defined as

$$\ell_{3D}(i,f) = \big\| o_i \odot \big(\hat{X}_f^{i\to i+1} - X_f^{i+1}\big) \big\|_1, \tag{5.4}$$

where $o_i \in \{0,1\}^{H\times W}$ is the occlusion mask and $\odot$ denotes elementwise multiplication.

Aggregating across all frames and focal candidates gives the final consistency loss:

$$\mathcal{L}_{3D} = \sum_{i\in N-1} \sum_{f\in\{f_1,\ldots,f_{32}\}} \ell_{3D}(i,f). \tag{5.5}$$

Here both the point maps $\hat{X}_f^i$ and the relative poses $P_{i\to i+1}^f$ come directly from AnyCam, without requiring ground-truth transformations.

Consistency Loss Instability During training we observed instability when applying the consistency loss directly between the predicted point maps of frames i and $i+1$. In this setting, both point maps and relative poses are produced by the model. The network could exploit this by predicting degenerate poses $P_{i\to i+1} \approx 0$, which trivially drives the loss to zero and leads to collapsed point maps without geometric structure.

A simple workaround is to detach the pose estimates so that gradients do not flow into the pose branch. However, even with this modification we found that the model still converged to trivial solutions, producing point maps clustered in a single location. We hypothesize that relying solely on the consistency loss makes the optimization problem too difficult.

To address this, we implemented two stabilizing strategies. First, we relaxed the consistency constraint by modifying the formulation for frame $i+1$. Instead of using the full predicted point map $\hat{X}_f^{i+1}$ from the DPT, we reconstruct its coordinates as follows: the (x,y) components are obtained from the inverse intrinsics K_f^{-1} parametrized by each focal length candidate f, while the depth coordinate z is taken directly from the DPT prediction. Formally, for a pixel $p=(u,v)$,

$$X_f^{i+1}(p) \;=\; \begin{bmatrix} \big(K_f^{-1}\tilde{p}\big)_{xy} \cdot z^{i+1}(p) \\ z^{i+1}(p) \end{bmatrix}, \qquad \tilde{p} = (u,v,1)^\top.$$

This makes the optimization problem easier without removing the need for plausible (x,y) predictions. At each consistency step, the model must still produce valid (x,y) coordinates for frame i, since these are transformed and compared against frame $i+1$. The simplification is that the comparison is against a geometry where (x,y) are fixed analytically by the intrinsics, so only one side of the consistency check carries learnable (x,y). Crucially, every frame eventually plays the role of i in the sequence, meaning the network must learn globally consistent (x,y) coordinates across time. This reduces the degrees of freedom in the loss at each step, avoids trivial collapsed solutions, and still enforces coherent point maps over the entire video.

Depth Loss

To further stabilize training we introduce an auxiliary depth loss. This loss follows the same structure as the 3D regression loss, but it supervises only the depth (the z coordinate). Concretely, from the DPT head we take the channel corresponding to depth $\hat{Z}^i \in \mathbb{R}^{H \times W}$. By reducing the prediction task to one dimension, we simplify optimization and improve not only depth estimation but also the overall quality of the point maps, since the other losses depend on correct depth.

For each frame $i \in N$, let Z^i denote the ground-truth depth and $\hat{Z}^i$ the predicted depth. The per-frame depth regression loss is

$$\ell_{\text{depth}}(i) = \frac{1}{\bar{z}_i} \left\| Z^i - \hat{Z}^i \right\|_2, \tag{5.6}$$

where $\bar{z}_i$ is the mean depth of frame i.

Aggregating over all frames gives

$$\mathcal{L}_{\text{depth}} = \sum_{i \in N} \ell_{\text{depth}}(i). \tag{5.7}$$

As in the confidence-aware regression, we extend this with per-pixel confidence weights C^i:

$$\mathcal{L}_{\text{depth-conf}} = \sum_{i \in N} \left(C^i \cdot \ell_{\text{depth}}(i) \; - \; \alpha \log C^i \right). \tag{5.8}$$

This auxiliary loss prevents collapse along the z-axis and enforces metric depth consistency, directly supporting both the regression and the consistency objectives.

6 Experiments

6.1 Setup

6.1.1 Data

Our training formulation is the same as AnyCam. The model is trained with no labels, making it a key benefit as dynamic videos with 3D labels are scarce. We use YouTube-VOS [31], RealEstate10K [32], WalkingTours [33], OpenDV, [34] EpicKitchens [35] based on Youtube or GoPro.

Following existing works on 3D point estimation in dynamic environments we perform quantitative evaluation on the dynamic subset of TUM-RGBD [36]. Furthermore we perform qualitative evaluation on the Davis dataset.

6.1.2 Implementation Details

We rely on the recent UniDepth [37] and UniMatch [38] methods to obtain depth and flow maps. Our model is implemented in PyTorch [39] and is initialized with a pretrained AnyCam model trained with the 2 sequence case. We train the model using 2 frames. All frames are sampled at a resolution of 336 × 336 from the videos in the datasets. During inference, we crop the input video to squares of 336 and pass sequences of up to 10 frames to the model at once. The model is configured to use 32 focal length candidates ranging from 0.1H to 3.5H where H is the image height. Training converges after 250k iterations at a batch size of 11 (seq. len 2) two days on two NVIDIA A100 40GB GPUs. For a video of 50 frames, it takes around 1) 15 seconds to obtain flows and depths, and 2) 5 seconds for AnyCam to predict a trajectory and 3D point maps.

We trained the AnyCam model end-to-end with a multi-task loss. Since the camera pose loss had a larger scale than the consistency and regression losses, we applied a weight of 0.25 to balance it with the other terms.

6.2 3D Point Estimation

6.2.1 Metrics for 3D Point Maps

We evaluate the quality of our point maps on the dynamic sequence subset of the TUM–RGBD dataset. Following E3D-Bench [40], we use *accuracy*, *completeness*, and *completeness ratio* as metrics.

Let $X_f^i \in \mathbb{R}^{3\times H\times W}$ denote the ground-truth point map for frame i and focal length f, and

$\hat{X}_f^i$ the predicted point map. We define a nearest-neighbor operator

$$\mathrm{KNN}(A, B) : \mathbb{R}^{3\times N} \times \mathbb{R}^{3\times M} \to \mathbb{R}^{3\times N},$$

which assigns to each point in A its closest point in B.

Accuracy. Accuracy measures the mean distance from each predicted point to its closest ground-truth point:

$$\mathrm{Acc} = \frac{1}{|\hat{X}_f^i|} \sum_{x \in \hat{X}_f^i} \left\| x - \mathrm{KNN}(\hat{X}_f^i, X_f^i) \right\|_2 .$$

Low accuracy means predicted points are close to the ground-truth surface. However, accuracy alone can be misleading: a trivial solution where all predictions collapse to a single ground-truth point would yield near-zero accuracy.

Completeness. To complement accuracy, completeness measures the mean distance from each ground-truth point to its closest predicted point:

$$\mathrm{Comp} = \frac{1}{|X_f^i|} \sum_{x \in X_f^i} \left\| x - \mathrm{KNN}(X_f^i, \hat{X}_f^i) \right\|_2 .$$

This metric penalizes missing regions in the reconstruction.

Completeness Ratio. The completeness ratio measures the fraction of ground-truth points that lie within a fixed threshold τ (set to 1 cm) of the predicted surface:

$$\text{Comp. Ratio} = \frac{1}{|X_f^i|} \sum_{x \in X_f^i} \mathbf{1}\Big[\left\| x - \mathrm{KNN}(X_f^i, \hat{X}_f^i) \right\|_2 < \tau \Big].$$

Together, these metrics provide a more complete evaluation: accuracy captures local alignment, completeness penalizes missing geometry, and completeness ratio summarizes coverage quality relative to a fixed tolerance.

Category	Method	Approx. Runtime	ATE↓	RPE_{trans} ↓	RPE_{rot} ↓
w/ intrinsics	DROID-SLAM	<2min	0.175	0.084	1.912
	LeapVO	<2min	0.089	0.066	1.250
w/o intrinsics	CasualSAM	>1h	0.141	0.035	1.712
	AnyCam (<20s)	<20sec	0.095	0.025	1.050
	AnyCam (<2min)	<2min	**0.056**	**0.005**	0.927
	AnyCam-3D (<20s)	<20sec	0.086	0.031	1.010
	AnyCam-3D (<2min)	<2min	0.062	**0.005**	**0.911**

Table 6.1: Evaluation on the TUM-RGBD dynamic sequences. We report Absolute Trajectory Error (ATE), Relative Pose Error translation (RPE_{trans}), and Relative Pose Error rotation (RPE_{rot}). Lower is better.

6.2.2 Camera Pose Estimation

We evaluate the modified AnyCam on its ability to recover camera trajectories in challenging dynamic environments and compare it against state-of-the-art methods. Approaches such as DPVO and LeapVO generally require camera intrinsics at test time, which are often difficult to obtain. CasualSLAM avoids this requirement but instead relies on expensive test-time optimization. In contrast, both AnyCam and our method can produce competitive results without bundle adjustment.

As shown in Table 6.1, our method achieves consistently strong results across all benchmarks. Without test-time refinement, AnyCam-3D obtains slightly better ATE and RPE_{rot} compared to AnyCam, demonstrating that incorporating point maps does not degrade the motion priors learned by AnyCam but instead enhances them. Test-time refinement further mitigates this drift by leveraging longer temporal dependencies across frames, resulting in more stable trajectories.

6.3 Ablation: Loss Formulation

Losses			TUM–RGBD (Dynamics)		
Depth	Consistency	Regression	ACC↓	Comp↓	Comp. ratio↑
✗	✗	✓	0.5210	0.5066	0.01685
✓	✓	✗	**0.5039**	0.5031	0.01894
✗	✓	✓	0.5076	**0.4912**	**0.02873**

Table 6.2: Results on the dynamic sequences of TUM–RGBD. ✓ means the loss was used, ✗ means it was not.

Table 6.2 reports results on the dynamic subset of TUM–RGBD for three different loss configurations. We evaluate using accuracy (ACC↓), completeness (Comp↓), and completeness ratio (Comp. ratio↑).

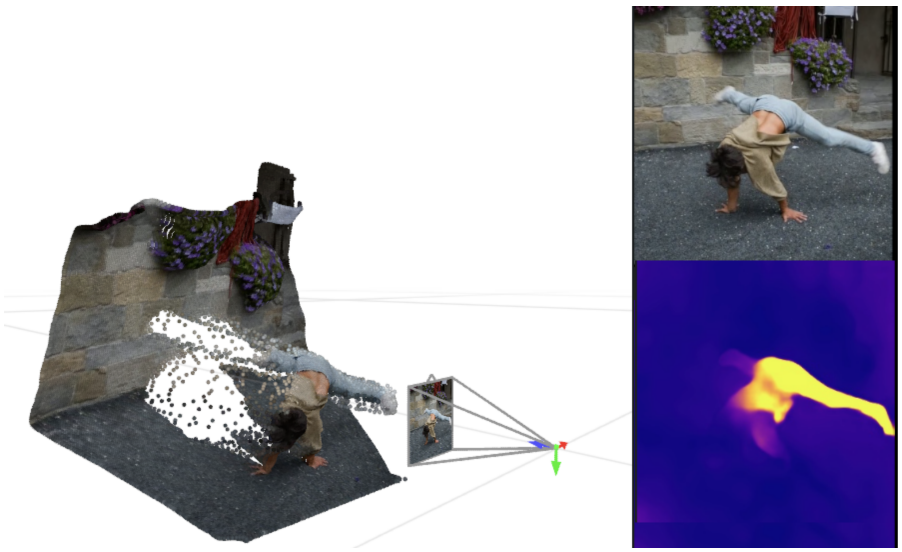

Figure 6.1: Predicted 3D point cloud for a frame from a dynamic sequence in the DAVIS dataset. Left: reconstructed point cloud and camera frustum visualization. Right: input RGB frame (top) and predicted per-pixel uncertainty map (bottom)

The first row corresponds to the regression-only baseline.This setup achieves reasonable performance but produces the worst results overall, with higher completeness errors and the lowest completeness ratio.

Adding the depth and consistency losses (second row) leads to an improvement in accuracy, achieving the lowest ACC among all variants. However, this configuration does not improve completeness or completeness ratio, showing that while depth helps refine local geometry, it is insufficient to improve global scene coverage.

Finally, combining regression and consistency (third row) provides the best overall trade-off. This variant yields the lowest completeness and the highest completeness ratio, demonstrating that consistency across frames is key to reducing drift and improving reconstruction coverage. At the same time, the regression term ensures per-frame accuracy. Together, these two objectives complement each other, leading to the strongest performance across metrics.

6.4 Ablation: DPT Head without Attention

Table 6.3 analyzes the model when we do not use attention in the DPT head for point–map prediction.

Without attention, the ranking of loss configurations remains the same as in the attention setting: the consistency loss is the key factor improving over the regression-only baseline. In this case, combining regression and consistency achieves the best results overall, with ACC 0.47436, Comp 0.48267, and the highest Comp. ratio 0.0309. Notably, these values are better than those obtained when using attention, showing that attention does not provide additional benefit for point map prediction in our setup.

Attention adds little in this setting because the supervision signals are mostly tied to

Losses			DPT head	TUM–RGBD (Dynamics)		
Depth	Consistency	Regression	Attention	ACC↓	Comp↓	Comp. ratio↑
✗	✗	✓	✗	0.5210	0.5066	0.01685
✓	✓	✗	✗	0.4944	0.49829	0.02208
✗	✓	✓	✗	**0.47436**	**0.48267**	**0.0309**

Table 6.3: Ablation without attention in the DPT head. ✓ means the component is used, ✗ means it is not.

per-frame geometry. The regression and depth losses supervise each frame directly, so information from neighboring RGB frames does not provide much extra benefit. The consistency loss, on the other hand, compares the current prediction—transformed with the estimated relative pose—to a geometric target in the next frame that is already defined by intrinsics and depth. Since depth is explicitly provided as input, the network does not need to recover it from RGB cues in other frames. Moreover, the short two-frame training window further limits the potential gains from modeling long-range dependencies.

Under these conditions, attention in the DPT head does not bring clear advantages. A combination of per-frame regression and geometric consistency is sufficient to reduce drift and to improve both accuracy and scene coverage.

6.4.1 Focal length variability and training instability

At the beginning of our experiments we constructed the ground–truth point maps using only the best focal length estimate f_{best}, defined as

$$f_{\text{best}} \;=\; argmax_{f\in\{f_1,\dots,f_m\}} \,\text{softmax}\big(H_{\text{seq}}(I)\big)_f,$$

where H_{seq} is the sequence token head in AnyCam that outputs a distribution over the focal candidates $\{f_1,\dots,f_m\}$. Given f_{best}, a pixel $p=(u,v)$ with depth $d_i(p)$ is unprojected into the point map as

$$X_i(p; f_{\text{best}}) = d_i(p)\, K(f_{\text{best}})^{-1}[u,v,1]^\top.$$

While this produces valid point maps, it introduces a critical issue: the lateral coordinates $X^{x,y}$ scale as d/f_{best}, so variability or error in f_{best} directly changes the geometric scale of the reconstructed points. Since the distribution over predicted focals is typically imbalanced, training gradients concentrate on a narrow subset of focal lengths. As a result, the network learns less robust point maps and struggles when confronted with unseen focal settings at test time.

To address this limitation we follow the original AnyCam design and adopt m=32 focal candidates spanning a broad range of fields of view. Rather than supervising only the single f_{best}, the model distributes likelihoods across all f_k and receives consistent supervision for each candidate. This reduces bias towards particular focal lengths, stabilizes the point–map scale during training, and yields more reliable geometry across sequences.

7 Conclusion

This thesis explored the extension of AnyCam to predict dense 3D point maps in dynamic video sequences. We proposed AnyCam-3D, a modification of the original architecture with a Dense Prediction Transformer (DPT) head and additional supervision strategies, including confidence-aware regression, 3D consistency, and depth losses. Through experiments on dynamic sequences from the TUM-RGBD dataset, we observed that combining regression and consistency terms leads to better reconstruction coverage and accuracy, while depth supervision stabilizes training. Our ablation studies showed that explicit attention in the DPT head does not provide significant gains in this setting, suggesting that geometric constraints and loss design play a larger role in reconstruction quality.

Overall, this work demonstrates that extending feed-forward models with point map prediction is feasible and beneficial for dynamic scene understanding. Future research could investigate integrating longer temporal context and improving handling of focal variability, with the goal of further bridging the gap between feed-forward models and optimization-based pipelines.

List of Figures

List of Tables

Bibliography

[1] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] J. L. Schonberger and J.-M. Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.

[3] C. Smith, D. Charatan, A. Tewari, and V. Sitzmann. "Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent". In: *2025 International Conference on 3D Vision (3DV)*. IEEE. 2025, pp. 389–400.

[4] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. "Dust3r: Geometric 3d vision made easy". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20697–20709.

[5] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *Aaai/iaai* 593598.2 (2002), pp. 593–598.

[6] S. Thrun and M. Montemerlo. "The graph SLAM algorithm with applications to large-scale mapping of urban structures". In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 403–429.

[7] F. Dellaert and M. Kaess. "Square root SAM: Simultaneous localization and mapping via square root information smoothing". In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: A versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. "Kinectfusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE international symposium on mixed and augmented reality*. Ieee. 2011, pp. 127–136.

[10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2017), pp. 1309–1332.

[11] F. Ingrand and M. Ghallab. "Deliberation for autonomous robots: A survey". In: *Artificial Intelligence* 247 (2017), pp. 10–44.

[12] S. Nahavandi, R. Alizadehsani, D. Nahavandi, S. Mohamed, N. Mohajer, M. Rokonuzzaman, and I. Hossain. "A comprehensive review on autonomous navigation". In: *ACM Computing Surveys* 57.9 (2025), pp. 1–67.

[13] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. "Building rome in a day". In: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[14] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[15] R. Collobert and J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[16] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[17] P. Weinzaepfel, V. Leroy, T. Lucas, R. Brégier, Y. Cabon, V. Arora, L. Antsfeld, B. Chidlovskii, G. Csurka, and J. Revaud. "Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 3502–3516.

[18] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, et al. "Habitat-matterport 3d semantics dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4927–4936.

[19] V. Leroy, Y. Cabon, and J. Revaud. "Grounding image matching in 3d with mast3r". In: *European Conference on Computer Vision*. Springer. 2024, pp. 71–91.

[20] A. v. d. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[21] Y. Cabon, L. Stoffl, L. Antsfeld, G. Csurka, B. Chidlovskii, J. Revaud, and V. Leroy. "Must3r: Multi-view network for stereo 3d reconstruction". In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 1050–1060.

[22] 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". In: *arXiv preprint arXiv:2410.03825* (2024).

[23] Z. Li, R. Tucker, F. Cole, Q. Wang, L. Jin, V. Ye, A. Kanazawa, A. Holynski, and N. Snavely. "MegaSaM: Accurate, fast and robust structure and motion from casual dynamic videos". In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 10486–10496.

[24] R. Sutton. *The Bitter Lesson*. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`. Accessed: 2025-08-26. 2019.

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

[26] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. "Vggt: Visual geometry grounded transformer". In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 5294–5306.

[27] R. Ranftl, A. Bochkovskiy, and V. Koltun. "Vision transformers for dense prediction". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 12179–12188.

[28] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. "Cotracker: It is better to track together". In: *European conference on computer vision*. Springer. 2024, pp. 18–35.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[30] F. Wimbauer, W. Chen, D. Muhle, C. Rupprecht, and D. Cremers. "AnyCam: Learning to Recover Camera Poses and Intrinsics from Casual Videos". In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 16717–16727.

[31] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang. "Youtube-vos: A large-scale video object segmentation benchmark". In: *arXiv preprint arXiv:1809.03327* (2018).

[32] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. "Stereo magnification: Learning view synthesis using multiplane images". In: *arXiv preprint arXiv:1805.09817* (2018).

[33] S. Venkataramanan, M. N. Rizve, J. Carreira, Y. M. Asano, and Y. Avrithis. "Is imagenet worth 1 video? learning strong image encoders from 1 long unlabelled video". In: *arXiv preprint arXiv:2310.08584* (2023).

[34] J. Yang, S. Gao, Y. Qiu, L. Chen, T. Li, B. Dai, K. Chitta, P. Wu, J. Zeng, P. Luo, et al. "Generalized predictive model for autonomous driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 14662–14672.

[35] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. "Scaling egocentric vision: The epic-kitchens dataset". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 720–736.

[36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 573–580.

[37] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. Van Gool, and F. Yu. "UniDepth: Universal monocular metric depth estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 10106–10116.

[38] L. Yang, L. Qi, L. Feng, W. Zhang, and Y. Shi. "Revisiting weak-to-strong consistency in semi-supervised semantic segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 7236–7246.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[40] W. Cong, Y. Liang, Y. Zhang, Z. Yang, Y. Wang, B. Ivanovic, M. Pavone, C. Chen, Z. Wang, and Z. Fan. "E3D-Bench: A Benchmark for End-to-End 3D Geometric Foundation Models". In: *arXiv preprint arXiv:2506.01933* (2025).