Enhancing 6D Object Pose Estimation using RGB-D Fusion

Nejla Dinçer, Mustafa Burak Erkoçak, Can Ersoy, Barış Tan Ünal
Politecnico di Torino
{s336887, s338631, s343290, s338210}@studenti.polito.it

Abstract

Estimating the 6D pose of objects is essential for robotics, augmented reality and autonomus driving. While RGB-only methods are computationally lighter, they have problems regarding the scale ambiguity which results in significant depth estimation errors. In this paper, we present the ***Disentangled RGB-D Pose Network****, a pipeline that is a RGB-D fusion architecture that utilizes YOLOv8 for bounding box predictions to detect Region of Interests (ROI) of the RGB image and it's corresponding depth image and decouples pose estimation into two specialized streams: a* ***Geometric Translaion Stream*** *and a* ***Semantic Rotation Stream****. In this way, we benefit from having a light cost computation while utilizing not only RBG information but also depth data as well, reaching satisfactory results. Evaluated on the LineMod dataset, our method demonstrates a substantial improvement over the RGB-only method baseline. Our results and scripts can be found in https://github.com/baristanunal/6DPose*

1. Introduction

For autonomous systems to interact effectively with the physical world, they must accurately determine the 6D pose of objects relative to the camera [7]. This task is conventionally split into estimating a translation vector $t \in \mathbb{R}^3$ and a rotation matrix $R \in SO(3)$. A primary challenge in this field is the **scale ambiguity** inherent in 2D images: without prior depth information or size assumptions, it is difficult to distinguish a small object close to the sensor from a larger object further away.

While recent deep learning approaches have attempted to solve this via direct regression [17] or keypoint matching [11], RGB-based methods often suffer from quadratic error propagation in depth estimation [8]. We propose a two-stage methodology which is the **disentanglement of pose components**, ensuring that geometric scale data and semantic appearance features are processed by specialized networks called **Geometric Translation Stream** and **Semantic Rotation Stream**. The Translation Stream employs a Residual Depth CNN that predicts corrections relative to a geometric median anchor, while the Rotation Stream utilizes a late-fusion MLP to combine ResNet-50 semantic features with geometric embeddings [16].

Figure 1. **Qualitative comparison on the LineMod *Drill*. Left:** The **Pure Pinhole RGB-Only** baseline suffers from scale ambiguity, resulting in inaccurate depth estimation (t_z) and a drifting wireframe. **Right:** Our **Disentangled Residual RGB-D** extension resolves this by anchoring translation to the geometric median, achieving precise alignment with the object surface. *(Visualizations generated using Gemini Nano Banana Pro [6].)*

Our contributions are as follows:

1. **Modular Pipeline:** We implement a two-stream architecture that isolates ROI detection using YOLO, followed by decoupled estimation of translation and rotation.
2. **Residual Depth Learning:** Rather than regressing absolute distance, our Translation Stream predicts a scalar residual (δ_z) on top of a "geometric anchor" derived from the median depth values (z_{med}).
3. **Late Fusion for Rotation:** We utilize a multi-modal architecture that fuses semantic features of the RBG image by a ResNet-50 backbone with geometric embeddings to regress rotation in a continuous 6D space [19].

Our experimental results on the LineMod dataset confirm that hybridizing geometric heuristics with deep residual learning effectively resolves scale ambiguity, significantly outperforming traditional RGB-only Pinhole projections, which acts as a comparison baseline for our methodology.

2. Related Work

Pose from RGB images. Classical methods rely on detecting and matching keypoints with known object models [1, 3, 4, 12, 20]. Newer methods address the challenge by learning to predict the 2D keypoints [2, 10, 13–15] and solve the poses with PnP [5].As speed demanding tasks become more prevalent, these methods become unreliable in low-texture or low-resolution inputs. Additionally, these methods do not perform as well as the ones using depth information to estimate object poses.

Pose estimation from RGB-D data. PoseCNN estimates the 3D translation of an object by localizing its center in the image and predicting its distance from the camera [17]. Similar to our method, PoseCNN leverages the pinhole camera translation model. However, they used a single CNN for translation and rotation estimation compared to our Depth Z predictor MLP. SSD-6D [8] uses 2D bounding boxes and classifies the object's rotation into discrete viewpoint bins, while predicting 3D translation using the size and position of the bounding box relative to the 3D model. Both of their initial predictions are not precise enough, therefore they use some post-processing steps to enhance the accuracy such as ICP.

Pixel-wise Voting. To solve the occlusion problem, recent works used dense pixel-wise prediction. Pix2Pose [9] predicts 3D coordinates and expected errors per pixel. These predictions are then used to perform 2D-3D translation to directly compute poses with the PnP algorithm. PVNet [11] further improved this by using a RANSAC-based voting scheme, where individual pixels vote for the location of object keypoints.

RGB-D Sensor Fusion. The integration of depth addresses the scale ambiguity problem **Global vs. Dense Fusion:** Pointfusion [18] uses a CNN to extract a fixed-size feature vector and fuse by directly concatenating the image features with the geometry features. It is one of the first methods to combine depth and RGB data, but has a major weakness due to global pooling which leads important geometric details to disappear in order to make an accurate prediction. **Dense Approaches:** To address this problem, DenseFusion [16] proposes a pixel-level fusion. It makes a single prediction using the global feature vector instead of performing per-point prediction. Our work is inspired by DenseFusion. We separated translation and rotation estimation into different branches, replacing computationally expensive PointNet branch with a residual CNN architecture that operates on depth maps directly.

3. Dataset

We evaluated our method on a subset of the LineMod dataset, a standard benchmark for 6D pose estimation consisting of 13 low-textured objects recorded in video sequences. Each object is annotated with ground truth 6D pose (rotation and translation), 2D bounding boxes, and aligned depth images. To handle geometric symmetries in objects such as "Eggbox" and "Glue", we account for equivalent poses during evaluation using the ADD-S metric. Data preprocessing involves cropping and resizing object regions from both RGB and depth images using 2D bounding boxes, utilizing ground truth boxes for training and YOLOv8 predictions for inference. Furthermore, depth values are converted from millimeters to meters to facilitate neural network convergence. During our analysis, we realized that in "Benchvise" object images, there are multiple annotations for each object in the corresponding image sowe had to apply a filter to only selecting the annotations of the "Benchvise" object.

4. Methodology

We address the problem of rigid 6D object pose estimation from a single RGB-D image. Formally, given an RGB image $I \in \mathbb{R}^{H\times W\times 3}$ and a corresponding spatially aligned depth map $D \in \mathbb{R}^{H\times W}$, our goal is to estimate the rigid transformation $\mathbf{P} = [\mathbf{R}|\mathbf{t}] \in SE(3)$ that aligns the object's local coordinate system to the camera coordinate system. The pose is parameterized by a rotation matrix $\mathbf{R} \in SO(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$.

We assume a pinhole camera model characterized by a known intrinsic calibration matrix $\mathbf{K} \in \mathbb{R}^{3\times 3}$. Let $\mathbf{x}_o = [x_o, y_o, z_o]^T$ denote a 3D point defined in the object's model space. Its transformation into the camera coordinate frame is given by $\mathbf{x}_c = \mathbf{R}\mathbf{x}_o + \mathbf{t}$. The relationship between this 3D point and its homogeneous 2D projection $\tilde{\mathbf{u}} = [u, v, 1]^T$ in the image plane is defined by the projective equation.

$$s\tilde{\mathbf{u}} = \mathbf{K}(\mathbf{R}\mathbf{x}_o + \mathbf{t}) \tag{1}$$

where s represents the scale factor corresponding to the depth of the point along the optical axis (z_c). Unlike RGB-only approaches where scale ambiguity makes recovering $\mathbf{t}$ ill-posed, our method leverages the dense depth map D to explicitly constrain the geometric translation.

4.1. Architecture Overview

Our proposed method, the **Disentangled RGB-D Pose Network**, diverges from traditional architectures that treat 6D pose estimation as a monolithic regression task. Although pixel-wise dense fusion methods [16] achieve high accuracy, they incur significant computational overhead by processing feature embeddings for every pixel within the object mask. Furthermore, global regression approaches often suffer from optimization conflicts, as the gradients derived from translation errors (which depend on scale and geometry) can dominate or interfere with rotation learning (which depends on semantic features).

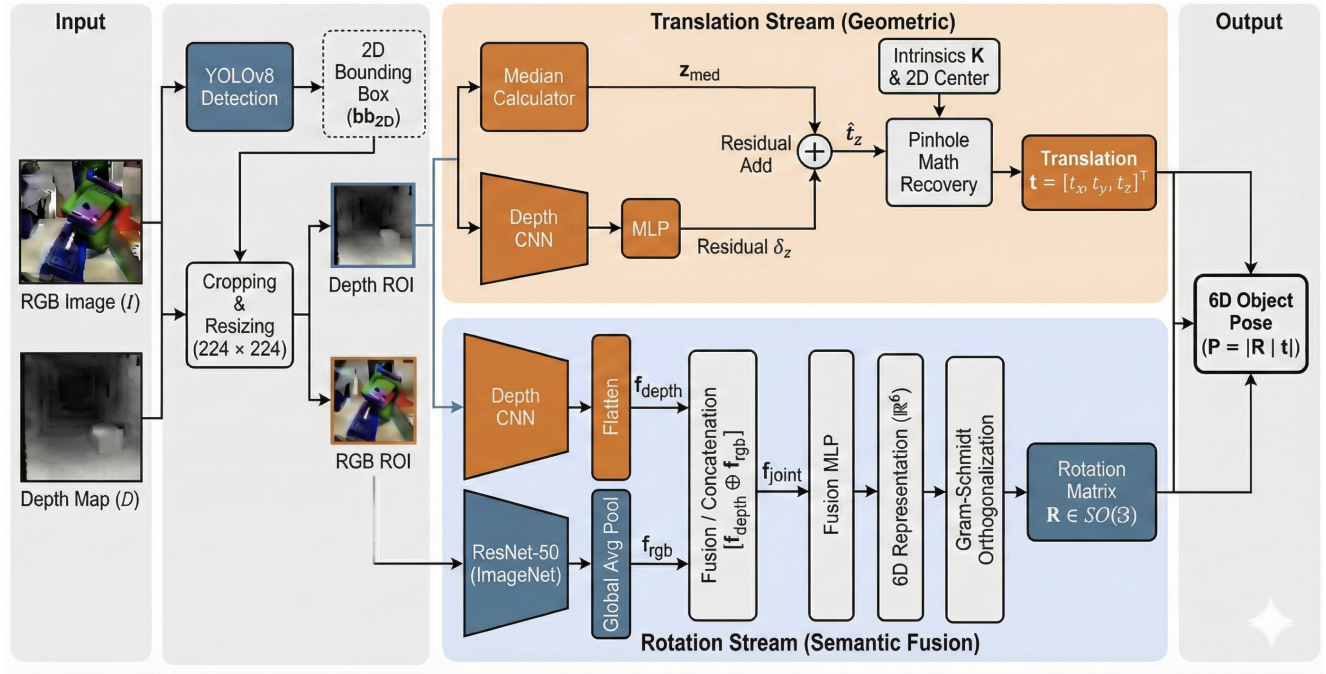

Figure 2. **Architecture of the Disentangled RGB-D Pose Network.** After isolating the ROI with YOLOv8, the pipeline splits into a *Geometric Stream* (top) that refines depth to recover translation $\mathbf{t}$, and a *Feature Fusion Stream* (bottom) that merges ResNet-50 and geometric features to regress rotation $\mathbf{R}$. *(Diagram generated using Gemini Nano Banana Pro [6].)*

To address these limitations, we propose a two-stream design philosophy (illustrated in Figure 2) that decouples the problem into three sequential stages:

1. **ROI Localization:** A lightweight 2D detector identifies the object of interest, isolating the relevant spatial region.
2. **Geometric Stream (Translation):** A specialized deep-residual network explicitly recovers the translation vector $\mathbf{t}$ by using the metric information on the depth map.
3. **Feature Fusion Stream (Rotation):** A parallel branch processes RGB and Depth features through late fusion to regress the orientation $\mathbf{R}$ in a continuous rotation space.

This disentanglement ensures that geometric ambiguity does not corrupt semantic feature learning, allowing each stream to specialize in its respective domain.

4.2. 2D Object Detection (ROI Extraction)

The first stage of our pipeline isolates the target object from the background clutter. We employ **YOLOv8-Nano**, a state-of-the-art single-stage detector known for its efficiency and real-time performance.

4.2.1. Data Preparation and Training

To adapt the detector to the LineMod dataset, we processed the ground truth annotations provided in the `gt.yml` files. The 2D bounding boxes were converted into the standard YOLO format (class, center_x, center_y, width, height), with coordinates normalized to the range $[0, 1]$ relative to the image dimensions (W, H). The dataset was split into training (60%), validation (20%), and testing (20%) subsets.

The network was fine-tuned for 50 epochs with an input resolution of 640×640 pixels. For training, we employed the standard Ultralytics YOLOv8-Nano hyperparameter configuration, which automatically applies augmentations such as mosaic, random scaling, and color jitter to enhance generalization.

4.2.2. Inference and ROI Cropping

During inference, the detector outputs a bounding box $\mathbf{bb}_{2D} = [u, v, w, h]$ for the target object class c. This bounding box serves as a hard attention mechanism. We use $\mathbf{bb}_{2D}$ to crop both the RGB image I and the Depth map D, resizing the resulting regions of interest (ROIs) to a fixed spatial resolution (224×224) before feeding them into the pose estimation streams.

4.3. Translation Stream: Residual Depth Prediction

Accurate estimation of the translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$ is critical for 6D pose estimation. In particular, the depth component t_z dictates the metric scale of the object; errors in t_z propagate directly to the 3D alignment, rendering even perfect rotation estimates useless. To address this, we design the Translation Stream (Figure 2, top

branch) as a geometry-first module that decouples depth regression from planar positioning.

4.3.1. Residual Learning of T_z

Directly regressing the absolute depth t_z from an image crop is optimizationally unstable due to the large variance in potential object distances (ranging from centimeters to meters) and the scale ambiguity inherent in perspective projection. However, the input depth map D provides a strong, albeit noisy, prior.

We propose a residual learning formulation anchored by the local geometry. Let D_{ROI} denote the masked depth values within the detected object's bounding box. We calculate a "geometric anchor" z_{med} as the median of these values:

$$z_{med} = \text{median}(D_{ROI}) \tag{2}$$

The median is selected over the mean to ensure robustness against segmentation outliers and background noise often present at the object boundaries.

The network learns a scalar residual correction term, δ_z, rather than the absolute value. The final depth estimate is formulated as:

$$\hat{t}_z = z_{med} + \mathcal{F}_{depth}(D_{ROI}) \tag{3}$$

where $\mathcal{F}_{depth}$ is our specialized Depth-CNN. As implemented in our `DepthZPredictor` module, this network consists of four sequential convolutional blocks. Each block comprises a 3×3 Convolution, Batch Normalization, ReLU activation, and a 2×2 Max Pooling layer. The channel dimensions increase progressively ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$) to capture hierarchical geometric structures. The resulting feature map is flattened and processed by a Multi-Layer Perceptron (MLP) to regress the scalar offset δ_z.

4.3.2. Analytical Recovery of T_{xy}

A common pitfall in pose estimation networks is the independent regression of t_x and t_y, which ignores the fixed projective constraints of the calibrated camera. If a network predicts a center (t_x, t_y) that does not project back to the center of the detection box, the pose becomes optically inconsistent.

To guarantee consistency between the 2D detection and the 3D pose, we do **not** learn t_x and t_y via neural regression. Instead, we analytically recover them using the inverse pinhole projection. Given the estimated depth $\hat{t}_z$ and the center of the 2D bounding box (c_u, c_v) in pixel coordinates, the 3D planar coordinates are derived as:

$$\hat{t}_x = \frac{(c_u - u_0) \cdot \hat{t}_z}{f_x}, \quad \hat{t}_y = \frac{(c_v - v_0) \cdot \hat{t}_z}{f_y} \tag{4}$$

where f_x, f_y are the focal lengths and (u_0, v_0) is the principal point from the camera intrinsic matrix $\mathbf{K}$. This formulation mathematically guarantees that the estimated 3D centroid projects exactly to the center of the detected Region of Interest (ROI), effectively eliminating the "drift" often observed in direct regression methods.

4.4. Rotation Stream: RGB-D Late Fusion

While the Translation Stream relies primarily on geometric depth consistency, the Rotation Stream (Figure 2, bottom branch) requires a rich semantic understanding of the object's appearance. Determining orientation often hinges on identifying specific texture markers, asymmetric details, or color patterns that are indistinguishable in a raw depth map. Consequently, we employ a multi-modal *Late Fusion* architecture that processes RGB and Depth information in parallel before fusing them in a high-level latent space.

4.4.1. Feature Extraction

The stream consists of two dedicated feature encoders:

RGB Semantic Branch: To capture high-level visual semantics, we utilize a ResNet-50 backbone initialized with weights pretrained on ImageNet. The network processes the 224×224 RGB crop and extracts a dense feature representation. We discard the final classification layer and utilize the output of the global average pooling layer, resulting in a semantic vector $\mathbf{f}_{rgb} \in \mathbb{R}^{2048}$. This branch is essential for resolving symmetries and identifying class-specific orientation cues.

Depth Geometric Branch: Complementary to the RGB features, we employ a lightweight CNN (structurally identical to the feature extractor in Section 4.3) to process the normalized depth crop. This branch encodes local surface curvature and geometric discontinuities into a compact embedding $\mathbf{f}_{depth} \in \mathbb{R}^{128}$.

4.4.2. Late Fusion and 6D Representation

Unlike DenseFusion [16], which relies on pixel-wise feature fusion, we implement a computationally efficient *Late Fusion* strategy. The semantic and geometric feature vectors are concatenated into a joint embedding $\mathbf{f}_{joint} = [\mathbf{f}_{rgb} \oplus \mathbf{f}_{depth}] \in \mathbb{R}^{2176}$. This vector is processed by a Fusion MLP to regress the pose parameters.

For the rotation output, we adopt the continuous 6D representation proposed by Zhou et al. [19] to circumvent the discontinuities and training instability associated with unit quaternions. The network predicts a raw vector $\mathbf{r} = [\mathbf{a}_1, \mathbf{a}_2]^T \in \mathbb{R}^6$. We recover the orthonormal rotation matrix $\mathbf{R} \in SO(3)$ via a differentiable Gram-Schmidt process, where $\mathbf{a}_1$ is normalized to form the first column, $\mathbf{a}_2$ is orthogonalized to form the second, and the third column is their cross-product. This representation ensures a continuous optimization landscape, leading to faster convergence.

4.5. Loss Functions

To simplify the optimization landscape and avoid gradient conflicts between the geometric and semantic tasks, we employ a decoupled training strategy. Instead of utilizing

a joint multi-task objective, which often requires sensitive hyperparameter tuning to balance gradient magnitudes, we train the Translation Stream and the Rotation Stream independently. This isolation ensures that the geometric regression of depth is not affected by the semantic learning required for orientation.

4.5.1. Translation Loss

The Translation Stream is trained to minimize the error in depth estimation. We utilize the L_1 loss (Mean Absolute Error) rather than the Mean Squared Error (L_2), as the L_1 norm is more robust to outliers arising from depth sensor noise and segmentation inaccuracies at the object boundary. The loss is defined on the depth component t_z:

$$\mathcal{L}_{trans} = ||\hat{t}_z - t_z^{gt}||_1 \tag{5}$$

Since the planar coordinates $\hat{t}_x$ and $\hat{t}_y$ are analytically derived from $\hat{t}_z$ via the fixed intrinsic matrix, minimizing the error in t_z implicitly minimizes the re-projection error of the object centroid.

4.5.2. Rotation Loss

The Rotation Stream is trained to minimize the angular difference between the predicted orientation and the ground truth. Unlike the Frobenius norm, which operates in the Euclidean space of matrix coefficients, we employ the **Geodesic Loss**, which measures the magnitude of the rotation angle required to align the predicted frame $\mathbf{R}$ with the ground truth frame $\mathbf{R}_{gt}$ in $SO(3)$. The loss is formulated as:

$$\mathcal{L}_{rot} = \arccos\left(\frac{\text{Tr}(\mathbf{R}\mathbf{R}_{gt}^T) - 1}{2}\right) \tag{6}$$

where $\text{Tr}(\cdot)$ denotes the matrix trace. This function directly penalizes the angular error in radians, ensuring that the optimization objective aligns perfectly with the evaluation metric. To prevent numerical instability during backpropagation, the argument of the arccos function is clamped to the range $[1-\epsilon, 1+\epsilon]$.

5. Experiments

In the experimental section, we would like to answer the following questions: (1) How much does the integration of depth information improve performance over the RGB-only baseline? (2) Does decoupling the translation and rotation streams improve convergence and accuracy compared to joint regression? (3) Can a specialized CNN effectively refine depth by predicting a residual δ_z relative to a geometric anchor?

5.1. Evaluation Metrics

To evaluate our Disentangled RGB-D Pose Network, we adopt the standard metrics used in 6D pose estimation benchmarks. Following the approach used in prior works, we employ different metrics for symmetric and non-symmetric objects to account for geometric attributes.

Average Distance Differentiable (ADD [7]): The primary metric for non-symmetric objects is ADD, which calculates the mean Euclidean distance between 3D model vertices transformed by the ground truth pose $[\mathbf{R}|\mathbf{t}]$ and those transformed by the predicted pose $[\hat{\mathbf{R}}|\hat{\mathbf{t}}]$.

For symmetric objects in our dataset (Eggbox and Glue), we calculate ADD-S, which is an ambiguity-invariant error metric that measures the distance to the closest point on the ground truth model rather than a fixed point. We calculate the mean error in centimeters for both ADD and ADD-S, as well as a combined metric ADD(S) that applies the appropriate method for each object class. Additionally, ADDR (cm), which is the rotation component of the ADD metric, is also presented for each object. On top of these, the following are also reported: **Accuracy@10%:** Percentage of test samples where the error ADD(S) is less than 10% of the object's diameter ($0.1d$). **ADDR@10%:** Percentage of test samples where the rotation-induced error (ADDR) is less than 10% of the object's diameter. **Area Under Curve (AUC):** Area under the curve for the ADD-S error. Following prior work, the maximum threshold of AUC is set to 0.1m.

5.2. Implementation Details

All models were implemented using PyTorch constructs and trained on Google Colab using a T4 GPU. We trained our models in both streams (Residual Depth Prediction and RGB-D Late Fusion) for 50 epochs using the Adam optimizer with an initial learning rate of 1×10^{-4} and weight decay. A `ReduceLROnPlateau` scheduler was employed to decay the learning rate upon loss stagnation, where the validation loss stopped improving for multiple epochs. We utilized a batch size of 64 for the relatively lightweight translation stream and 32 for the rotation stream.

5.3. Ablation Studies & Model Evolution

In the following section, we present the different approaches we had before reaching our final methodology and how we evolved our model from a RGB-Only baseline, along with the different architectures we evaluated.

5.3.1. The Monocular Baseline

To establish a lower bound performance and isolate the contribution of depth information, our first evolution was the implementation of a monocular RGB-only baseline. This iteration relies exclusively on the 2D detections from the YOLOv8 model and standard pinhole camera model constraints that relate 2D pixel projections to 3D space, without the utilization of depth information.

The translation along the optical axis, t_z, was inferred from the scale of the object's projection on the image plane.

We employed a standard pinhole procedure that assumes the diagonal of the 2D bounding box corresponds to the physical diameter of the object.

Let D_{real} be the known physical diameter of the object class (in millimeters). From the predicted bounding box $\mathbf{bb}_{2D} = [u, v, w, h]$ that the YOLO model inferred, we extracted the width w and height h to approximate the pixel-space diagonal d_{px}. We then derived the depth t_z using the camera's focal length f:

$$t_z \approx \frac{f_{avg} \cdot D_{real}}{\sqrt{w^2 + h^2}} \tag{7}$$

Once t_z is estimated, the coordinates t_x and t_y are recovered via the inverse projection equations defined in Eq. 4. For rotation, we utilized a standalone **ResNet-50** backbone initialized with ImageNet weights. The network was modified by replacing the final classification head with a regression MLP that outputs a 4-dimensional unit quaternion $\mathbf{q} \in \mathbf{R}^4$. This quaternion represents the 3D orientation of the object.

Fundamental issue of this method is that Pinhole calculation assumes that the 2D bounding box tightly encloses the object's maximum dimension and it fails because the 2D projection of an irregular object changes drastically with rotation (e.g. a drill viewed from the side vs. the front). The diagonal of the bounding box (d_{px}) does not consistently represent the object's true physical diameter (D_{real}) because the object's apparent size changes significantly depending on the angle from which it is viewed. This causes errors in finding the t_z value.

Since t_x and t_y are derived mathematically from t_z, errors in depth estimation propagate to the lateral position. An error in t_z displaces the estimated centroid in 3D space and results in high translation errors. This confirms the necessity for the usage of depth information for improvement.

5.3.2. Joint RGBD Late Fusion

After implementing the baseline approach, we found out incorporating depth information in the training process is necessary. We built a new CNN (**RGBDFusionNet**). We used **ResNet-50** backbone initialized with ImageNet weights and froze the early layers.

Processing of geometric information is handled by a depth encoder CNN that predicts our t_z value. RGBDFusionNet, which has 2 separate regression heads, first retrieves RGB features from RGB encoder and geometric features from depth encoder. Then these information are fused in a convolutinal layer outputting a combined feature vector. This vector is flattened and metadata (normalized values of bounding box coordinates x, y, w, h and camera instrincts f_x, f_y, c_x, c_y) is injected into it, which is later used for rotation prediction in the quaternion form. Lastly, the translation head predicts the value of t_z and then t_x and t_y are calculated from the predicted t_z (using the Pinhole Camera Eq. 4).

We utilized a joint loss function that evaluates both translation and rotation loss at the same time. The ratio of the weights of the translation and rotation loss is 15 to 1. $Weight_{\text{rot}}$ is much smaller because $Loss_{\text{trans}}$ is in meters where $Loss_{\text{rot}}$ is a unitless similarity score. Also, t_x and t_y are dependent on t_z prediction so if t_z is wrong, this error propagates to t_x and t_y, which yields translation to have a greater importance than rotation. L1 loss is used for $Loss_{\text{trans}}$ and cosine similarity is used for $Loss_{\text{rot}}$.

Overall, this method performed better than the baseline model however still was not good enough in terms of translation and rotation accuracy as it resulted in a X% ADD-Accuracy@10 (see Tab. 4 table and loss graphs in Fig. 3). This is mainly due to the loss function which evaluates both translation and rotation loss together. Later, we discovered separating the depth and RGB branches would give more accurate results.

5.3.3. Validation of Analytical Recovery

To assess the decision of mathematically deriving coordinates t_x and t_y (using Eq. 5), we conducted an "Oracle" experiment using ground truth depth (t_z^{gt}) and ground truth bounding box centers. This test helped us to isolate the geometric error relying on the projection assumption that is implied by the inverse projection approach.

The experiment yielded a mean Euclidean error of less than 1 cm across the dataset, with compact and regular objects exhibiting errors below or around 5 mm. These negligible values confirm that the analytical back-projection model is mathematically robust. More importantly, it demonstrates that translation error is predominantly a function of depth (t_z) accuracy. Such finding justifies our architectural strategy to isolate and optimize z-axis regression as the main improvement design of pipeline performance.

5.3.4. Joint vs. Disentangled

The *RGBDFusionNet* introduced in a previous evolution incorporated depth data, yet its architecture relied on a shared latent space in which fused RGB-Depth features were used to regress both translation and rotation simultaneously. Results obtained from this coupled design revealed a critical optimization issue: The gradients derived from scalar depth errors frequently conflicted with the complex, high-dimensional feature learning required for orientation. This happens as the model attempts to optimize a single combined loss function. Such "entanglement" between objectives resulted in unstable convergence. We can attribute this to the network struggling to balance two separate tasks within a single embedding.

Additionally from the previous "Oracle" experiment, which proved that translation accuracy was dominantly affected by the capability of our model's geometric depth pre-

Table 1. Quantitative comparison between the Monocular Baseline and the Main Extension. The metrics reported are ADD(S) error (cm), Accuracy@10% ($< 0.1d$), Area Under Curve (AUC), Translation Error (cm), and Rotation Error ($^\circ$).

	RGB-Only (Baseline)					**Disentangled RGB-D Pose Network (Extension)**				
	ADD(S) (cm)	**Acc@10%**	**AUC**	**Tr (cm)**	**Rot ($^\circ$)**	**ADD(S) (cm)**	**Acc@10%**	**AUC**	**Tr (cm)**	**Rot ($^\circ$)**
Ape	10.72	0.43	16.97	10.64	19.72	0.85	76.09	91.53	0.79	5.41
Benchvise	9.48	1.63	23.97	9.31	17.95	1.71	73.98	82.85	1.63	4.86
Camera	14.08	0.00	3.60	14.01	17.55	1.55	61.79	84.45	1.48	4.46
Can	13.77	6.40	13.32	13.67	15.73	1.05	96.80	89.47	0.95	4.51
Cat	7.94	2.93	30.99	7.85	17.05	1.08	90.79	89.17	1.03	4.86
Driller	9.59	5.11	23.74	9.21	17.44	1.85	82.13	81.46	1.75	4.59
Duck	15.34	0.00	0.58	15.35	22.65	0.98	67.80	90.19	0.94	5.30
Eggbox	7.57	6.30	34.64	11.91	17.83	0.32	100.00	96.79	0.58	4.85
Glue	10.01	22.31	48.99	13.85	19.87	0.53	100.00	94.65	1.04	5.19
Holepuncher	11.61	0.00	14.68	11.53	19.26	1.03	76.36	89.72	0.96	5.22
Iron	7.29	4.60	37.08	6.93	17.46	2.30	74.48	77.02	2.22	4.78
Lamp	8.29	9.66	29.57	7.97	17.50	1.69	88.66	83.13	1.56	4.65
Phone	9.26	2.98	24.13	9.18	20.68	1.76	70.64	82.43	1.70	5.06
MEAN	**10.39**	**4.91**	**23.25**	**10.93**	**18.55**	**1.27**	**81.80**	**87.14**	**1.27**	**4.91**

diction, we designed a Disentangled architecture. Unlike the fused approach, we isolated the translation task to a dedicated stream that processes depth information exclusively. This separation ensured that the model defined in the translation stream converged efficiently to a precise, focused metric solution without its gradients being corrupted by the representations and learning process in the rotation stream. Likewise, the rotation stream also benefitted from the same improvements. The results showed that with such an architecture, we were able to increase the final pose accuracy. This initial disentangled structure serves as the foundation for our final proposed method, as detailed in the Methodology (Section 4).

5.3.5. Target Representation

In the final stage of our design evolution, we addressed the performance of the depth regression task within the Translation Stream. We compared the two approaches: Directly regressing the absolute depth value (t_z) versus learning a residual offset (δ_z) relative to already known depth average.

Direct regression of absolute depth was relatively inefficient due to the high variance in object distances and the scale ambiguity inherent to perspective projection. To mitigate this, we adopted a Residual Learning formulation. By calculating the median depth of the masked ROI (z_{med}) prior to learning, the network is relieved of the task of estimating the object's entire distance from scratch. Instead, it focuses solely on learning the local correction δ_z required to shift the centroid from the surface median to the true object center. This reformulation significantly simplified the optimization, as the Depth CNN/MLP deals with smaller values more efficiently, and minimized variance. As a result of this last iteration, we obtained the depth estimation model formally defined in Eq. 3 of our Methodology.

5.4. Quantitative Results

The quantitative analysis is structured to validate two central hypotheses: first, that decoupling depth regression from rotation estimation significantly reduces optimization interference; and second, that residual learning on geometric priors is superior to absolute coordinate regression.

5.4.1. Overall Performance and Ablation Gains

The global performance gains achieved by our method are summarized in Table 1, which presents a head-to-head comparison between the Monocular Baseline and our final Extension (main Methodology).

The quantitative leap is significant. The baseline implementation, which relied on RGB-based pinhole projection, yielded a mean accuracy (Acc@10%) of only 4.91%, with a prohibitive average translation error of 10.93 cm, which according to our previous inferences is the primary factor of the error. This high error margin confirms our initial assumption: inferring depth solely from 2D bounding box ratios is insufficient for 6D pose tasks due to the scale ambiguity inherent in perspective projection.

In contrast, our proposed Extension achieves a mean accuracy of 81.80%, which corresponds to an improvement of 16 times over the baseline. More critically, the translation error (t_{error}) was reduced from 10.93 cm to 1.27 cm, while the rotation error (R_{error}) dropped from 18.55° to 4.91°.

It is important to note that this improvement in rotation accuracy is not merely a result of better visual features, but a direct consequence of our disentangled architecture. By referring to the detailed breakdowns in the Appendix (specifically comparing Table 4 and Table 5), we observe that the Joint RGBD Late Fusion architecture achieved a translation error of 1.63 cm but retained a high rotation er-

ror of $18.38°$. This indicates that when translation and rotation share a common loss function (as in the Joint model), the high gradients generated by geometric depth corrections destabilize the learning of orientation features. By separating these streams in our proposed method, we allowed the rotation branch to converge to a precise $4.91°$ error without interference.

5.4.2. Comparative Analysis with State-of-the-Art

Table 2 compares our progressive results against the DenseFusion[16] baseline, comparing our method against both their per-pixel and iterative refinement variants.

Our approach achieves a mean accuracy of 81.80%, which is highly competitive with the DenseFusion (Per-pixel) baseline of 86.2%, particularly given that our architecture utilizes a significantly more computationally efficient depth stream. By shifting the learning objective to regress a simple local offset (δ_z) rather than generating high-dimensional embeddings for every single pixel, we drastically reduce the model complexity while still maintaining satisfactory translation accuracy. For symmetric objects, our method surpasses the state-of-the-art benchmarks. That is, we achieve 100% accuracy on the *Eggbox* and *Glue* objects, and 96.80% on the *Can*, surpassing the DenseFusion (Per-pixel) score of 86.6% for the latter.

The 4.4% performance gap between our method (81.80%) and the DenseFusion Per-pixel baseline (86.2%) could be attributed to the structural distinction between global regression and dense voting. DenseFusion aggregates pose estimates from thousands of individual pixels, a consensus mechanism that inherently suppresses noise. In contrast, our translation stream compresses the entire Region of Interest (ROI) into a single vector for scalar regression. While this high-level feature compression enables faster inference, it naturally introduces a slight reduction in granularity compared to the dense, multi-point generation of the SoTA method.

However, the trade-off could be justified by computational efficiency and stability. While DenseFusion (Iterative) pushes the mean accuracy to 94.3% through an intense post-processing, our method delivers a satisfactory precision ($<$ 1.3 cm translation error) in a single forward pass, without the need for iterative refinement steps.

5.5. Qualitative Results

Qualitative results of per-object analysis (that we can deduce from Tables 3, 4, and 5) confirm our proposed method performs competitively across diverse object topologies. It achieves near-perfect predictions for objects with consistent and/or symmetric geometries such as the *Eggbox*, *Glue*, and *Phone*. For these objects, the predicted YOLO bounding boxes tightly cover the physical object, which then allows the Residual MLP to successfully minimize the variance in depth estimation (δ_z) by using the uniform surface depth of the Region of Interest. These results go in parallel with the quantitative data, where these type of classes showed the lowest translation errors, suggesting that our architecture effectively resolves the scale ambiguity when visual features are relatively consistent.

Table 2. Comparison of accuracy (Acc@10% / ADD(S)-0.1d)

	Baseline (Mono)	**Joint RGBD**	**Extension**	**DF (Per-pix)**	**DF (Iterative)**
Ape	0.43	18.26	76.09	79.5	92.3
Benchvise	1.63	45.12	73.98	84.2	93.2
Camera	0.00	25.47	61.79	76.5	94.4
Can	6.40	43.60	96.80	86.6	93.1
Cat	2.93	46.03	90.79	88.8	96.5
Driller	5.11	38.72	82.13	77.7	87.0
Duck	0.00	11.36	67.80	76.3	92.3
Eggbox	6.30	100.00	100.00	99.9	99.8
Glue	22.31	98.85	100.00	99.4	100.0
Holepuncher	0.00	30.23	76.36	79.0	92.1
Iron	4.60	41.42	74.48	92.1	97.0
Lamp	9.66	66.39	88.66	92.3	95.3
Phone	2.98	36.17	70.64	88.0	92.8
MEAN	**4.91**	**46.77**	**81.80**	**86.2**	**94.3**

However, results also indicate the limitations of our method when dealing with complex or highly compact geometries. For objects with significant irregular geometries, such as the *Iron* and *Driller*, slight spatial drifts in the bounding box coverage occur. This can be attributed to the variation in depth values across the object's surface; a minor misalignment in the 2D ROI cropping can shift the median depth (z_{med}) away from the true centroid, forcing the network to predict a larger, more difficult residual offset. Similarly, for small objects like the *Ape* and *Duck*, while translation remains accurate, minor rotational discrepancies are visible. This suggests that the spatial resolution of the cropped RGB features may be insufficient for precise orientation regression on compact targets.

6. Conclusion

We have presented a new approach for 6D pose estimation of objects from their RGB-D images. Our results confirm that while RGB-only approaches provide a computationally light baseline, they are insufficient for high-precision tasks. Our two-stream disentanglement approach provides a better estimation of both translation and rotation for objects in the LineMod dataset. Future work could include leveraging segmentation masks to isolate the object from the background and calculate average depth could further refine T_z estimation, a strategy not explored in this study due to computational and time constraints.

References

[1] Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3d chairs: Exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 3762–3769, 2014. 2

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision (ECCV)*, pages 536–551. Springer, 2014. 2

[3] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011. 2

[4] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188, 2006. 2

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[6] Google DeepMind. Gemini Nano Banana Pro: High-fidelity ai image generation model. https://deepmind.google/gemini, 2025. Accessed: 2026-01-29. 1, 3

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, 2012. 1, 5

[8] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[9] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[10] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. *arXiv preprint arXiv:1703.04670*, 2017. 2

[11] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[12] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66 (3):231–259, 2006. 2

[13] Supasorn Suwajanakorn, Noah Snavely, Jonathan Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *arXiv preprint arXiv:1807.03146*, 2018. 2

[14] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 2

[16] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 4, 8

[17] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018. 1, 2

[18] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[19] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019. 1, 4

[20] Menglong Zhu, Konstantinos G Derpanis, Y Yang, Samarth Brahmbhatt, M Zhang, C Phillips, M Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, 2014. 2

Enhancing 6D Object Pose Estimation using RGB-D Fusion

Supplementary Material

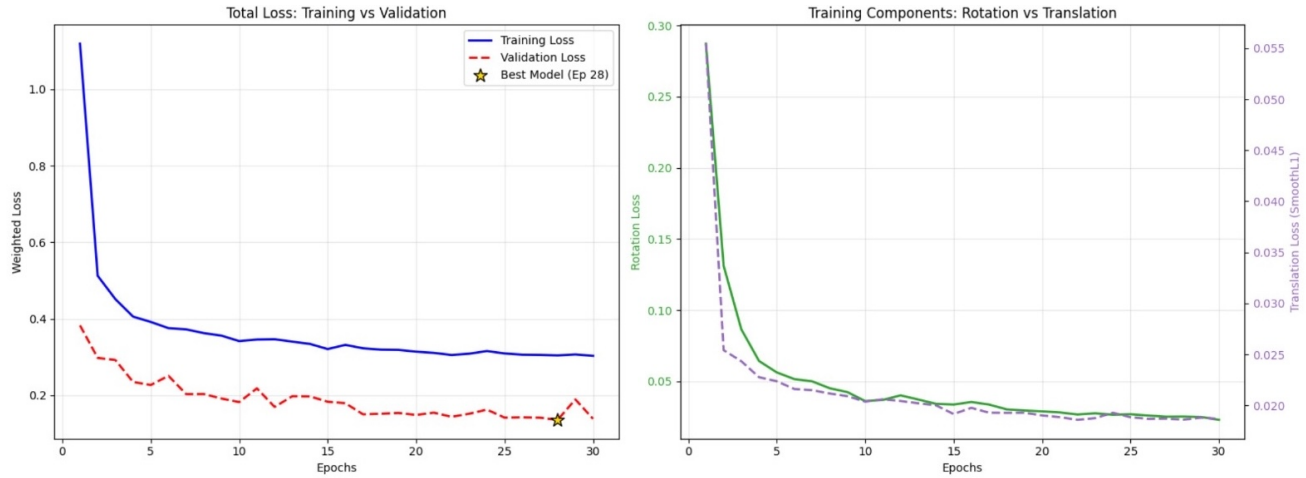

Figure 3. Overall training loss convergence for the RGB-D Fusion Network. The graph demonstrates the steady minimization of the combined pose loss function over 50 epochs.

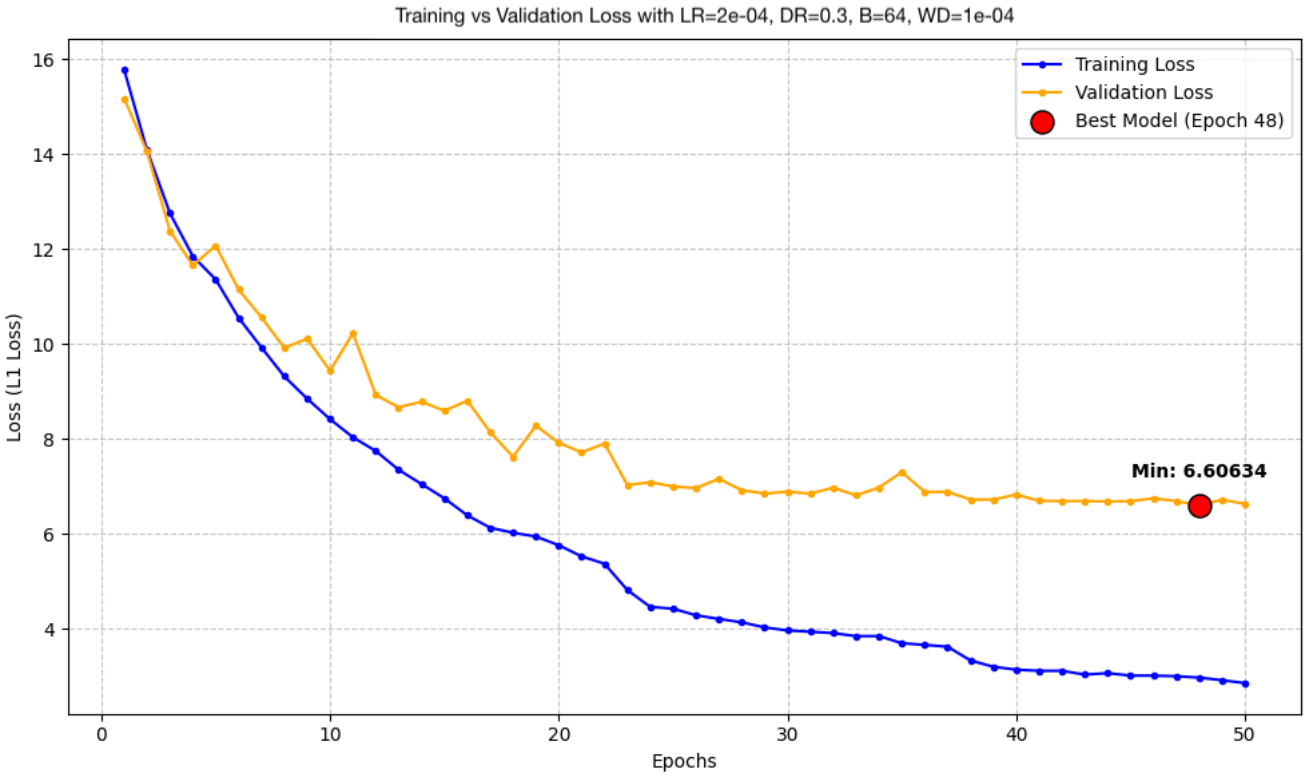

Figure 4. Training loss for the Translation stream. The rapid convergence indicates the effectiveness of the residual depth learning module.

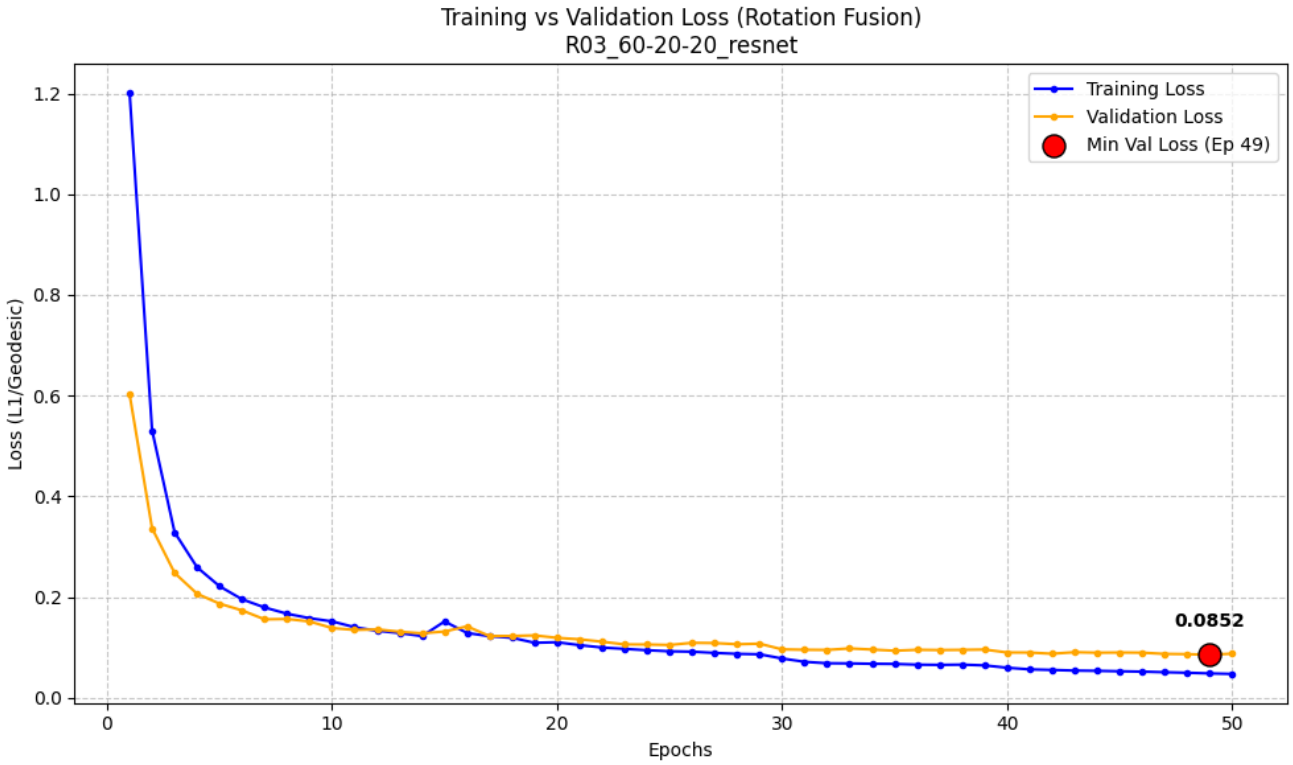

Figure 5. Training loss for the Rotation stream. The stability of the curve confirms that disentangling rotation from depth regression prevents optimization interference.

Table 3. Detailed performance metrics for Monocular Baseline. This table presents the complete breakdown of translation, rotation, and accuracy metrics for all objects.

Object	# Sam	Acc@10%	ADDR (cm)	ADDR@10%	ADD(S) (cm)	ADD (cm)	ADDS (cm)	AUC	Rot (°)	Tr (cm)
Ape	230	0.43%	0.98	57.83%	10.72	10.72	-	16.97%	19.72	10.64
Benchvise	246	1.63%	1.82	79.27%	9.48	9.48	-	23.97%	17.95	9.31
Camera	212	0.00%	1.51	74.53%	14.08	14.08	-	3.60%	17.55	14.01
Can	250	6.40%	1.46	81.20%	13.77	13.77	-	13.32%	15.73	13.67
Cat	239	2.93%	1.09	81.17%	7.94	7.94	-	30.99%	17.05	7.85
Driller	235	5.11%	1.92	77.45%	9.59	9.59	-	23.74%	17.44	9.21
Duck	264	0.00%	1.24	54.55%	15.34	15.34	-	0.58%	22.65	15.35
Eggbox	254	6.30%	0.44	100.00%	7.57	12.05	7.57	34.64%	17.83	11.91
Glue	260	22.31%	0.57	99.23%	10.01	13.76	10.01	48.99%	19.87	13.85
Holepuncher	258	0.00%	1.30	66.67%	11.61	11.61	-	14.68%	19.26	11.53
Iron	239	4.60%	2.00	79.08%	7.29	7.29	-	37.08%	17.46	6.93
Lamp	238	9.66%	1.87	86.97%	8.29	8.29	-	29.57%	17.50	7.97
Phone	235	2.98%	1.86	72.77%	9.26	9.26	-	24.13%	20.68	9.18
MEAN	**3160**	**4.91%**	**1.38**	**77.85%**	**10.39**	**11.06**	**8.80**	**23.25%**	**18.55**	**10.93**

Table 4. Detailed performance metrics for Joint RGBD Architecture. This table presents the complete breakdown of translation, rotation, and accuracy metrics for all objects using the split-stream approach.

Object	**# Sam**	**Acc@10%**	**ADDR (cm)**	**ADDR@10%**	**ADD(S) (cm)**	**ADD (cm)**	**ADDS (cm)**	**AUC**	**Rot (°)**	**Tr (cm)**
Ape	230	18.26%	0.87	67.83%	1.53	1.53	-	84.72%	17.31	1.26
Benchvise	246	45.12%	1.83	78.86%	2.84	2.84	-	71.66%	18.36	2.07
Camera	212	25.47%	1.46	71.23%	2.41	2.41	-	75.88%	16.37	1.82
Can	250	43.60%	1.77	70.00%	2.40	2.40	-	75.98%	19.66	1.51
Cat	239	46.03%	1.05	84.94%	1.76	1.76	-	82.43%	16.39	1.38
Driller	235	38.72%	2.18	73.19%	3.13	3.13	-	68.91%	21.23	2.02
Duck	264	11.36%	1.01	62.88%	1.88	1.88	-	81.22%	18.07	1.57
Eggbox	254	100.00%	0.43	100.00%	0.67	1.78	0.67	93.27%	16.33	1.09
Glue	260	98.85%	0.53	99.62%	0.81	1.84	0.81	91.86%	19.03	1.36
Holepuncher	258	30.23%	1.38	66.67%	1.94	1.94	-	80.58%	20.51	1.25
Iron	239	41.42%	2.07	75.73%	3.21	3.21	-	67.89%	18.16	2.39
Lamp	238	66.39%	1.89	86.55%	2.59	2.59	-	74.11%	17.79	1.62
Phone	235	36.17%	1.78	73.19%	2.70	2.70	-	72.98%	19.44	2.00
MEAN	**3160**	**46.77%**	**1.39**	**77.88%**	**2.13**	**2.30**	**0.74**	**78.58%**	**18.38**	**1.63**

Table 5. Detailed performance metrics for Main Extension. This table presents the complete breakdown of translation, rotation, and accuracy metrics for all objects, demonstrating the final improvements of our methodology.

Object	**# Sam**	**Acc@10%**	**ADDR (cm)**	**ADDR@10%**	**ADD(S) (cm)**	**ADD (cm)**	**ADDS (cm)**	**AUC**	**Rot (°)**	**Tr (cm)**
Ape	230	76.09%	0.28	100.00%	0.85	0.85	-	91.53%	5.41	0.79
Benchvise	246	73.98%	0.51	100.00%	1.71	1.71	-	82.85%	4.86	1.63
Camera	212	61.79%	0.39	100.00%	1.55	1.55	-	84.45%	4.46	1.48
Can	250	96.80%	0.43	100.00%	1.05	1.05	-	89.47%	4.51	0.95
Cat	239	90.79%	0.32	100.00%	1.08	1.08	-	89.17%	4.86	1.03
Driller	235	82.13%	0.52	100.00%	1.85	1.85	-	81.46%	4.59	1.75
Duck	264	67.80%	0.31	99.62%	0.98	0.98	-	90.19%	5.30	0.94
Eggbox	254	100.00%	0.16	100.00%	0.32	0.70	0.32	96.79%	4.85	0.58
Glue	260	100.00%	0.20	100.00%	0.53	1.11	0.53	94.65%	5.19	1.04
Holepuncher	258	76.36%	0.35	100.00%	1.03	1.03	-	89.72%	5.22	0.96
Iron	239	74.48%	0.54	100.00%	2.30	2.30	-	77.02%	4.78	2.22
Lamp	238	88.66%	0.54	100.00%	1.69	1.69	-	83.13%	4.65	1.56
Phone	235	70.64%	0.50	100.00%	1.76	1.76	-	82.43%	5.06	1.70
MEAN	**3160**	**81.80%**	**0.39**	**99.97%**	**1.27**	**1.35**	**0.43**	**87.14%**	**4.91**	**1.27**