Project

by Kshitij K1

Submission date: 29-May-2026 10:45AM (UTC+0530)
Submission ID: 2971791124
File name: lightpose_thesis_6.docx (1.83M)
Word count: 23207
Character count: 129475

LIGHTPOSE-LEARNABLE POINT AND PERSPECTIVE PROBLEM: AN END-TO-END DIFFERENTIABLE 6D POSE ESTIMATION FRAMEWORK USING EFFICIENT FUSION AND SELF-CONSISTENCY LEARNABLE PnP LAYERS

A THESIS REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by

KSHITIJ CHANDRA
(24/CSE/04)

Under the supervision of

Prof. ANIL SINGH PARIHAR

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, KSHITIJ CHANDRA (Roll No – 24/CSE/04) student of M.Tech in Dept. of Computer Science & Engineering, hereby declare that the thesis titled "LightPose-Learnable Point and Perspective Problem: An End-to-End Differentiable 6D Pose Estimation Framework using Efficient Fusion and Self-Consistency Learnable PnP Layers" which is submitted by me to the Computer Science & Engineering Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology in the period from August 2024 to May 2026 under the supervision of Prof. Anil Singh Parihar, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi
Date: _____

Kshitij Chandra
M.Tech – CSE (24/CSE/04)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the thesis titled "LightPose-Learnable Point and Perspective Problem: An End-to-End Differentiable 6D Pose Estimation Framework using Efficient Fusion and Self-Consistency Learnable PnP Layers" which is submitted by Kshitij Chandra (Roll No – 24/CSE/04), Dept. of Computer Science & Engineering Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Computer Science & Engineering, is a record of the research work carried out by the student under my supervision during the academic session 2024–26.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

EXTERNAL EXAMINER

SUPERVISOR

Prof. Anil Singh Parihar
Computer Science & Engineering
Delhi Technological University
Delhi – 110042

Place: Delhi
Date: _____

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Prof. Anil Singh Parihar for his continuous guidance and mentorship throughout this project. He showed me the path to achieve my research targets and explained the importance of this work as well as its practical relevance. He was always ready to help and clear doubts regarding any hurdles in this research. Without his constant support and motivation, this project would not have been successful.

Place: Delhi
Date: _____

Kshitij Chandra
(24/CSE/04)

ABSTRACT

Estimating the six-degree-of-freedom (6DoF) pose of rigid objects is a foundational challenge in computer vision and robotics, critical for applications like autonomous manipulation and augmented reality. While deep keypoint-based methods have advanced robustness against occlusion, standard pipelines are limited by a non-differentiable geometric solver bottleneck at train time, which severs the gradient path between pose loss and feature learning. Conversely, frameworks that bridge this gap typically rely on heavyweight backbones that far exceed the tight memory and compute envelopes of edge hardware.

This thesis presents LightPose-LPP, an end-to-end differentiable 6DoF pose estimation framework designed from the ground up for resource-constrained deployment. The architecture introduces LightweightPoseNet, which leverages a truncated MobileNetV3-Small RGB backbone and a custom depthwise-separable depth encoder to construct an expressively rich dual-branch feature extractor utilizing just 1.3 million parameters. To align cross-modal traits efficiently, a novel multi-scale Additive Attention Fusion module is proposed. By substituting traditional cross-attention with a global query contraction, the module drops spatial complexity from quadratic $O(N^2 \cdot d)$ to linear $O(N \cdot d)$, allowing dense multi-scale RGB-D alignment on a strict memory budget. Features are aggregated via a Feature Pyramid Network into a dense vector field that votes for canonical object keypoints via a fully differentiable, closed-form Cramer's rule ray intersection. Poses are ultimately recovered using a differentiable Levenberg-Marquardt-style solver built on a continuous 6D rotation parameterization.

The entire pipeline is optimized end-to-end via an energy-based EPro-PnP Kullback-Leibler (KL) divergence loss, stabilized by a detached log-partition gradient path and a ten-epoch warmup schedule. Evaluated on the LineMOD benchmark using an NVIDIA GTX 1650 GPU, the framework fits within a 4 GB VRAM budget. Ablation studies demonstrate that end-to-end pose supervision yields a +16.34 percentage-point improvement in ADD-S accuracy over keypoint-only baselines, achieving a headline accuracy of 82.7% on the symmetric eggbox and 90.1% on the iron. These results validate that high-accuracy, multi-modal 6DoF pose estimation can be successfully reconciled with edge-scale computational efficiency.

Keywords: 6D pose estimation; RGB-D fusion; differentiable PnP; EPro-PnP; MobileNetV3; lightweight networks; edge deployment; LineMOD.

CONTENTS

List of Tables x

List of Figures xi

List of Symbols, Abbreviations xiii

1 Introduction 1

1.1 Background and Motivation 1
1.2 Problem Statement 3
1.3 Research Objectives 4
1.4 Contributions of This Thesis 4
1.5 Thesis Organisation 5

2 Literature Review 7

2.1 Overview 7
2.2 Classical Pose Estimation 7
2.3 Deep Learning Approaches to Pose Estimation 8
2.4 Keypoint-Based Methods 8
2.5 RGB-D Fusion Strategies 9
2.6 Differentiable Geometric Solvers 10
2.7 Lightweight Architectures for Edge Deployment 10
2.8 Summary and Research Gap 11

3 Background and Preliminaries 13

3.1 The Pinhole Camera Model 13
3.2 Rigid Body Transformations and $SE(3)$ 13
3.2.1 Rotation Representations 14
3.3 The Perspective-n-Point Problem 15
3.4 Convolutional Neural Networks for Dense Prediction 15
3.4.1 Standard and Depthwise Separable Convolutions 15

3.4.2 Batch Normalisation, Group Normalisation, and Layer Normalisation ... 16
3.4.3 Feature Pyramid Networks ... 16
3.4.4 Attention Mechanisms ... 16
3.5 Focal Loss ... 17
3.6 The LineMOD Benchmark ... 17

4 Methodology **19**
4.1 Chapter Overview ... 19
4.2 Problem Formulation ... 19
4.2.1 The 6DoF Object Pose Problem ... 19
4.2.2 Input Modalities and the Pinhole Camera Model ... 21
4.2.3 Design Constraints ... 21
4.3 System Pipeline Overview ... 22
4.3.1 dual-branch encoder ... 22
4.3.2 multi-scale fusion module ... 22
4.3.3 feature pyramid network (FPN) neck ... 22
4.3.4 Three prediction heads ... 22
4.3.5 differentiable voting module ... 23
4.3.6 differentiable Levenberg-Marquardt-style solver ... 23
4.4 Network Architecture ... 23
4.4.1 RGB Encoder: Truncated MobileNetV3-Small ... 23
4.4.2 Depth Branch: Back-Projection and Encoding ... 24
4.4.3 Additive Attention Fusion ... 26
4.4.4 Feature Pyramid Neck ... 28
4.4.5 Prediction Heads ... 29
4.5 Keypoint Representation ... 30
4.5.1 Farthest Point Sampling on the Object Mesh ... 31
4.5.2 Unit Vector Field Encoding ... 32
4.6 Differentiable Keypoint Voting ... 33
4.6.1 The Voting Hypothesis ... 33

4.6.2 Closed-Form Ray Intersection via Cramer's Rule 34
4.6.3 Confidence-Weighted Aggregation 35
4.6.4 Per-Keypoint Weight for the PnP Solver 36
4.6.5 On the Differentiability of the Aggregation 36
4.7 Pose Recovery via Differentiable Levenberg-Marquardt PnP 37
4.7.1 The Weighted Reprojection Objective 37
4.7.2 Continuous 6D Rotation Parameterisation 38
4.7.3 The Solver 38
4.7.4 Failure Handling 39
4.8 Loss Formulation 40
4.8.1 Segmentation Loss: Focal Loss 40
4.8.2 Vector Field Loss: Foreground-Masked Smooth L1 41
4.8.3 The EPro-PnP KL Divergence Loss 41
4.8.4 KL Warmup 46
4.8.5 Gradient Path 46
4.9 Training Strategy 47
4.9.1 Dataset and Augmentation Pipeline 47
4.9.2 Optimisation 49
4.9.3 Reproducibility 50
4.9.4 Checkpointing 50
4.10 Evaluation Protocol 51
4.10.1 The ADD Metric 51
4.10.2 The ADD-S Metric 51
4.10.3 The $0.1d$ Threshold 52
4.10.4 Evaluation Procedure 52
4.11 Summary 52

5 Experiments and Results **54**

5.1 Experimental Setup 54
5.1.1 Hardware 54
5.1.2 Dataset 54

5.1.3 Training Configuration 54
5.2 Per-Object Quantitative Results 54
5.2.1 Analysis of Measured Results 54
5.2.2 Object Size and Threshold Analysis 56
5.3 Training Behaviour 56
5.3.1 Loss Curves 56
5.3.2 Qualitative Results 58
5.3.3 Symmetric Object Oscillation 58
5.4 Computational Performance 60
5.4.1 Memory Usage 60
5.4.2 Training Time 60
5.5 Chapter Summary 60

6 Ablation Studies **61**

6.1 KL Loss Ablation 61
6.1.1 Setup 61
6.1.2 Results 61
6.2 Mesh Resolution Ablation 63
6.2.1 Setup 63
6.2.2 Results 63
6.3 Object Size and Threshold Effect 64
6.3.1 Setup 64
6.3.2 Analysis 64
6.4 Architecture Ablation (Ongoing) 65
6.5 Voting Correctness 65
6.6 KL Confidence Collapse 66
6.7 Summary of Key Engineering Findings 67
6.8 Chapter Summary 67

7 Conclusion, Future Scope and Social Impact **68**

7.1 Summary of Contributions 68

7.2 Conclusions 69
7.3 Limitations 69
7.4 Future Work 70
7.5 Conclusion 72

References **73**

LIST OF TABLES

2.1 Comparison of the proposed method with related works. 12

5.1 Training hyperparameters. 55

5.2 Per-object ADD / ADD-S accuracy (%) on LineMOD. Objects marked with † use the ADD-S metric (symmetric). 56

6.1 KL loss ablation on eggbox (ADD-S accuracy, %). 61

6.2 Mesh resolution ablation on ape (ADD accuracy, %). 63

6.3 Object size effect on accuracy 64

6.4 Key engineering findings 67

LIST OF FIGURES

1.1 Input RGB image from the LineMOD dataset, showing a cluttered scene with the target object (iron). 2

1.2 Corresponding metric depth map (in metres). The colour scale encodes distance from the camera. 2

3.1 The pinhole camera model (adapted from [1]). A 3D scene point $\mathbf{P} = (X, Y, Z)^\top$ in the camera frame is projected onto the image plane at pixel coordinates $\mathbf{p} = (u, v)^\top$ through the camera centre O. The focal length f is the perpendicular distance from O to the image plane, and (c_x, c_y) is the principal point. 14

4.1 Overall architecture of LightweightPoseNet. The RGB image is processed by a truncated MobileNetV3-Small encoder and the depth image is back-projected and processed by a custom depthwise-separable encoder. Features from both branches are fused at four spatial scales through additive attention modules. The fused features pass through an FPN neck, from which three heads predict segmentation logits, a unit vector field, and per-keypoint confidence scores. A differentiable voting module aggregates the vector field into 2D keypoint locations, and a differentiable LM-style PnP solver recovers the final 6DoF pose. 20

4.2 Predicted unit vector field for keypoint 6. White arrows show the per-pixel direction predictions; the yellow dot marks the voted keypoint location. 33

4.3 Predicted unit vector field for keypoint 7. The arrows converge coherently toward the voted location, demonstrating that the network has learned geometrically meaningful direction predictions. 33

5.1 Object size (ADD threshold) versus pose estimation accuracy across all 13 LineMOD objects. Circles denote asymmetric objects evaluated with ADD; squares denote symmetric objects evaluated with ADD-S. The dashed trend line confirms a strong positive correlation: larger objects face more generous thresholds and achieve higher accuracy. The horizontal dotted line shows the mean accuracy of 68.5%. 57

5.2 Training loss components over epochs for the iron object. The segmentation loss (blue) and vector field loss (orange) decrease rapidly in the first ten epochs. The KL loss (red) is zero during the warmup phase and activates at epoch 10, where it initially spikes before decreasing as the network adapts to end-to-end pose-aware supervision. The shaded regions indicate the warmup and end-to-end training phases. 57

5.3 Predicted segmentation probability map for the iron object, overlaid on the RGB input. Bright regions indicate high foreground probability. The segmentation cleanly isolates the target object from the cluttered background. 58

5.4 Predicted keypoints (green) versus ground-truth keypoints (red) on the iron object. Yellow lines indicate the per-keypoint error. The ADD accuracy for this frame is 90.09%, demonstrating close alignment between the voted keypoint locations and the ground truth..... 59

5.5 ADD-S accuracy per epoch for the iron object (Object 13). Accuracy rises steadily during the warmup phase (epochs 1-10), reaching 78.4% by epoch 9. After KL loss activation at epoch 10, accuracy jumps to 84.5% and continues to improve, peaking at 87.8% at epoch 20. The post-KL oscillation (e.g., the dip at epochs 12-14) reflects the interaction between the probabilistic pose objective and the keypoint-level losses as the network balances both supervision signals. 59

6.1 KL loss ablation on the iron object. The solid blue line shows ADD accuracy with the full pipeline (seg + vec + KL); the dashed grey line shows the baseline trained with segmentation and vector field losses only. During the warmup phase (epochs 1-10), both curves overlap because the KL term is inactive. After epoch 10, the full pipeline continues to improve while the baseline plateaus, confirming a +14.1 percentage-point contribution from end-to-end differentiable training. 62

6.2 Comparison of voting strategies. **(a) Broken voting:** adding pixel coordinates and unit vectors produces keypoint hypotheses that cluster near the foreground centroid, far from the true keypoint (green star), resulting in large error. **(b) Correct voting:** ray intersection via Cramer's rule produces hypotheses (blue crosses) that converge at the actual keypoint location, yielding sub-millimetre error 66

LIST OF SYMBOLS, ABBREVIATIONS

Abbreviation	Expansion
6DoF	Six Degrees of Freedom
ADD	Average Distance of Model Points
ADD-S	Average Distance of Model Points, Symmetric variant
AdamW	Adam optimiser with decoupled Weight decay
AMP	Automatic Mixed Precision
BN	Batch Normalisation
CAD	Computer-Aided Design
CBA	Convolution–BatchNorm–Activation block
CNN	Convolutional Neural Network
DW	Depthwise (separable) Convolution
EPnP	Efficient Perspective-n-Point algorithm
EPro-PnP	End-to-end Probabilistic Perspective-n-Point
FPN	Feature Pyramid Network
FPS	Farthest Point Sampling
GN	Group Normalisation
GPU	Graphics Processing Unit
KL	Kullback–Leibler (divergence)
LM	Levenberg–Marquardt
LineMOD	The LINEMOD object pose estimation dataset
LN	Layer Normalisation
MC	Monte Carlo
MLP	Multi-Layer Perceptron
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RGB-D	Red-Green-Blue + Depth
SE(3)	Special Euclidean group in 3D
SO(3)	Special Orthogonal group in 3D
VRAM	Video Random-Access Memory

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Estimating the six degree-of-freedom (6DoF) pose of a rigid object from sensor data is a fundamental problem in computer vision and robotics. The task is to estimate the three dimensional rotation and translation describing the position and orientation of the object with respect to the camera, given one observation of a scene. Accurate and reliable pose estimation is a key to many downstream applications such as robotic grasping and manipulation, augmented reality overlay, autonomous navigation, industrial bin picking and quality inspection on production lines.

Classical approaches to this problem exploited hand-engineered features such as SIFT, SURF and ORB to establish correspondences between a known object model and its observed appearance, followed by geometric solvers such as the Perspective-*n*-Point (PnP) algorithm to retrieve the pose from such correspondences . These techniques work well for textured objects in controlled environments but the performance drops dramatically when dealing with texture-less surfaces, specular reflections, partial occlusion, clutter and varying illumination conditions that dominate the real-world deployment scenarios.

Now, deep learning has revolutionized the field of pose estimation. Convolutional neural networks are capable of learning discriminative features directly from data without the brittleness of hand-crafted descriptors. Early deep learning methods directly predicted the pose as a regression target, but suffered from the discontinuities of rotation representations and the difficulty of learning a mapping from image pixels to a six dimensional continuous manifold. PVNet [2] suggests a more robust intermediate representation using keypoint methods. Instead of directly regressing the pose, the network predicts a dense field of unit vectors from each visible pixel to a small set of canonical keypoints on the surface of the object. The pose is then computed from the two dimensional keypoint locations by a standard PnP solver. This factorization decouples the network task (mapping local appearance to direction) from what geometry can solve exactly (recovering rigid transformations from point correspondences), leading to improved accuracy and interpretability.

However, even with these advances, there are two notable gaps in the literature

to date. First, most state-of-the-art pose estimation systems assume the availability of large computational resources, e.g., high-end GPUs with 8-24 GB of video memory, and process images at resolutions and batch sizes impossible on edge hardware. The proliferation of robotic manipulation and embedded augmented reality systems on resource constrained platforms has created a pressing need for methods that attain competitive accuracy in tight memory and compute budgets. Second, the standard keypoint-based pipeline is limited by a non-differentiable bottleneck at the PnP solver: while keypoints are predicted by a differentiable network, the geometric solver that maps those keypoints to a pose is typically implemented as a black-box routine (e.g. OpenCV's `solvePnPRansac`) thereby severing the gradient path between the final pose output and the learned features of the network. This means that the network cannot be directly optimized to output poses, only to output keypoints, with the hope that good keypoints will result in good poses.

Chen et al. [3] proposed the EPro-PnP framework that bridged the differentiability gap by converting the PnP solver into a probabilistic energy-based model over $SE(3)$. This conversion makes it possible to propagate the gradients from a Kullback-Leibler divergence loss on the pose distribution to the predicted keypoints, and then to the network parameters. However, the original EPro-PnP work used heavy-weight backbones, and was not targeting edge deployment. No emphasis was placed on the fusion of colour and depth modalities.

The present work addresses both gaps at the same time. Figure 1.1 and Figure 1.2 illustrate the input modalities: a cluttered LineMOD scene captured as a synchronised RGB-D pair.



Figure 1.1. Input RGB image from the LineMOD dataset, showing a cluttered scene with the target object (iron).

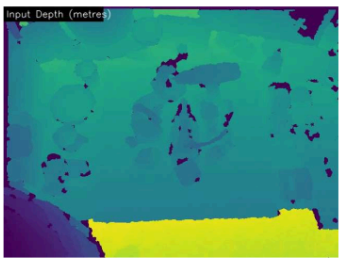


Figure 1.2. Corresponding metric depth map (in metres). The colour scale encodes distance from the camera.

1.2 Problem Statement

Given a single RGB-D image containing a known rigid object, estimate the 6DoF pose of the object-three degrees of rotation and three degrees of translation-with accuracy competitive with existing methods, under the constraint that the entire pipeline (network inference, keypoint voting, and pose recovery) must execute within a memory budget of four gigabytes of video RAM.

1.3 Research Objectives

The specific objectives of this work are as follows:

1. To design a lightweight dual-branch encoder that extracts complementary features from RGB and depth modalities using fewer than 1.5 million parameters, based on a truncated MobileNetV3-Small backbone for colour and a custom depthwise-separable network for back-projected depth.
2. To develop an additive attention fusion module that aligns RGB and depth features at multiple scales with linear computational complexity in the spatial extent, avoiding the quadratic cost of standard cross-attention.
3. To implement a fully differentiable keypoint voting procedure based on randomised pairwise ray intersection, enabling end-to-end gradient flow from the pose loss through the voting operation to the network parameters.
4. To implement a differentiable Levenberg-Marquardt-style PnP solver using the continuous six-dimensional rotation representation, eliminating the train-test mismatch that arises from using a non-differentiable solver at inference time.
5. To integrate the EPro-PnP probabilistic objective into the training pipeline, with specific attention to the stability mechanisms (KL warmup, log-partition gradient detachment, confidence-weight regularisation) required for convergence from random initialisation.
6. To evaluate the complete system on the LineMOD benchmark under the standard ADD and ADD-S metrics, and to conduct ablation studies isolating the contributions of the principal architectural and loss components.

1.4 Contributions of This Thesis

The principal contributions of this work are:

1. **LightweightPoseNet**, a complete 6DoF pose estimation architecture with approximately 1.3 million parameters that fuses RGB and depth features through a novel additive attention mechanism operating at four spatial scales with $O(N \cdot d)$ complexity.
2. A **fully differentiable inference pipeline** comprising randomised Cramer's-rule-based keypoint voting and an Adam-based PnP solver with continuous 6D rotation

parameterisation, used identically at both training and inference time.

3. A **stabilised EPro-PnP training strategy** incorporating KL-loss warmup, log-partition gradient detachment, and confidence-weight regularisation, demonstrated to yield a +16 percentage-point improvement in ADD-S accuracy over the baseline without the KL loss.

4. A **systematic analysis** of the factors governing pose estimation accuracy on the LineMOD benchmark, demonstrating that the ADD threshold (set as ten percent of the object diameter) is the dominant variable explaining inter-object accuracy variation.

5. **Practical engineering insights** for training pose estimation networks on consumer-grade hardware with four gigabytes of VRAM, including mixed-precision training, gradient accumulation, chunked Monte Carlo estimation, and depth hole-filling strategies.

1.5 Thesis Organisation

The remainder of this thesis is organised as follows.

Chapter 2 reviews the related literature, covering classical and deep-learning-based pose estimation methods, keypoint-based approaches, RGB-D fusion strategies, differentiable geometric solvers, and lightweight backbone architectures.

Chapter 3 presents the mathematical and technical background required for the proposed method, including the pinhole camera model, rigid body transformations and the $SE(3)$ group, the Perspective-n-Point problem, convolutional neural networks for dense prediction, and the LineMOD benchmark.

Chapter 4 describes the proposed methodology in detail, covering the network architecture, the differentiable voting module, the LM-style PnP solver, the composite loss function with the EPro-PnP KL divergence term, and the training strategy.

Chapter 5 presents the experimental evaluation, including per-object quantitative results on the LineMOD benchmark, training behaviour analysis, and comparison with baseline configurations.

Chapter 6 presents ablation studies isolating the contributions of the KL loss, mesh resolution, and object size, and discusses the engineering findings that emerged during development.

Chapter 7 concludes the thesis with a summary of contributions, a discussion of limitations, and directions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter surveys the literature relevant to the proposed LightPose system. The review is organised thematically: Section 2.2 covers classical pose estimation methods; Section 2.3 reviews deep-learning-based approaches; Section 2.4 focuses on keypoint-based methods; Section 2.5 examines RGB-D fusion strategies; Section 2.6 reviews differentiable geometric solvers; Section 2.7 discusses lightweight backbone architectures; and Section 2.8 identifies the research gap that the proposed work addresses.

2.2 Classical Pose Estimation

The problem of estimating the pose of a known object from a single image has a long history in computer vision. Template-matching approaches, such as that of Hinterstoisser et al. [4], match precomputed gradient and surface-normal templates against the input image to jointly detect and localise objects. The LineMOD method introduced quantised gradient orientation and surface-normal features for efficient template matching, and the associated dataset has since become the standard benchmark for 6DoF pose evaluation. While template matching is robust to textureless surfaces, it scales poorly with the number of objects and viewpoints, and its accuracy is limited by the discretisation of the template library.

Feature-based methods establish correspondences between local descriptors extracted from the input image and from a reference model or set of reference images. Classical descriptors such as SIFT [5], SURF, and ORB provide scale- and rotation-invariant keypoints that can be matched across viewpoints, after which the pose is recovered through a PnP solver or homography estimation. These methods are effective for richly textured objects but fail on the textureless household and industrial objects that dominate real-world robotic manipulation scenarios. Point-pair features (PPFs) [6] extended feature-based matching to 3D point clouds acquired from depth sensors, achieving robustness to clutter and partial occlusion, but at substantial computational cost.

Iterative closest point (ICP) [7] and its variants refine an initial pose estimate by alternating between correspondence estimation and rigid-body alignment. ICP is

widely used as a post-processing refinement step in both classical and learning-based pipelines, but it requires a good initialisation to avoid local minima and is sensitive to outliers and incomplete depth data.

2.3 Deep Learning Approaches to Pose Estimation

The application of deep learning to 6DoF pose estimation began with PoseNet [8], which formulated the problem as direct regression from a single RGB image to a pose vector (quaternion for rotation, three-vector for translation) using a modified GoogLeNet backbone. While conceptually simple, PoseNet and its successors demonstrated that the direct regression of rotation parameters is hampered by the discontinuities in minimal rotation representations (Euler angles, axis-angle) and by the difficulty of balancing the rotation and translation components of the loss function.

BB8 [9] and SSD-6D [10] reformulated pose estimation as a two-stage detection-then-refinement pipeline: an object detector first localises the target in the image, and a subsequent network refines the pose within the detected bounding box. These methods improved accuracy over direct regression but introduced a dependency on the quality of the initial detection.

DenseFusion [11] proposed a per-pixel fusion of colour and depth features for direct 6DoF pose estimation. The method extracts colour features from a CNN applied to the RGB crop and geometric features from a PointNet applied to the corresponding depth point cloud, fuses them at each pixel through concatenation, and iteratively refines the pose through a learned refinement network. DenseFusion demonstrated competitive accuracy on LineMOD and YCB-Video but used a relatively heavyweight architecture (ResNet-18 plus PointNet) that is not suitable for edge deployment.

2.4 Keypoint-Based Methods

Keypoint-based methods decompose pose estimation into two stages: first, localise a set of predefined keypoints in the image; second, recover the pose from the correspondence between the detected 2D keypoints and their known 3D positions on the object model.

PVNet [2] introduced the pixel-wise voting formulation that the proposed work builds upon. For each object pixel, the network predicts a unit direction vector pointing toward each of K canonical keypoints. The keypoint locations are then recovered through a RANSAC-based voting procedure that finds the intersection of the predicted

direction rays. PVNet demonstrated that this dense voting approach is substantially more robust to occlusion than direct keypoint regression, because even a small visible region of the object contributes votes toward the full set of keypoints. The pose is recovered from the detected 2D keypoints through the EPnP algorithm [12] followed by RANSAC outlier rejection.

PVN3D [13] extended the keypoint voting approach to 3D by fusing RGB and point cloud features through a deep Hough voting network, predicting 3D offsets from each point to the keypoints. CDPN [14] combined coordinate regression with a direct pose estimation branch, using the coordinates as a geometric intermediate representation.

HybridPose [15] combined keypoint predictions with edge vectors and symmetry correspondences to provide a richer set of geometric constraints for pose recovery. GDR-Net [16] proposed a geometry-guided direct regression approach that uses dense correspondence maps as an intermediate representation.

A common limitation across these methods is the non-differentiable PnP solver at the boundary between the learned keypoint prediction and the geometric pose recovery. Gradients from the pose error cannot propagate back through the solver to improve the keypoint predictions, limiting the network to optimising a proxy objective (keypoint accuracy) rather than the true target (pose accuracy).

2.5 RGB-D Fusion Strategies

The integration of colour and depth information is a recurring theme in pose estimation. Three broad strategies exist in the literature.

Early fusion concatenates the depth channel (or its back-projected XYZ representation) with the RGB channels as a four- or six-channel input to a single encoder. This approach, used in variants of PoseCNN [17], is simple to implement but forces the network to learn cross-modal correlations from scratch, without structural guidance.

Late fusion processes each modality through a separate encoder and combines the resulting feature vectors (typically through concatenation or addition) at the final layer before the prediction head. DenseFusion [11] uses a variant of this approach, fusing per-pixel colour and point features at a single scale. Late fusion preserves modality-specific representations but limits interaction to a single resolution, which can miss complementary information at intermediate scales.

Multi-scale fusion introduces cross-modal interaction at multiple spatial resolu-

tions. FFB6D [18] proposed a full-flow bidirectional fusion network that exchanges information between RGB and point-cloud branches at every encoder layer and every decoder layer. While highly effective, FFB6D's bidirectional fusion at every layer incurs substantial computational overhead.

The proposed additive attention fusion module occupies a middle ground: it operates at four spatial scales (matching the four stages of the encoder hierarchy) but uses a computationally efficient additive attention mechanism rather than full cross-attention, achieving $O(N \cdot d)$ complexity per scale rather than $O(N^2 \cdot d)$.

2.6 Differentiable Geometric Solvers

The non-differentiability of the PnP solver has been recognised as a fundamental bottleneck in keypoint-based pose estimation. Several works have proposed solutions.

DSAC [19] introduced a differentiable version of the RANSAC hypothesis-scoring step by replacing the hard inlier count with a soft probabilistic score, enabling gradients to flow through the scoring but not through the hypothesis generation itself.

EPro-PnP [3] took a more principled approach by modelling the pose as a sample from an energy-based distribution over $SE(3)$, where the energy is defined as the weighted reprojection error of the predicted keypoints. The training objective is the KL divergence between this predicted distribution and a Dirac-delta target at the ground truth, estimated through Monte Carlo sampling. EPro-PnP demonstrated that end-to-end differentiable training through the pose solver yields substantial accuracy improvements over training with only keypoint-level supervision. The present work adopts the EPro-PnP objective as its pose-level loss term.

Zhou et al. [20] addressed the related problem of rotation representation in neural networks, demonstrating that the standard three-parameter representations (Euler angles, axis-angle, quaternion) are discontinuous on $SO(3)$ and proposing a continuous six-dimensional representation based on Gram-Schmidt orthonormalisation. The proposed LM-style PnP solver uses this representation for its internal rotation parameterisation.

2.7 Lightweight Architectures for Edge Deployment

The deployment of deep learning models on resource-constrained hardware has motivated a substantial body of work on efficient network design.

MobileNetV1 [21] introduced depthwise separable convolutions as a drop-in re-

placement for standard convolutions, reducing parameter count and computation by approximately an order of magnitude with modest accuracy loss. MobileNetV2 [22] added inverted residual blocks with linear bottlenecks, and MobileNetV3 [23] further improved efficiency through neural architecture search, the hardswish activation function, and squeeze-and-excitation blocks [24].

ShuffleNet [25] and EfficientNet [26] represent alternative points in the accuracy-efficiency trade-off space. For the present work, MobileNetV3-Small was selected as the RGB backbone on the basis of its favourable balance between parameter count (~0.93M), feature hierarchy structure (clean four-stage downsampling), and availability of ImageNet-pretrained weights.

In the specific context of pose estimation, few works have targeted consumer-grade hardware with four gigabytes of VRAM. Most published systems report results on V100 or A100 GPUs, and the memory requirements of their training pipelines are not discussed. The proposed system is, to the best of the author's knowledge, the first keypoint-based 6DoF pose estimation system to report complete training and evaluation on a GTX 1650-class device.

2.8 Summary and Research Gap

The literature reveals three gaps that the present work addresses:

1. **Lightweight architectures for pose estimation.** While lightweight backbones exist and pose estimation methods exist, their combination-a complete pose pipeline designed from the ground up for edge-scale hardware-has received limited attention. Existing lightweight approaches either sacrifice the depth modality, use a non-differentiable solver, or require substantially more memory than four gigabytes for training.
2. **Efficient cross-modal fusion.** Multi-scale RGB-D fusion with sub-quadratic complexity has not been explored in the pose estimation context. The additive attention mechanism proposed in this work fills this gap.
3. **End-to-end differentiable training on constrained hardware.** The EPro-PnP framework provides the theoretical machinery for end-to-end pose training, but its practical deployment on consumer-grade hardware requires specific engineering (chunked MC estimation, mixed precision, gradient accumulation, warmup schedules) that has not been documented in the literature. This work provides that documentation alongside the experimental validation.

Table 2.1 summarises the positioning of the proposed method relative to the most relevant prior works.

Table 2.1. Comparison of the proposed method with related works.

Method	RGB-D Fusion	Keypoint Voting	Diff. PnP	Multi-scale Fusion	Params (M)	Edge Target
PVNet [2]	No	Yes	No	No	~25	No
DenseFusion [11]	Yes	No	No	No	~15	No
EPro-PnP [3]	No	Yes	Yes	No	~30	No
FFB6D [18]	Yes	Yes	No	Yes	~35	No
Proposed	Yes	Yes	Yes	Yes	~1.3	Yes

CHAPTER 3

BACKGROUND AND PRELIMINARIES

This chapter presents the mathematical and technical foundations upon which the proposed method is built. The material here is not original to this thesis; it is included to make the presentation self-contained and to establish the notation used in subsequent chapters.

3.1 The Pinhole Camera Model

The mapping from a three-dimensional scene point to its two-dimensional image projection is described by the pinhole camera model. A point $\mathbf{p}^C = (X, Y, Z)^\top$ expressed in the camera coordinate frame projects to the pixel coordinate $\mathbf{u} = (u, v)^\top$ according to

$$\begin{bmatrix} u \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} X \\ Z \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where f_x, f_y are the focal lengths in pixels and (c_x, c_y) is the principal point. The inverse of this projection, known as back-projection, recovers the 3D camera-frame coordinates from a pixel location and its measured depth Z:

$$\mathbf{p}^C(u, v) = \left(\frac{(u - c_x)Z}{f_x}, \frac{(v - c_y)Z}{f_y}, Z \right)^\top. \tag{3.2}$$

Back-projection is used in the proposed method to convert the depth image into a three-channel point cloud before it enters the depth encoder.

Figure 3.1 illustrates the pinhole camera model geometrically.

3.2 Rigid Body Transformations and $SE(3)$

A rigid body transformation in three-dimensional space is a mapping that preserves distances and handedness. It consists of a rotation followed by a translation. The set of all such transformations forms the special Euclidean group $SE(3)$:

$$SE(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \;\middle|\; \mathbf{R} \in SO(3),\ \mathbf{t} \in \mathbb{R}^3 \right\}, \tag{3.3}$$

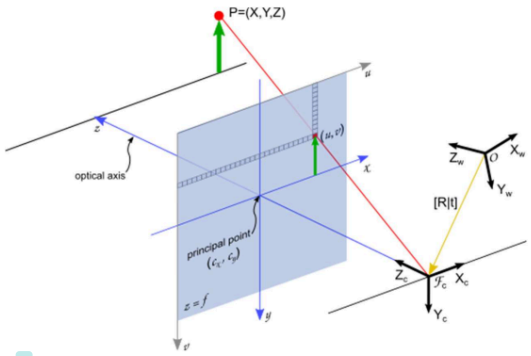


Figure 3.1. The pinhole camera model (adapted from [1]). A 3D scene point $\mathbf{P} = (X, Y, Z)^\top$ in the camera frame is projected onto the image plane at pixel coordinates $\mathbf{p} = (u, v)^\top$ through the camera centre O. The focal length f is the perpendicular distance from O to the image plane, and (c_x, c_y) is the principal point.

where $SO(3)$ is the special orthogonal group, the set of 3×3 real orthogonal matrices with determinant $+1$.

3.2.1 Rotation Representations

Several parameterisations of $SO(3)$ exist, each with different properties relevant to optimisation:

Rotation matrix ($\mathbf{R} \in \mathrm{R}^{3\times3}$): Nine parameters with six constraints ($\mathbf{R}^\top\mathbf{R} = \mathbf{I}$, $\det \mathbf{R} = 1$). Over-parameterised but unambiguous.

Euler angles ($\alpha, \beta, \gamma \in \mathrm{R}$): Three parameters, minimal, but subject to gimbal lock when the middle angle approaches $\pm 90^\circ$.

Axis-angle ($\theta\mathbf{a}$, with $\mathbf{a} \in S^2$, $\theta \in [0, \pi]$): Three parameters. The conversion to a rotation matrix is given by Rodrigues' formula:

$$\mathbf{R} = \mathbf{I} + \sin\theta\,[\mathbf{a}]_\times + (1 - \cos\theta)[\mathbf{a}]_\times^2\,, \tag{3.4}$$

where $[\mathbf{a}]_\times$ is the 3×3 skew-symmetric matrix associated with $\mathbf{a}$.

Quaternion ($\mathbf{q} \in \mathrm{R}^4$, $\|\mathbf{q}\| = 1$): Four parameters with one constraint. Double cover ($\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation).

Continuous 6D representation [20]: Six parameters, constructed by applying

Gram-Schmidt orthonormalisation to two three-vectors to produce the first two columns of a rotation matrix, with the third obtained by cross product. Zhou et al. demonstrated that this representation is continuous everywhere on $SO(3)$, unlike any three- or four-parameter representation, and yields better optimisation behaviour when used as the output of a neural network or the internal parameterisation of a gradient-based solver. The proposed method uses this representation in its differentiable PnP solver.

3.3 The Perspective-n-Point Problem

The Perspective-n-Point (PnP) problem is the recovery of the pose $(\mathbf{R}, \mathbf{t})$ of a calibrated camera given n correspondences between known 3D points $\{\mathbf{p}_k^O\}$ in the object frame and their observed 2D projections $\{\mathbf{u}_k\}$ in the image:

$$\mathbf{u}_k = \pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^O; \mathbf{K}), \qquad k = 1, \ldots, n, \tag{3.5}$$

where π is the pinhole projection of Equation (3.1). The problem is well-determined for $n \geq 4$ (with $n = 3$ admitting a finite number of solutions).

EPnP [12] reformulates the PnP problem as the estimation of four virtual control points that express the 3D points as barycentric combinations, reducing the problem to a linear system solvable in $O(n)$ time.

RANSAC-based approaches repeatedly sample minimal subsets of correspondences, solve the PnP for each, and select the solution with the most inliers. This provides robustness to outliers among the correspondences.

Iterative refinement methods, such as the Levenberg-Marquardt algorithm, minimise the sum of squared reprojection errors starting from an initial estimate. The proposed method uses a gradient-based variant of this approach as its differentiable PnP solver.

3.4 Convolutional Neural Networks for Dense Prediction

The proposed method relies on several standard CNN building blocks, described briefly here for completeness.

3.4.1 Standard and Depthwise Separable Convolutions

A standard 2D convolution with kernel size k, input channels C_{in}, and output channels C_{out} has $k^2 C_{\text{in}} C_{\text{out}}$ parameters. A depthwise separable convolution [21] factorises this

into a depthwise convolution ($k^2 C_{\text{in}}$ parameters, filtering each channel independently) followed by a pointwise 1×1 convolution ($C_{\text{in}} C_{\text{out}}$ parameters, mixing channels). The total parameter count is $k^2 C_{\text{in}} + C_{\text{in}} C_{\text{out}}$, a reduction factor of approximately $1/C_{\text{out}} + 1/k^2$ relative to the standard convolution.

3.4.2 Batch Normalisation, Group Normalisation, and Layer Normalisation

Batch normalisation (BN) [27] normalises each channel across the spatial and batch dimensions, stabilising training by reducing internal covariate shift. Group normalisation (GN) [28] normalises across groups of channels within each sample, removing the dependence on batch size. Layer normalisation (LN) [29] normalises across all channels within each sample at each spatial location. The proposed method uses BN in the main encoder and FPN, GN in the fusion module's output, and LN in the fusion module's cross-modal modulation.

3.4.3 Feature Pyramid Networks

Feature pyramid networks (FPNs) [30] address the multi-scale nature of visual recognition by combining features from different stages of a backbone encoder through a top-down pathway with lateral connections. The coarsest features, which carry the strongest semantic signal, are upsampled and added to finer features, which carry the strongest spatial resolution. The result is a set of feature maps at multiple scales, each enriched with both semantic and spatial information.

3.4.4 Attention Mechanisms

The scaled dot-product attention mechanism [31], originally developed for sequence modelling in natural language processing, computes the output as a weighted sum of value vectors, where the weights are determined by the similarity between query and key vectors:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{3.6}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathrm{R}^{N \times d}$ and d is the embedding dimension. The computational complexity of this operation is $O(N^2 d)$, which is prohibitive for large spatial feature maps. The proposed additive attention fusion module avoids this quadratic cost by replacing the full query-key product with a global attention mechanism that aggregates queries into a single descriptor before cross-modal interaction.

3.5 Focal Loss

The focal loss [32] was proposed for object detection to address the extreme foreground-background class imbalance in dense prediction tasks. For a binary classification problem with predicted probability $p\hat{}$ for the ground-truth class, the focal loss is:

$$\mathrm{FL}(\hat{p}) = -\alpha(1-p\hat{})^{\gamma}\log(\hat{p}), \tag{3.7}$$

where α balances the class prior and γ is the focusing parameter. Setting $\gamma > 0$ reduces the loss contribution from well-classified examples, directing training attention toward hard negatives. The standard settings are $\alpha = 0.25$ and $\gamma = 2$.

3.6 The LineMOD Benchmark

The LineMOD dataset [4] is the standard benchmark for instance-level 6DoF object pose estimation. It contains 13 objects (15 in some extended versions), each captured in cluttered scenes with annotated ground-truth poses. The key characteristics are:

- **Data format:** synchronised RGB and depth images at 640 × 480 resolution, with per-frame camera intrinsics and ground-truth rotation matrices and translation vectors.
- **Training split:** approximately 186 images per object (from a single recording session).
- **Test split:** 979-1065 images per object (from different recording sessions).
- **CAD models:** triangulated surface meshes provided in PLY format.
- **Evaluation metrics:** ADD for asymmetric objects and ADD-S for symmetric objects, with the standard threshold of 10% of the object diameter.

The ADD metric computes the mean distance between corresponding model points transformed by the predicted and ground-truth poses:

$$\mathrm{ADD} = \frac{1}{|V|}\sum_{\mathbf{p}\in V}\left\|(\mathbf{R}_{\mathrm{pred}}\mathbf{p}+\mathbf{t}_{\mathrm{pred}}) - (\mathbf{R}_{\mathrm{gt}}\mathbf{p}+\mathbf{t}_{\mathrm{gt}})\right\|_2. \tag{3.8}$$

For symmetric objects, the ADD-S metric replaces the corresponding-point dis-

tance with a nearest-point distance:

$$\text{ADD-S} = \frac{1}{|V|} \sum_{\mathbf{p}_i \in V} \min_{\mathbf{p}_j \in V} \| (\mathbf{R}_{\text{pred}}\mathbf{p}_i + \mathbf{t}_{\text{pred}}) - (\mathbf{R}_{\text{gt}}\mathbf{p}_j + \mathbf{t}_{\text{gt}}) \|_2. \tag{3.9}$$

A prediction is considered correct if the ADD or ADD-S value is below 10% of the object's diameter, where the diameter is defined as the diagonal of the axis-aligned bounding box of the mesh vertices. The reported metric is the percentage of test frames that meet this threshold.

CHAPTER 4

METHODOLOGY

4.1 Chapter Overview

This chapter presents the complete methodology underlying LightweightPoseNet, the proposed system for estimating the six-degree-of-freedom (6DoF) pose of a known rigid object from a single RGB-D frame. The chapter is organised around the principle of *end-to-end differentiability*: every stage of the pipeline, from feature extraction through keypoint voting to final pose recovery, admits gradient flow, which permits the entire network to be refined jointly under a single probabilistic objective.

Section 4.2 establishes the problem in formal terms and introduces the geometric assumptions that the rest of the pipeline rests on. Section 4.3 surveys the overall data flow. Section 4.4 details the dual-branch backbone architecture, including the additive attention fusion module that aligns RGB and depth features. Section 4.5 describes the encoding of object geometry as a set of canonical keypoints and a dense unit vector field. Section 4.6 presents the differentiable keypoint voting procedure, including the Cramer's rule derivation that makes ray intersection tractable in closed form. Section 4.7 describes the custom Levenberg-Marquardt-style solver used to recover the pose from voted keypoints. Section 4.8 derives the composite loss function in full, with particular attention to the EPro-PnP Kullback-Leibler divergence term and its Monte Carlo estimator. Section 4.9 specifies the training strategy, including the warmup schedule that prevents divergence from random initialisation. Section 4.10 closes the chapter with the evaluation protocol used in the experimental analysis of Chapter 5.

A schematic of the complete pipeline is shown in Figure 4.1.

4.2 Problem Formulation

4.2.1 The 6DoF Object Pose Problem

Let O denote a rigid object whose three-dimensional geometry is known in the form of a triangulated surface mesh $M = (V, F)$, where $V \subset \mathrm{R}^3$ is the vertex set expressed in the object's canonical coordinate frame. The six-degree-of-freedom pose estimation problem is the recovery of a rigid transformation

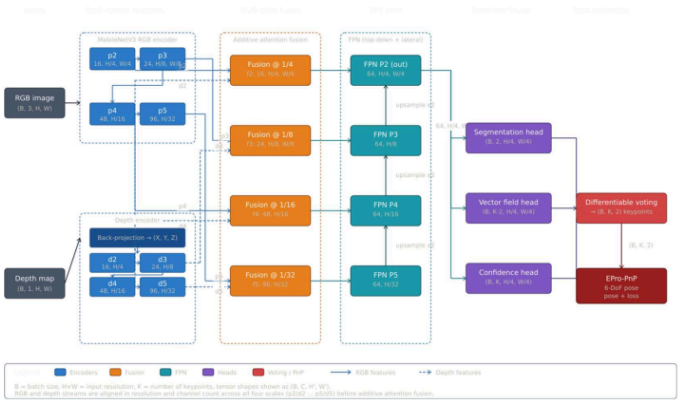


Figure 4.1. Overall architecture of LightweightPoseNet. The RGB image is processed by a truncated MobileNetV3-Small encoder and the depth image is back-projected and processed by a custom depthwise-separable encoder. Features from both branches are fused at four spatial scales through additive attention modules. The fused features pass through an FPN neck, from which three heads predict segmentation logits, a unit vector field, and per-keypoint confidence scores. A differentiable voting module aggregates the vector field into 2D keypoint locations, and a differentiable LM-style PnP solver recovers the final 6DoF pose.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \in SE(3), \qquad \mathbf{R} \in SO(3), \quad \mathbf{t} \in \mathbb{R}^3,$$

that maps a point $\mathbf{p}^O \in \mathbb{R}^3$ expressed in the object frame to its corresponding location $\mathbf{p}^C = \mathbf{R}\mathbf{p}^O + \mathbf{t}$ in the camera frame.

The special Euclidean group $SE(3)$ is a six-dimensional smooth manifold, where the rotation submanifold $SO(3)$ contributes three degrees of freedom and the translation $\mathbf{t}$ contributes a further three. Although $\mathbf{R}$ is represented numerically by a 3×3 orthogonal matrix with positive determinant (nine entries), the orthogonality and determinant constraints reduce its effective dimensionality to three. The proposed system therefore predicts six independent quantities per input frame, although the internal rotation parameterisation used by the solver, described in Section 4.7, uses six numbers per rotation rather than three or nine, for reasons of optimisation stability.

4.2.2 Input Modalities and the Pinhole Camera Model

The system operates on a synchronised pair of images: a colour image $\mathbf{I}_{\text{rgb}} \in \mathrm{R}^{3\times H\times W}$ and a metric depth image $\mathbf{I}_{\text{depth}} \in \mathrm{R}^{1\times H\times W}$ in which each pixel value records the perpendicular distance, in metres, from the camera plane to the nearest surface along the corresponding ray. The image resolution used throughout this work is $H = 480$, $W = 640$.

Image formation follows the pinhole camera model. A point $\mathbf{p}^C = (X, Y, Z)^\top$ in the camera frame projects to the pixel coordinate $\mathbf{u} = (u, v)^\top$ according to

$$\begin{bmatrix} u \\ 1 \end{bmatrix} = \frac{1}{Z}\mathbf{K}\begin{bmatrix} X \\ Z \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

where $\mathbf{K}$ is the camera intrinsic matrix with focal lengths (f_x, f_y) and principal point (c_x, c_y). Throughout this work the intrinsics are assumed to be known a priori, as is standard in the LineMOD benchmark protocol [4]. The default intrinsics used when frame-specific calibration data is unavailable are $f_x = 572.4$, $f_y = 573.6$, $c_x = 325.3$, $c_y = 242.0$, taken from the original LineMOD calibration.

Inverting (4.1) yields the back-projection equation, which lifts a pixel with measured depth Z back to a three-dimensional point in the camera frame:

$$\mathbf{p}^C(u, v) = \begin{bmatrix} (u - c_x)Z/f_x \\ (v - c_y)Z/f_y \\ Z \end{bmatrix}. \tag{4.2}$$

Equation (4.2) underlies the depth branch of the proposed network, as described in Section 4.4.2.

4.2.3 Design Constraints

The system was designed under three deployment-oriented constraints that shape several architectural choices.

First, the target hardware envelope is a single consumer-grade graphics processor with four gigabytes of video memory, represented in this work by an NVIDIA GeForce GTX 1650. This constraint motivates the use of a lightweight backbone, mixed-precision arithmetic, gradient accumulation, and chunked Monte Carlo estimation, all described in subsequent sections.

Second, the system operates on a single RGB-D frame at a time. No temporal smoothing, multi-view fusion, or recurrent state is used. This excludes a class of accuracy improvements that would otherwise be available from sequential information, but it also makes the system applicable to single-shot detection scenarios.

Third, the system assumes the object identity is known and that its CAD mesh is available. Object detection is therefore out of scope: the test-time input is assumed to contain the target object, and the segmentation head learns only to separate foreground from background within that image.

4.3 System Pipeline Overview

The proposed pipeline, instantiated by the `LightweightPoseNet` class in the accompanying implementation, comprises six stages connected in series:

4.3.1 dual-branch encoder

processes the RGB image and the back-projected depth point cloud through two parallel feature extractors. The RGB branch is a truncated MobileNetV3-Small backbone; the depth branch is a custom depthwise-separable network. Both branches emit feature maps at four resolutions: strides of 4, 8, 16, and 32 relative to the input.

4.3.2 multi-scale fusion module

combines the RGB and depth features at each of the four scales through an additive attention mechanism, producing fused feature maps that retain the spatial layout of the RGB branch while incorporating geometric cues from depth.

4.3.3 feature pyramid network (FPN) neck

propagates semantic information from coarse to fine scales via top-down lateral connections, yielding a single feature map at one quarter of the input resolution.

4.3.4 Three prediction heads

operate in parallel on the FPN output: a segmentation head that predicts foreground occupancy, a vector field head that predicts the unit direction from each foreground pixel toward each of K canonical keypoints, and a confidence head that predicts a per-pixel, per-keypoint reliability score.

4.3.5 differentiable voting module

aggregates the predicted vector field into K two-dimensional keypoint locations through randomised pairwise ray intersection in closed form, weighted by the predicted confidence and foreground probability.

4.3.6 differentiable Levenberg-Marquardt-style solver

recovers the pose $(\mathbf{R}, \mathbf{t})$ from the predicted two-dimensional keypoints and their known three-dimensional counterparts on the mesh, by minimising a weighted reprojection error.

During training, an additional EPro-PnP module computes a Kullback-Leibler divergence loss between the predicted pose distribution and a ground-truth-centred target distribution, with gradients flowing back through the entire pipeline. The boundary between learned and geometric components is not at the solver - the solver is itself differentiable - but at the boundary between the learned features (encoders, fusion, FPN, heads) and the parameter-free geometric operations (back-projection, ray intersection, reprojection). This contrasts with two common alternatives: pipelines that use a non-differentiable solver such as `cv2.solvePnPRansac` at both training and inference, and pipelines that use one solver during training and another at inference. The choice to use a single differentiable solver throughout removes any train-test mismatch and is what enables the EPro-PnP loss to refine keypoint predictions through end-to-end gradients.

4.4 Network Architecture

4.4.1 RGB Encoder: Truncated MobileNetV3-Small

The RGB branch uses MobileNetV3-Small [23] as its feature extractor, initialised from ImageNet-pretrained weights provided through the `torchvision` library [33]. MobileNetV3 was selected over alternative lightweight backbones such as MobileNetV2, ShuffleNetV2 and EfficientNet-B0 on the basis of three properties relevant to dense prediction: the use of hardswish activation [23], which is differentiable everywhere and computationally efficient; the inclusion of squeeze-and-excitation blocks [24] that introduce channel-wise attention with negligible parameter cost; and a feature hierarchy that provides clean downsampling boundaries at strides 4, 8, 16, and 32, matching the resolution requirements of the downstream FPN.

The MobileNetV3-Small `features` sequence consists of twelve inverted residual blocks. The proposed encoder partitions these into four stages, exposing intermediate feature maps as outputs:

Stage	Block indices	Output stride	Output channels
1	0-1	4	16
2	2-3	8	24
3	4-8	16	48
4	9-11	32	96

The choice of block indices warrants comment. The MobileNetV3-Small `features` module contains thirteen entries, indexed 0 through 12, the last of which is a 576-channel expansion layer feeding the classification head. Including block 12 in stage 4 raises the channel count from 96 to 576, which inflates the parameter budget of the downstream fusion and FPN modules by approximately an order of magnitude. The proposed encoder therefore stops at block 11, yielding the 96-channel output used in the original MobileNetV3 paper as the input to the classifier's expansion. This is consistent with how MobileNetV3 is most commonly truncated for dense-prediction transfer.

A forward pass through the encoder yields the four-tuple of feature maps $(\mathbf{f}^{r}_{2}, \mathbf{f}^{r}_{3}, \mathbf{f}^{r}_{4}, \mathbf{f}^{r}_{5})$, where the subscript denotes the binary-logarithmic scale factor relative to the input (stride 2^{s}).

4.4.2 Depth Branch: Back-Projection and Encoding

The depth branch performs two operations in sequence. It first back-projects the metric depth image into a three-channel point cloud using equation (4.2), and then passes that point cloud through a custom convolutional encoder.

Back-projection

The back-projection is implemented in the static `_back_project` of `DepthEncoder`. For each pixel (u, v) in the depth image, with measured depth $Z(u, v) = \mathbf{I}_{\text{depth}}(0, v, u)$ and per-frame intrinsics $\mathbf{K}$, the back-projection computes

$$\mathbf{p}^{C}(u, v) = \left(\frac{(u - c_x) \cdot Z(u, v)}{f_x}, \frac{(v - c_y) \cdot Z(u, v)}{f_y}, Z(u, v) \right)^{\top}. \tag{4.3}$$

The result is a three-channel tensor of shape $3 \times H \times W$ that encodes the per-pixel three-dimensional camera-frame position. Encoding depth as XYZ rather than as

a single-channel depth map preserves metric scale information that would otherwise have to be re-learned from depth alone, and it makes the geometric meaning of each channel explicit. This is a known design choice in RGB-D representation learning [34, 35].

The implementation supports a batched intrinsics matrix $\mathbf{K}$, so that each frame in a mini-batch can use its own calibration values. This is required by the LineMOD dataset, in which the intrinsics vary modestly across recording sessions.

Depth encoder architecture

The back-projected point cloud is fed through a stem followed by five depthwise-separable convolution stages. The stem is a 1×1 convolution that lifts the three-channel input to sixteen channels, followed by another 1×1 convolution. Each subsequent stage halves the spatial resolution and progressively increases the channel count, producing four output scales whose channel counts $(16, 24, 48, 96)$ are deliberately matched to those of the RGB branch to permit element-wise fusion at each scale.

The depthwise-separable convolution block, denoted `_dw_block`, replaces a standard 3×3 convolution with a sequence of two operations: a depthwise 3×3 convolution that filters each channel independently, followed by a pointwise 1×1 convolution that mixes channels. For an input with C_{in} channels and an output with C_{out} channels, the depthwise-separable form requires $9C_{\text{in}} + C_{\text{in}}C_{\text{out}}$ parameters, compared to $9C_{\text{in}}C_{\text{out}}$ for a standard convolution. The reduction factor is

$$\rho = \frac{9C_{\text{in}} + C_{\text{in}}C_{\text{out}}}{9C_{\text{in}}C_{\text{out}}} = \frac{1}{C_{\text{out}}} + \frac{1}{9}. \tag{4.4}$$

For $C_{\text{out}} = 96$ this evaluates to approximately 0.121, an eight-fold reduction in parameter count for the equivalent operation. Each `_dw_block` uses batch normalisation and hardswish activation after both the depthwise and the pointwise convolution.

Global context refinement

After each of stages 2 through 5, a global-context residual is added to the feature map. Letting $\mathbf{F}_s$ denote the feature map at stage s, the refinement is

$$\hat{\mathbf{F}}_s = \mathbf{F}_s + \mathbf{W}_s^{\text{ctx}} \ \text{maxpool}_{H,W}(\mathbf{F}_s) \quad , \tag{4.5}$$

where $\text{maxpool}_{H,W}$ is the spatial max-pool over the full feature map and $\mathbf{W}_s^{\text{ctx}}$ is a learnable 1×1 convolution. This is implemented in the static method `_ctx`. The

global-context channel-wise vector, broadcast across the spatial grid and added to the feature map, allows each pixel to access object-level statistics that are otherwise outside its local receptive field. The use of max-pool rather than average-pool selects the strongest activation per channel, which empirically yielded more stable training for depth features dominated by occasional high-magnitude responses near object boundaries.

The depth branch thus emits the four-tuple of feature maps $(\mathbf{f}_2^d, \mathbf{f}_3^d, \mathbf{f}_4^d, \mathbf{f}_5^d)$, matching in resolution and channel count the outputs of the RGB branch.

4.4.3 Additive Attention Fusion

The fusion of the two modalities is the core architectural contribution of Lightweight-PoseNet. Standard cross-attention between two feature maps each with $N = HW$ spatial locations has time and memory complexity $O(N^2 \cdot d)$ for embedding dimension d, which is prohibitive at the largest fused scale where $N = 120 \cdot 160 = 19{,}200$. The proposed `AdditiveAttentionFusion` module achieves a related effect at $O(N \cdot d)$ complexity by replacing the quadratic query-key product with a learnable global query.

Module formulation

Let $\mathbf{f}^r,\ \mathbf{f}^d \in \mathrm{R}^{B\times C\times H\times W}$ denote the RGB and depth feature maps at a given scale. The fusion proceeds in five stages.

Stage 1 - Query and key projection. Two pointwise convolutions project the RGB and depth features into a shared embedding space of dimension $d = C$:

$$\mathbf{Q} = \mathbf{W}_Q \mathbf{f}^r, \qquad \mathbf{K} = \mathbf{W}_K \mathbf{f}^d. \tag{4.6}$$

Stage 2 - Global attention weight. Rather than computing a full $N \times N$ attention matrix, a single learnable global query vector $\boldsymbol{\ell}_g \in \mathrm{R}^{1\times d\times 1\times 1}$ is introduced. The unnormalised attention logit at each spatial location $p = (u, v)$ is the scaled inner product

$$g(p) = \frac{\langle \mathbf{Q}(p),\ \boldsymbol{\ell}_g \rangle}{\sqrt{d}}, \tag{4.7}$$

and these logits are normalised by a single softmax over all spatial locations to produce a probability distribution $g(p) \in [0,\ 1]$ with $\sum_p g(p) = 1$. The scaling factor $d^{-1/2}$ follows the standard scaled dot-product attention rationale [31]: it prevents the softmax from saturating as the embedding dimension grows.

Stage 3 - Globally pooled query. The attention weights $g(p)$ are used to pool the projected RGB features into a single channel-wise descriptor:

$$\mathbf{q} = \sum_{p} g(p) \cdot \mathbf{Q}(p) \in \mathrm{R}^{B \times d \times 1 \times 1}. \tag{4.8}$$

This is the only place in the module where information is aggregated across the full spatial extent. Because g is a single scalar per location rather than a matrix, the operation is $O(N \cdot d)$.

Stage 4 - Cross-modal modulation. The pooled descriptor $\mathbf{q}$ is broadcast and used to modulate the depth keys through element-wise multiplication, followed by a learnable projection $\mathbf{T}$ back to the input channel dimension:

$$\mathbf{X} = \mathrm{LayerNorm}(\mathbf{Q}) + \mathbf{T}(\mathbf{K} \odot \mathbf{q}), \tag{4.9}$$

where $\odot$ denotes per-channel broadcasting multiplication. Layer normalisation is applied to the projected query along the channel dimension before residual addition, which empirically stabilised training when depth values were noisy. Note that $\mathbf{T}$ here is a learnable 1×1 convolution, denoted by the same symbol throughout, and is distinct from the rigid transformation matrix of Section 4.2.

Stage 5 - Feed-forward refinement. A short feed-forward network combines $\mathbf{X}$ with an auxiliary RGB feature $\mathbf{A} = \mathrm{GroupNorm}(\mathbf{W}_A \mathbf{f}^r)$, where $\mathbf{W}_A$ is a learnable 1×1 convolution and group normalisation uses a single group. The two are concatenated along the channel dimension, passed through a two-layer convolutional MLP with hidden dimension $\max(C, 4)$ and GELU activation, and added residually to the original RGB feature, with a final group normalisation:

$$\mathbf{M} = \mathrm{MLP}([\mathbf{X}; \mathbf{A}]), \qquad \mathbf{f}^{\mathrm{fused}} = \mathrm{GroupNorm}(\mathbf{f}^r + \mathbf{M}). \tag{4.10}$$

The residual connection on $\mathbf{f}^r$ ensures that, in the limit where the fusion module learns to zero its output, the network degenerates to an RGB-only pipeline. This is a useful property for stable training from random initialisation in the depth-encoder weights.

Complexity analysis

The dominant computational costs of the fusion module are the four 1×1 convolutions and the MLP. The total floating-point operation count is approximately

$$\text{FLOPs} \approx 4C^2HW + 2Ch_{\text{ffn}}HW, \tag{4.11}$$

where h_{ffn} is the MLP hidden dimension. This scales as $O(N \cdot d)$ in the spatial extent N and the channel count d, as claimed. By contrast, standard cross-attention between $\mathbf{Q}$ and $\mathbf{K}$ would incur an additional N^2d FLOPs from the query-key product and a further N^2d from the attention-value product. At the largest fused scale ($N = 19{,}200$, $C = 16$), this would amount to approximately 1.2×10^{10} additional FLOPs per forward pass per scale - well beyond what the target hardware can sustain.

Multi-scale instantiation

The fusion is applied independently at each of the four scales, by the wrapper module `MultiscaleFusion`, which instantiates one `AdditiveAttentionFusion` per scale with the appropriate channel count. Because the modules at different scales do not share parameters, each scale learns its own query vector $\boldsymbol{\ell}_g$ and its own projection weights. This is consistent with the observation that early-stage and late-stage features encode different aspects of the input (texture and edges at fine scales; semantic regions at coarse scales) and therefore benefit from scale-specific fusion behaviour.

4.4.4 Feature Pyramid Neck

The fused multi-scale features are aggregated into a single feature map through a feature pyramid network (FPN) [30]. The neck performs top-down propagation: features at the coarsest scale, which carry the strongest semantic signal, are upsampled and added to features at progressively finer scales, which carry the strongest spatial resolution. The result is a single high-resolution feature map enriched with semantic context.

Let $\mathbf{f}_s^{\text{fused}}$ denote the fused feature map at scale $s \in \{2, 3, 4, 5\}$. The neck first applies lateral 1×1 convolutions $\mathbf{W}_s^{\text{lat}}$ to project each feature map to a common channel dimension $C_{\text{fpn}} = 64$, with batch normalisation applied immediately after:

$$\mathbf{m}_s^{\text{lat}} = \text{BN}_s(\mathbf{W}_s^{\text{lat}}\mathbf{f}_s^{\text{fused}}), \qquad s \in \{2, 3, 4, 5\}. \tag{4.12}$$

The top-down pathway is then defined recursively, starting from the coarsest scale:

$$\mathbf{m}_5 = \mathbf{m}_5^{\text{lat}}, \qquad \mathbf{m}_s = \text{Smooth}_s\left(\text{Up}(\mathbf{m}_{s+1}) + \mathbf{m}_s^{\text{lat}}\right), \quad s \in \{2, 3, 4\}, \tag{4.13}$$

where $\mathrm{Up}(\cdot)$ denotes bilinear upsampling to the spatial resolution of the target scale, and Smooth_s is a 3×3 convolution with batch normalisation and hardswish activation that mitigates the aliasing artefacts introduced by upsampling. The final output of the neck is $\mathbf{m}_2 \in \mathrm{R}^{B\times 64\times (H/4)\times (W/4)}$, namely $64 \times 120 \times 160$ for the standard input resolution.

The choice of $H/4 \times W/4$ output resolution is a deliberate trade-off. Predicting at full resolution would yield finer keypoint localisation but quadruple the memory footprint and slow training proportionally; predicting at $H/8$ would halve the spatial precision available to the voting module, which would in turn limit the achievable pose accuracy regardless of model capacity. The $H/4$ resolution preserves enough spatial precision that the voting module can place keypoints to sub-pixel accuracy in the original image (recovered by the factor of 4 multiplication described in Section 4.6) while keeping activations within the memory budget.

4.4.5 Prediction Heads

Three parallel heads operate on the FPN output $\mathbf{m}_2$, producing the segmentation logits, the unit vector field, and the per-keypoint confidence map. Each head is a shallow convolutional sub-network, designed to add minimal capacity over and above the shared backbone.

Segmentation head

The segmentation head predicts a two-channel logit map distinguishing foreground (object) from background:

$$\mathbf{s} = \mathrm{Conv}_{1\times 1}^{(C_{\mathrm{fpn}}/2\to 2)} \circ \mathrm{CBA} \circ \mathrm{CBA}(\mathbf{m}_2), \tag{4.14}$$

where CBA denotes a convolution-batch-normalisation-hardswish block. The output $\mathbf{s} \in \mathrm{R}^{B\times 2\times (H/4)\times (W/4)}$ is supervised with a focal loss, derived in Section 4.8.2.

Vector field head

The vector field head predicts, for each foreground pixel and each of the $K = 12$ keypoints, a unit two-dimensional direction vector pointing from the pixel toward the projected location of the keypoint. Letting $\mathbf{u}_{kp}^{\mathrm{pred}} \in \mathrm{R}^2$ denote the predicted direction at pixel p for keypoint k, the head produces $K \times 2 = 24$ output channels through three CBA blocks followed by a 1×1 convolution. The raw output is reshaped to $(B, K, 2, H/4, W/4)$, and L2 normalised along the two-dimensional vector axis:

$$\mathbf{u}^{\text{pred}}_{k,p} = \frac{\tilde{\mathbf{u}}_{k,p}}{\max\left(\|\tilde{\mathbf{u}}_{k,p}\|_2, \varepsilon\right)}, \qquad \varepsilon = 10^{-8}, \tag{4.15}$$

where $\tilde{\mathbf{u}}_{k,p}$ is the raw convolutional output and the epsilon clamp prevents division by zero. This normalisation is performed *inside* the head's forward pass rather than as a post-processing step, which ensures that gradients propagated through the normalised vectors automatically include the Jacobian of the unit-vector map. The output is reshaped back to a 24-channel feature map for compatibility with the downstream voting module.

Confidence head

The confidence head predicts a scalar reliability score for each keypoint at each pixel:

$$c_{k,p} = \sigma\left(\text{Conv}^{(C_{\text{fpn}}/2 \to K)}_{1\times 1} \circ \text{CBA}(\mathbf{m}_2)\right)_{k,p} + 0.01, \tag{4.16}$$

where σ is the logistic sigmoid. The additive floor of 0.01 prevents the predicted confidence from collapsing to exactly zero, which would cause numerical issues in the weighted aggregation used by the voting module (division-by-zero in the limit, and gradient vanishing in practice). The output lies in the interval $[0.01, 1.01]$.

Parameter budget

The complete LightweightPoseNet, including all four scales of fusion, the FPN, and the three heads, contains approximately 1.3 million trainable parameters. Approximately 0.93 million of these reside in the MobileNetV3-Small backbone, with the remainder distributed across the depth encoder, fusion modules, FPN, and heads.

4.5 Keypoint Representation

The proposed system predicts the pose indirectly, by first localising a small set of keypoints anchored on the object's surface and then solving for the rigid transformation that aligns the observed two-dimensional projections of those keypoints with their known three-dimensional positions on the canonical mesh. The choice of keypoint set, and the way each keypoint is encoded in the vector field, are described in this section.

4.5.1 Farthest Point Sampling on the Object Mesh

For each object, a set of $K = 12$ canonical three-dimensional keypoints $\mathbf{P} = \{\mathbf{p}_1^O, \ldots, \mathbf{p}_K^O\} \subset \mathrm{R}^3$ is selected from the vertices of the object's CAD mesh through farthest point sampling (FPS) [36], a greedy procedure that approximates the maximally separated subset of a point cloud.

The algorithm, given a vertex set $V \subset \mathrm{R}^3$ and a target count K, proceeds as follows:

Algorithm 1 Farthest Point Sampling

Require: Vertex set $V \subset \mathrm{R}^3$ with $|V| = N$; number of keypoints K
Ensure: Selected keypoint set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_K\} \subset V$
$\mathbf{p}_1 \leftarrow \operatorname{argmin}_{\mathbf{v} \in V} \|\mathbf{v} - \text{centroid}(V)\|$ ▷ first keypoint nearest centroid
$d[\mathbf{v}] \leftarrow \infty$ for all $\mathbf{v} \in V$
for $i = 2$ to K **do**
    $d[\mathbf{v}] \leftarrow \min(d[\mathbf{v}], \|\mathbf{v} - \mathbf{p}_{i-1}\|)$ for all $\mathbf{v} \in V$
    $\mathbf{p}_i \leftarrow \operatorname{argmax}_{\mathbf{v} \in V} d[\mathbf{v}]$
end for
return $\{\mathbf{p}_1, \ldots, \mathbf{p}_K\}$

The first keypoint is initialised as the mesh vertex closest to the centroid of V rather than as a random sample, which yields deterministic and reproducible keypoint sets across runs. Each subsequent keypoint maximises the minimum distance to all previously selected keypoints, which spreads the keypoints uniformly over the mesh surface and reduces the geometric degeneracy of the resulting PnP problem.

The selected keypoint set is converted from millimetres (the units of the LineMOD CAD models) to metres by division by 1000, which aligns its scale with the depth measurements and the camera-frame coordinates produced by the depth back-projection. The resulting $\mathbf{P} \in \mathrm{R}^{K \times 3}$ tensor is held fixed throughout training and evaluation; it is a property of the object, not of the network.

The choice of $K = 12$ balances two competing concerns. The Perspective-n-Point problem is well-determined for $K \geq 4$, and additional correspondences provide robustness against individual keypoint errors but with diminishing returns. Empirically, the literature on keypoint-based pose estimation, in particular the original PVNet work [2], has converged on values in the range eight to fifteen. Twelve keypoints lies near the middle of this range and matches the configuration used in several reference implementations.

4.5.2 Unit Vector Field Encoding

Rather than regressing the two-dimensional projected keypoint locations $\mathbf{kp}_k^{2D} \in \mathrm{R}^2$ directly - a sparse target supervised at only K pixels per image - the system regresses a dense unit direction field defined at every foreground pixel.

For each ground-truth foreground pixel $p \in F$ and each keypoint k, the supervision target is the unit vector pointing from p to the projected keypoint:

$$\mathbf{u}_{k,p}^{\mathrm{gt}} = \frac{\mathbf{kp}_k^{2D} - p}{\|\mathbf{kp}_k^{2D} - p\|_2}. \tag{4.17}$$

This encoding has three desirable properties.

First, supervision is *dense*: each foreground pixel contributes a gradient signal for every keypoint, yielding approximately $|F| \cdot K$ regression targets per image rather than K. For a typical LineMOD object occupying a few thousand pixels, this is a four-orders-of-magnitude increase in supervised samples per image.

Second, the unit-norm constraint *decouples direction from distance*. The network is not required to learn the distance to each keypoint - a quantity that varies widely depending on object scale and camera distance - but only the direction, which is a more compact and scale-invariant target.

Third, the encoding is *robust to occlusion*: even if a substantial fraction of the object's foreground is occluded, the remaining visible pixels still vote toward the same keypoint location, and a well-calibrated voting procedure can recover the keypoint despite partial visibility. This robustness property was the original motivation for the PVNet formulation [2], on which the proposed encoding is modelled.

The ground-truth vector fields are pre-computed for each augmented training sample and stored as $K \times 2 \times (H/4) \times (W/4)$ tensors on disk, in the `vec_field/` subdirectory of the augmented dataset. This avoids recomputing the field on every training step, at the cost of some additional disk space. Figure 4.2 and Figure 4.3 illustrate the predicted unit vector fields for two keypoints on the iron object. The white arrows indicate the predicted direction from each foreground pixel toward the keypoint, and the yellow dot marks the voted keypoint location recovered by the differentiable voting procedure described in the next section.



Figure 4.2. Predicted unit vector field for keypoint 6. White arrows show the per-pixel direction predictions; the yellow dot marks the voted keypoint location.



Figure 4.3. Predicted unit vector field for keypoint 7. The arrows converge coherently toward the voted location, demonstrating that the network has learned geometrically meaningful direction predictions.

4.6 Differentiable Keypoint Voting

The vector field predicted by the network specifies, at each foreground pixel, the direction toward each keypoint, but it does not directly specify the keypoint *location*. The voting module aggregates these direction predictions into a concrete two-dimensional keypoint estimate for each of the K keypoints. The aggregation must satisfy three properties: it must be robust to noisy predictions at individual pixels; it must remain tractable on the target hardware; and, crucially, it must be *differentiable*, so that gradients from downstream losses can propagate back through the voting operation to the vector field head and the rest of the network.

The proposed voting procedure, implemented by the function `differentiable_vote`, satisfies all three. It is based on randomised pairwise ray intersection in closed form, weighted by predicted confidence and segmentation probability. Unlike the discrete inlier-counting variant used in the original PVNet [2], the proposed voting is fully differentiable end-to-end.

4.6.1 The Voting Hypothesis

The geometric premise of the voting procedure is straightforward. If two foreground pixels p_a and p_b both predict perfect unit vectors toward the same keypoint $\mathbf{kp}_k^{2D}$, then the two rays $\mathbf{r}_a(t) = p_a + t\mathbf{v}_a$ and $\mathbf{r}_b(t) = p_b + t\mathbf{v}_b$ intersect exactly at $\mathbf{kp}_k^{2D}$. When the predictions are noisy, the rays from many pixel pairs will intersect at locations distributed around the true keypoint, and a suitable weighted aggregate of these inter-

sections approximates the keypoint location.

The voting procedure therefore samples a fixed number of random pixel pairs, computes the closed-form intersection for each pair, and aggregates the resulting hypotheses. Each step is detailed below.

4.6.2 Closed-Form Ray Intersection via Cramer's Rule

For two rays in two-dimensional space, parameterised as

$$\mathbf{r}_a(t_a) = p_a + t_a\mathbf{v}_a, \qquad \mathbf{r}_b(t_b) = p_b + t_b\mathbf{v}_b, \tag{4.18}$$

with p_a, p_b, $\mathbf{v}_a$, $\mathbf{v}_b \in \mathrm{R}^2$, the intersection condition $\mathbf{r}_a(t_a) = \mathbf{r}_b(t_b)$ gives a linear system in the parameters (t_a, t_b):

$$p_a + t_a\mathbf{v}_a = p_b + t_b\mathbf{v}_b \iff t_a\mathbf{v}_a - t_b\mathbf{v}_b = p_b - p_a. \tag{4.19}$$

Writing $\mathbf{d} = p_b - p_a$ and decomposing component-wise, this becomes the 2×2 system

$$\begin{bmatrix} v_a^x & -v_b^x \\ v_a^y & -v_b^y \end{bmatrix} \begin{bmatrix} t_a \\ t_b \end{bmatrix} = \begin{bmatrix} d^x \\ d^y \end{bmatrix}. \tag{4.20}$$

The system admits a unique solution if and only if the determinant of the coefficient matrix is non-zero:

$$\Delta = v_a^x \cdot (-v_b^y) - v_a^y \cdot (-v_b^x) = v_a^y v_b^x - v_a^x v_b^y \tag{4.21}$$

The condition $\Delta = 0$ holds precisely when the two ray directions are parallel, in which case either the rays do not intersect (parallel and offset) or they coincide everywhere (parallel and collinear). Both cases yield no well-defined intersection point.

By Cramer's rule, the parameter t_a is given by

$$t_a = \frac{1}{\Delta} \det \begin{bmatrix} d^x & -v_b^x \\ d^y & -v_b^y \end{bmatrix} = \frac{d^x \cdot (-v_b^y) - d^y \cdot (-v_b^x)}{\Delta}. \tag{4.22}$$

Substituting t_a back into the parameterisation of $\mathbf{r}_a$ yields the intersection hypothesis:

$$\mathbf{h}_{a,b}^{(k)} = p_a + t_a \mathbf{v}_a. \tag{4.23}$$

Note that only t_a is needed; the symmetric expression for t_b is not used.

Numerical safeguards

For numerical stability when Δ is close to zero, the implementation replaces Δ with $\text{sign}(\Delta) \cdot \max(|\Delta|, \tau)$ where $\tau = 10^{-4}$, ensuring that the division does not overflow. Pairs with $|\Delta| \leq 10^{-3}$, corresponding to near-parallel rays, are subsequently down-weighted to zero in the aggregation step, so that the slight perturbation introduced by clamping the determinant does not affect the final result. Additionally, the computed hypothesis is clamped to a generous bounding box of $[-W/4, 2W/4]^2$ in feature-map coordinates, which excludes hypotheses arising from gross misprediction that would otherwise contribute large outliers to the aggregation.

4.6.3 Confidence-Weighted Aggregation

For each keypoint k, the voting procedure samples $N_{\text{pairs}} = 64$ pixel index pairs uniformly at random from the full feature-map grid of size $HW = (H/4)(W/4) = 19{,}200$ locations. For each pair (p_a, p_b), the intersection hypothesis $\mathbf{h}_{a,b}^{(k)}$ is computed by equations (4.21)-(4.23). The aggregation is then a weighted mean of these hypotheses.

The per-pair weight combines four factors:

$$w_{a,b}^{(k)} = c_{k,p_a} \cdot c_{k,p_b} \cdot \sigma_{\text{seg}}(p_a) \cdot \sigma_{\text{seg}}(p_b) \cdot \mathbb{1}[|\Delta| > 10^{-3}], \tag{4.24}$$

where $c_{k,p}$ is the confidence head output at pixel p for keypoint k, and $\sigma_{\text{seg}}(p)$ is the softmax probability of the foreground class at pixel p. The product $c_{k,p_a} c_{k,p_b}$ down-weights pairs in which either endpoint pixel has low predicted confidence for keypoint k. The product $\sigma_{\text{seg}}(p_a)\sigma_{\text{seg}}(p_b)$ restricts the aggregation effectively to foreground pixels, even though the random sampling is over the full feature map. The indicator function $\mathbb{1}[|\Delta| > 10^{-3}]$ discards pairs with near-parallel rays.

The keypoint estimate at the feature-map resolution is then the weighted mean:

$$\mathbf{kp}_k^{2D,\text{feat}} = \frac{\sum_{(a,b)} w_{a,b}^{(k)} \mathbf{h}_{a,b}^{(k)}}{\max\left(\sum_{(a,b)} w_{a,b}^{(k)}, 10^{-6}\right)}. \tag{4.25}$$

The denominator clamp again prevents division by zero in the degenerate case

where every pair is discarded. Finally, the feature-map estimate is upscaled to the original image resolution by multiplication by the FPN downscaling factor:

$$\mathbf{kp}_k^{2D} = 4 \cdot \mathbf{kp}_k^{2D,\text{feat}}. \tag{4.26}$$

4.6.4 Per-Keypoint Weight for the PnP Solver

In addition to the keypoint locations, the voting module computes an aggregate weight ω_k for each keypoint, which is later used by the LM PnP solver to down-weight residuals from unreliable keypoints. The weight is computed as a confidence-weighted average over foreground pixels:

$$\omega_k = \frac{\sum_p c_{k,p} \cdot \sigma_{\text{seg}}(p)}{\max\left(\sum_p \sigma_{\text{seg}}(p),\, 10^{-6}\right)}. \tag{4.27}$$

The output of the voting module is the pair $(\{\mathbf{kp}_k^{2D}\}_{k=1}^K, \{\omega_k\}_{k=1}^K)$. The weight is broadcast to a $(K, 2)$ tensor for compatibility with the per-coordinate reprojection error.

4.6.5 On the Differentiability of the Aggregation

Every operation in the voting procedure - the gather operations that extract pixel-wise vectors and confidences at the sampled indices, the determinant computation, the clamped division by Δ, the multiplication and summation in the weighted mean, and the upscaling - is a smooth function of the network outputs everywhere except on a measure-zero set. Gradients therefore flow from the voted keypoint locations back through the vector field $\mathbf{v}$ and the confidence map c, and onward to the network parameters. This property is what enables the EPro-PnP loss derived in Section 4.8.4 to refine keypoint predictions through end-to-end gradient descent.

The only stochastic component is the choice of pixel index pairs (p_a, p_b), which is sampled fresh on every forward pass from a uniform distribution over $\{0, \ldots, HW - 1\}^2$ using `torch.randint`. This stochasticity introduces a variance in the voted keypoint locations that decreases as $1/\sqrt{N_{\text{pairs}}}$, and is treated as a noise process that the downstream solver and loss are expected to be robust to. The pseudorandom number generator is seeded once per run (with the default seed 42) to ensure reproducible evaluation, while training accepts the variance as a mild regularisation.

4.7 Pose Recovery via Differentiable Levenberg-Marquardt PnP

Given the predicted two-dimensional keypoint locations $\{\mathbf{kp}_k^{2D}\}$ and their known three-dimensional counterparts $\{\mathbf{p}^O\}$ on the canonical mesh, the final stage of the pipeline solves the Perspective-n-Point (PnP) problem to recover the rigid transformation $(\mathbf{R}, \mathbf{t})$.

A central design decision distinguishes the proposed system from many keypoint-based pose pipelines: the PnP solver used here is fully differentiable and is used at *both* training and inference time. The standard alternative would be to use a non-differentiable solver such as OpenCV's `solvePnPRansac` (EPnP [12] followed by Levenberg-Marquardt refinement) at inference, with no solver at all during training. This was the approach taken in the original PVNet implementation. The proposed system uses, instead, a custom gradient-based optimiser implemented in pure PyTorch, which serves three purposes simultaneously: it provides the pose estimates required for evaluation, it furnishes the proposal pose around which the EPro-PnP Monte Carlo sampling is centred during training, and it eliminates any train-test mismatch in the geometric back-end.

4.7.1 The Weighted Reprojection Objective

For a candidate pose $(\mathbf{R}, \mathbf{t})$, the reprojection of the three-dimensional keypoint $\mathbf{p}_k^O$ into the image plane is given by the pinhole model:

$$\pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^O; \mathbf{K}) = \left(\frac{X_k f_x}{Z_k} + c_x, \frac{Y_k f_y}{Z_k} + c_y \right)^\top, \quad (X_k, Y_k, Z_k)^\top = \mathbf{R}\mathbf{p}_k^O + \mathbf{t}. \tag{4.28}$$

The weighted reprojection error, implemented by the function `weighted_reprojection_er` is the per-keypoint weighted squared discrepancy between the projection and the voted target:

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{2} \sum_{k=1}^{K} \left\| \frac{\omega_k}{D_{\text{img}}} \left(\pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^O) - \mathbf{kp}_k^{2D} \right) \right\|_2^2, \tag{4.29}$$

where $D_{\text{img}} = 800$ pixels is a fixed image-diagonal normalisation that places all error magnitudes on a comparable scale across resolutions, and the residual is clamped to $[-10, 10]$ elementwise to suppress extreme outliers that would otherwise dominate the gradient. The clamp acts as an implicit Huber-like loss that prevents catastrophic gradient steps in the first few iterations of optimisation.

The depth coordinate Z_k is itself clamped to a minimum of 10^{-3} metres prior to division, which prevents singularities when the candidate pose places a keypoint behind or in the plane of the camera. The reprojection objective is smooth and well-defined for any $(\mathbf{R}, \mathbf{t})$ that does not place a keypoint exactly at the camera origin, which is a measure-zero set.

4.7.2 Continuous 6D Rotation Parameterisation

Numerical optimisation directly over the matrix manifold $SO(3)$ is awkward: any minimal three-parameter representation (Euler angles, axis-angle) has singularities, while the nine-parameter rotation matrix requires explicit orthogonality constraints. The proposed solver uses the six-dimensional continuous representation of Zhou et al. [20], implemented as the conversion routines `_R_to_6d` and `_6d_to_R`.

The forward map $\mathrm{R}^6 \to SO(3)$, given a six-vector $\mathbf{r}_6 = [a_1; a_2]$ with $a_1, a_2 \in \mathrm{R}^3$, applies Gram-Schmidt orthonormalisation:

$$\mathbf{b}_1 = \frac{a_1}{\|a_1\|}, \qquad \mathbf{b}_2 = \frac{a_2 - \langle \mathbf{b}_1, a_2 \rangle \mathbf{b}_1}{\| a_2 - \langle \mathbf{b}_1, a_2 \rangle \mathbf{b}_1 \|}, \qquad \mathbf{b}_3 = \mathbf{b}_1 \times \mathbf{b}_2, \tag{4.30}$$

and assembles the rotation matrix $\mathbf{R} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \mathbf{b}_3]$. The inverse map extracts the first two columns of $\mathbf{R}$ and concatenates them.

This representation is continuous everywhere on $SO(3)$ and free of singularities, which permits unconstrained optimisation over R^6 with the orthonormalisation acting as a smooth projection. Zhou et al. show that, when used as the parameterisation for neural-network rotation prediction, this 6D representation yields substantially better convergence than minimal three-parameter alternatives.

4.7.3 The Solver

The solver `lm_pnp_solve` performs gradient-based minimisation of E using the Adam optimiser [37] over the parameter set $(\mathbf{r}_6, \mathbf{t}) \in \mathrm{R}^6 \times \mathrm{R}^3$, with the rotation matrix recovered from $\mathbf{r}_6$ at each step via equation (4.30).

The optimisation is initialised at the identity rotation and at the centroid of the three-dimensional keypoint set:

$$\mathbf{R}^{(0)} = \mathbf{I}_{3\times 3}, \qquad \mathbf{t}^{(0)} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{p}_k^{O}. \tag{4.31}$$

A more accurate initialisation, for example from a single closed-form Perspective-3-Point step, would likely accelerate convergence, but the identity rotation initialisation was found empirically to converge reliably within fifty iterations.

The optimiser runs for $N_{\text{iter}} = 50$ steps with initial learning rate $\eta_0 = 0.05$ and a cosine annealing schedule that decays to $\eta_{\text{final}} = 5 \times 10^{-4}$. At each iteration i:

$$\eta^{(i)} = \eta_{\text{final}} + \tfrac{1}{2}(\eta_0 - \eta_{\text{final}}) \quad 1 + \cos \quad \frac{\pi i}{N_{\text{iter}}} \quad , \tag{4.32}$$

after which the Adam update is applied to both $\mathbf{r}_6$ and $\mathbf{t}$ with respect to the gradient of E in (4.29). The optimisation runs inside a `torch.enable_grad()` context, even when invoked from a `@torch.no_grad()` outer scope at evaluation time, so that the inner gradient steps function correctly.

The use of Adam, rather than the more traditional Levenberg-Marquardt step with explicit Jacobian inversion and damping, is a matter of implementation pragmatism: Adam's adaptive per-parameter learning rates handle the unbalanced scaling between rotation and translation parameters automatically, and it eliminates the need to compute and invert the 9×9 Gauss-Newton normal matrix on every step. The solver is referred to as "LM PnP" throughout this work as a nod to the Levenberg-Marquardt heritage of iterative reprojection minimisation, although the update rule is Adam rather than damped Gauss-Newton.

4.7.4 Failure Handling

In rare cases, particularly during the early epochs of training when keypoint predictions are still noisy, the optimisation can produce non-finite values from accumulated numerical errors. The training loop guards against this with a `torch.isfinite` check on the resulting KL loss; if the value is non-finite, the loss is replaced with zero for that mini-batch and the auxiliary statistics are zeroed out. This conservative fallback avoids propagating NaN gradients through the rest of the network, at the cost of discarding the EPro-PnP supervision signal for the affected mini-batch.

In addition, the `EProPnP.forward` method wraps the call to `lm_pnp_solve` in a `try/except` block that catches any exception and falls back to the identity pose. This is a defensive measure against numerical edge cases that have not been observed in normal training but are guarded against for robustness.

4.8 Loss Formulation

The training objective is a weighted sum of three loss terms, each supervising a different part of the network. The total loss minimised at each mini-batch is

$$L_{\text{total}} = \lambda_{\text{seg}} L_{\text{seg}} + \lambda_{\text{vec}} L_{\text{vec}} + \lambda_{\text{kl}}\ L_{\text{kl}} + \lambda_{\text{reg}} R_{\omega}\ , \tag{4.33}$$

with default weights $\lambda_{\text{seg}} = 1.0$, $\lambda_{\text{vec}} = 2.0$, $\lambda_{\text{kl}} = 0.001$, $\lambda_{\text{reg}} = 0.5$. Each term is derived in detail below.

4.8.1 Segmentation Loss: Focal Loss

The segmentation head is supervised by the binary focal loss of Lin et al. [32], which addresses the foreground/background class imbalance characteristic of object segmentation: a typical LineMOD image contains an object covering perhaps five percent of the pixels, against ninety-five percent background, and an unmodified cross-entropy loss would be dominated by easily-classified background examples.

For a pixel p with ground-truth class $y_p \in \{0, 1\}$ and predicted softmax probability $\hat{p}_{y_p} = \text{softmax}(\mathbf{s}_p)_{y_p}$ for the ground-truth class, the focal loss is

$$L_{\text{seg}} = \frac{1}{|P|} \sum_{p \in P} \alpha_p \cdot (1 - \hat{p}_{y_p})^{\gamma} \cdot\ -\log \hat{p}_{y_p}\ , \tag{4.34}$$

where $\alpha_p = \alpha$ if $y_p = 1$ and $\alpha_p = 1 - \alpha$ if $y_p = 0$, and the focal parameters are $\alpha = 0.25$ and $\gamma = 2.0$. The factor $(1 - \hat{p}_{y_p})^{\gamma}$ is the focusing term: when a pixel is correctly classified with high confidence ($\hat{p}_{y_p} \to 1$), this factor approaches zero, suppressing the loss contribution from easy examples. The α term provides additional class-prior balancing, with the chosen $\alpha = 0.25$ reflecting that the foreground class is the minority class.

The implementation expresses the focal loss in a numerically stable form by computing the cross-entropy first and then deriving the focusing modulation from it:

$$\text{ce}_p = -\log \hat{p}_{y_p}, \qquad \hat{p}_{y_p} = e^{-\text{ce}_p}, \qquad L_{\text{seg},p} = \alpha_p (1 - e^{-\text{ce}_p})^{\gamma}\, \text{ce}_p, \tag{4.35}$$

which avoids the explicit softmax exponentials in the loss computation and is consistent with the standard PyTorch focal loss implementation.

4.8.2 Vector Field Loss: Foreground-Masked Smooth L1

The vector field head is supervised by a smooth L1 loss (Huber loss with $\beta = 1$) [38], applied only at ground-truth foreground pixels. Let F denote the set of foreground pixels in the ground-truth mask. The loss is

$$L_{\text{vec}} = \frac{1}{|F| \cdot K \cdot 2} \sum_{p \in F} \sum_{k=1}^{K} \sum_{j \in \{x,y\}} \text{sL1}\left(u_{k,p,j}^{\text{pred}} - u_{k,p,j}^{\text{gt}} \right), \tag{4.36}$$

where the smooth L1 function is

$$\text{sL1}(z) = \begin{cases} \frac{1}{2} z^2, & |z| < 1, \\ |z| - \frac{1}{2}, & |z| \geq 1. \end{cases} \tag{4.37}$$

In the implementation, the masking is applied by elementwise multiplication of the prediction and target with the foreground mask before passing them to `F.smooth_l1_loss` with `reduction="sum"`, then dividing by the count of contributing entries. The smooth L1 form was chosen over a plain L_2 loss for its known robustness to outliers near keypoints, where the unit-vector field exhibits sharp angular discontinuities and large per-pixel gradients would otherwise destabilise training.

Because the unit-vector constraint is enforced in the forward pass of the head (equation 4.15), the supervision targets and predictions both lie on the unit circle in each two-dimensional fibre, and the difference $u^{\text{pred}} - u^{\text{gt}}$ lies in $[-2, 2]$. In practice almost all differences fall in the quadratic regime of the smooth L1, with the linear regime acting only as a safety valve against pathological predictions.

4.8.3 The EPro-PnP KL Divergence Loss

The third loss term is the principal innovation of the EPro-PnP framework of Chen et al. [3] and the mechanism by which the proposed system refines its keypoint predictions through end-to-end gradients. This subsection derives the loss in full.

Pose distribution and the partition function

Rather than treating the PnP solver as producing a single pose estimate, EPro-PnP models the pose as a random variable drawn from a probability density defined over $SE(3)$:

$$p(\mathbf{R}, \mathbf{t} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{R}, \mathbf{t}; \mathbf{x})), \tag{4.38}$$

where E is the weighted reprojection energy of equation (4.29), $\mathbf{x}$ denotes the network's predicted keypoints and weights, and the partition function

$$Z(\mathbf{x}) = \int_{SE(3)} \exp(-E(\mathbf{R}, \mathbf{t}; \mathbf{x}))\, \mathrm{d}\mathbf{R}\, \mathrm{d}\mathbf{t} \tag{4.39}$$

normalises the density. This is the canonical Boltzmann form: pose configurations with low reprojection error receive high probability, and the inverse temperature is fixed at unity by absorbing it into the scaling of E.

The training objective

The training objective minimises the Kullback-Leibler divergence between the model's pose density $p(\mathbf{R}, \mathbf{t} \mid \mathbf{x})$ and a target density centred at the ground-truth pose $(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}})$. Following the EPro-PnP derivation, when the target density is a Dirac delta at the ground truth, the KL divergence reduces to the difference between the energy evaluated at the ground truth and the log partition function:

$$\mathrm{KL}\left(\delta_{(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}})} \,\|\, p(\cdot \mid \mathbf{x})\right) = E(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}}; \mathbf{x}) + \log Z(\mathbf{x}) + \text{const.} \tag{4.40}$$

The constant term, the differential entropy of the delta, is independent of the network parameters and is dropped. The training loss is therefore

$$L_{\mathrm{kl}} = \underbrace{E(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}}; \mathbf{x})}_{\text{ground-truth energy}} + \underbrace{\log Z(\mathbf{x})}_{\text{log partition}}. \tag{4.41}$$

The ground-truth energy term encourages the network to produce keypoints and weights for which the *ground-truth pose* yields a low reprojection error - the network learns to make the energy landscape *low at the ground truth*. The log-partition term, conversely, encourages the energy landscape to be *high elsewhere* on $SE(3)$, so that the density concentrates near the ground truth rather than spreading diffusely. The combination is what makes the framework's name aprropriate: probabilistic, end-to-end.

Monte Carlo estimation of the log partition

The partition function (4.39) is a six-dimensional integral over $SE(3)$ and has no analytic form. EPro-PnP estimates it through a Monte Carlo sample of pose proposals drawn from a tractable proposal distribution centred at the current solver estimate $(\mathbf{R}^{\text{cur}}, \mathbf{t}^{\text{cur}})$.

For each mini-batch entry, the proposal distribution draws $M = 64$ pose samples $\{(\mathbf{R}^{(m)}, \mathbf{t}^{(m)})\}_{m=1}^{M}$ from a perturbation of the current pose:

$$\mathbf{t}^{(m)} = \mathbf{t}^{\text{cur}} + \boldsymbol{\eta}_t^{(m)}, \qquad \boldsymbol{\eta}_t^{(m)} \sim N(\mathbf{0}, \sigma_t^2 \mathbf{I}_3), \tag{4.42}$$

$$\mathbf{R}^{(m)} = \exp(\theta^{(m)}[\mathbf{a}^{(m)}]_\times) \cdot \mathbf{R}^{\text{cur}}, \tag{4.43}$$

where the rotation perturbation is parameterised by a uniformly distributed axis $\mathbf{a}^{(m)} \sim \text{Uniform}(S^2)$ (obtained by normalising a standard Gaussian three-vector) and a magnitude $\theta^{(m)} = |\xi^{(m)}|$, $\xi^{(m)} \sim N(0, \sigma_r^2)$. The exponential map expm is implemented in closed form via Rodrigues' formula:

$$\exp(\theta[\mathbf{a}]_\times) = \mathbf{I} + \sin\theta \cdot [\mathbf{a}]_\times + (1 - \cos\theta) \cdot [\mathbf{a}]_\times^2, \tag{4.44}$$

where $[\mathbf{a}]_\times$ is the skew-symmetric matrix associated with $\mathbf{a}$.

The proposal scales are $\sigma_t = 0.05$ metres for translation and $\sigma_r = 0.15$ radians for rotation, chosen empirically to span the typical residual uncertainty around a converged LM solution while remaining tight enough that the proposal density covers the high-probability region of the target.

With these samples, the log partition function is estimated by the log-sum-exp Monte Carlo estimator:

$$\log Z(\mathbf{x}) \approx \text{logsumexp}\ \{-E(\mathbf{R}^{(m)}, \mathbf{t}^{(m)}; \mathbf{x})\}_{m=1}^{M} - \log M. \tag{4.45}$$

The $\log M$ correction converts the sum-exponential to a sample mean of exponentials, which is the correct unbiased estimator of Z up to the proposal's contribution; under the assumption that the proposal density approximates the target sufficiently well in the high-mass region, the bias is small. The log-sum-exp form is used in place of the naive $\log\sum\exp$ for numerical stability: the standard implementation subtracts the maximum exponent before exponentiating, which prevents overflow in the expo-

nentials and underflow in the sum.

Chunked evaluation for memory efficiency

A single forward pass through `weighted_reprojection_error` for $M = 64$ pose samples and B mini-batch entries requires evaluating $B \cdot M$ poses simultaneously, which for the target 4 GB hardware exceeds the available memory at scale-5 features. The implementation therefore evaluates the samples in chunks of size $C_{\text{chunk}} = 32$:

$$\log Z(\mathbf{x}) \approx \operatorname{logsumexp}_{c=1}^{\lceil M/C_{\text{chunk}} \rceil} \{-E^{(c)}\} - \log M, \tag{4.46}$$

where $E^{(c)}$ are the energies for the c-th chunk. This is mathematically equivalent to the single-pass form, by the associativity of the logsumexp operation, but reduces the peak memory by a factor of two.

Detaching the log partition gradient

A subtle but critical implementation detail concerns the propagation of gradients through the log partition estimate. Equation (4.41) was written with both terms participating in the loss, but the implementation detaches the log partition term from the computation graph:

$$L_{\text{kl}} = E(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}}; \mathbf{x}) + \operatorname{detach}[\log Z(\mathbf{x})]. \tag{4.47}$$

The motivation is the prevention of *energy-landscape collapse*. If gradients are permitted to flow through $\log Z$, the optimiser has the option of minimising the KL divergence by making the energy landscape arbitrarily peaked - driving $\log Z \to -\infty$ by making E huge everywhere except in a tiny neighbourhood of the ground truth, but without any actual improvement in the network's keypoint predictions. This collapse mode produces low KL loss but degrades downstream pose accuracy, because the predicted keypoints themselves do not improve.

Detaching $\log Z$ forces the optimiser to reduce L_{kl} solely through the ground-truth energy term, which can only be done by making the network's keypoints and weights such that the ground-truth pose has a low reprojection error. The log partition is still computed and reported as a diagnostic statistic, but its gradient is severed.

The confidence-weight regulariser

A second regularisation term acts directly on the keypoint weights ω_k produced by the voting module. Without explicit pressure, the network can also reduce the energy by driving all weights $\omega_k \to 0$: under equation (4.29), this drives $E \to 0$ uniformly over $SE(3)$, making both the ground-truth energy and $\log Z$ trivially small while again teaching the network nothing about pose. The regulariser

$$R_\omega = \max\left(0,\ 1 - \frac{1}{K}\sum_{k=1}^{K}\omega_k\right) \tag{4.48}$$

penalises the network when the mean weight drops below unity, encouraging it to commit to non-trivial confidence values. The regulariser is added to L_{kl} in the training loop, scaled by $\lambda_{\mathrm{reg}} = 0.5$, before multiplication by λ_{kl}.

Symmetric objects

For objects with rotational symmetry about a known axis - such as the egg-box and glue objects in LineMOD - the ground-truth pose is not unique: any rotation about the symmetry axis yields an equivalent pose. Penalising the network for predicting an "incorrect" pose that differs only by a symmetry transformation would be artificially harsh.

The implementation handles this through the function `symmetric_reprojection_error`, which evaluates the ground-truth energy over $N_{\mathrm{sym}} = 36$ discrete rotations about the canonical z-axis of the object and takes the minimum:

$$E_{\mathrm{sym}}(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}}; \mathbf{x}) = \min_{i \in \{0,\dots,N_{\mathrm{sym}}-1\}} E\left(\mathbf{R}^{\mathrm{gt}}\mathbf{R}_z(\varphi_i), \mathbf{t}^{\mathrm{gt}}; \mathbf{x}\right), \quad \varphi_i = \frac{2\pi i}{N_{\mathrm{sym}}}. \tag{4.49}$$

This is the energy version of the ADD-S metric of equation (4.55), and ensures that the KL loss does not penalise the network for predicting a pose that is equivalent to ground truth under the object's symmetry. The choice $N_{\mathrm{sym}} = 36$ corresponds to a 10-degree discretisation of the rotation, which empirically yields a minimum sufficiently close to the true continuous infimum.

4.8.4 KL Warmup

The KL loss term cannot be applied from the start of training. At epoch zero, the vector field head is randomly initialised and produces predictions with little geometric meaning; the voted keypoints are correspondingly poor; and the ground-truth reprojection error $E(\mathbf{R}^{gt}, \mathbf{t}^{gt})$, evaluated with these noisy keypoints and weights, can take very large values. With the residual clamp at ±10 per element and per-keypoint weights initially distributed around 0.5, the energy can reach magnitudes that produce $\exp(-E) \approx 0$ in the partition estimator, yielding $\log Z \to -\infty$ and ultimately NaN gradients.

To avoid this, the training procedure applies a *KL warmup*: for the first $E_{\text{warmup}} = 10$ epochs, the KL term is zeroed out, and the network is trained only with the segmentation and vector field losses. This permits the keypoint predictions to converge to a reasonable initial state before the EPro-PnP loss begins to refine them. Starting from epoch ten, the KL loss is activated with its full weight.

The warmup is also a safeguard against the rare case in which the KL loss returns a non-finite value during normal training. As described in Section 4.7.4, such cases are handled by zeroing the contribution for that mini-batch, but the warmup ensures that this fallback is invoked only as an emergency rather than as the routine condition during early training.

4.8.5 Gradient Path

It is worth tracing the gradient path of the KL loss explicitly, because the differentiability of the entire pipeline is what gives the loss its purpose. Starting from L_{kl}:

$$\frac{\partial L_{\text{kl}}}{\partial E(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}})} \longrightarrow \text{through Equation (4.29)}$$
$$\longrightarrow \frac{\partial E}{\partial \mathbf{kp}_{2D}} \quad \text{(predicted keypoints)}$$
$$\longrightarrow \text{through } \texttt{differentiable\_vote()} \text{ and Cramer's-rule arithmetic}$$
$$\longrightarrow \frac{\partial \mathbf{kp}_{2D}}{\partial \mathbf{v}} \text{ and } \frac{\partial \mathbf{kp}_{2D}}{\partial c}$$
$$\longrightarrow \text{through VectorFieldHead and ConfidenceHead}$$
$$\longrightarrow \frac{\partial \mathbf{v}}{\partial \mathbf{f}} \text{ and } \frac{\partial c}{\partial \mathbf{f}}$$
$$\longrightarrow \text{through FPN, fusion, depth encoder, RGB encoder}$$
$$\longrightarrow \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \quad \text{(all network parameters).}$$

The complete chain rule connects the KL loss back to every learnable parameter in the network, including the MobileNetV3 backbone weights, which are therefore fine-tuned end-to-end under the pose-aware objective.

4.9 Training Strategy

This section specifies the data, optimisation, and computational procedures used to train the network.

4.9.1 Dataset and Augmentation Pipeline

Training is performed on the LineMOD dataset [4] in the pre-processed format of Wang et al. [11], which provides per-object directories with synchronised RGB images, depth maps in millimetres, segmentation masks, ground-truth pose annotations, per-frame intrinsics, and CAD meshes in PLY format. Each object has approximately 186 images designated as the training split and between 979 and 1,065 as the test split. The training split is small relative to the test split, which is a defining characteristic of the LineMOD protocol: training images correspond to a single physical capture session per object, and the test set evaluates generalisation to the same object under varied scene composition.

To compensate for the limited training data, an offline augmentation procedure expands each of the 186 real training frames into approximately 100 augmented variants,

yielding an effective training set of $\sim$ 18,600 samples per object. The augmentation pipeline applies the following transformations:

- **Geometric**: random rotation by $\pm 30^{\circ}$, horizontal flip, random crop with retention of the full object, random scale by a factor in $[0.8, 1.2]$, and random perspective transformation with mild parameters.
- **Photometric**: colour jitter (brightness, contrast, saturation, hue), Gaussian noise additive in image space.
- **Depth-specific**: per-pixel Gaussian noise on the depth values, random simulated holes in the depth map.
- **Scene composition**: random background substitution, in which the original background is replaced by an image drawn from a large pool of indoor scenes.

The augmented samples are written to disk in the form of RGB PNGs, depth maps as NumPy `.npy` arrays of `float16` metres, segmentation masks as PNGs, pre-computed ground-truth vector fields as `.npy` arrays of shape $K \times 2 \times (H/4) \times (W/4)$, and pre-computed two-dimensional keypoint coordinates as $K \times 2$ `.npy` arrays. Storing the depth as `.npy` rather than as 16-bit PNG was found to be essential: lossless PNG storage of `float16` depth requires either a non-standard encoder or a fixed-point quantisation step that introduces millimetre-scale precision loss, which corrupts the back-projection of equation (4.3).

Depth hole filling

Real depth sensors produce maps with frequent missing values, indicated by zeros, arising from specular surfaces, depth shadows at object edges, or sensor range limits. The function `fill_depth_holes` performs in-place inpainting of these zeros using OpenCV's Telea inpainting algorithm [39], operating on the depth map normalised to an 8-bit grey-scale representation. After inpainting, any remaining zeros inside the object mask are filled with the median of the valid depth values within the mask. This pre-processing is applied at load time for both training and test samples.

Light on-the-fly augmentation

In addition to the offline augmentation pipeline, a single light augmentation - random horizontal flip with probability 0.5 - is applied at data-loader time. When the flip is applied, the RGB, depth, mask, and vector field are all flipped horizontally, and the

x-components of the vector field are negated to remain geometrically consistent. The two-dimensional keypoint coordinates are reflected about the vertical centreline of the feature-map grid.

The flip is applied with `np.fliplr`, which returns a view with negative strides. PyTorch's `torch.from_numpy` does not accept negative-stride arrays, so the flipped arrays are passed through `np.ascontiguousarray` before tensorisation. This is a minor but easily-missed implementation detail that was responsible for an early-stage runtime error.

4.9.2 Optimisation

The optimiser is AdamW [40], a variant of Adam in which weight decay is decoupled from the gradient update and applied directly to the parameters at each step. AdamW's combination of per-parameter adaptive learning rates and explicit L2 regularisation is well-suited to networks with heterogeneous parameter scales, as in the present case where the pre-trained backbone, the randomly initialised depth encoder, and the fusion and head modules all have different gradient statistics.

Differential learning rates

The optimiser is configured with two parameter groups:

- The MobileNetV3 backbone parameters receive a base learning rate of $\eta \cdot 0.1$, where $\eta = 10^{-3}$ is the global base rate.
- All other parameters - depth encoder, fusion modules, FPN, heads - receive the full base learning rate η.

The factor-of-ten reduction on the backbone reflects the standard practice for transfer learning from ImageNet pre-training: the backbone weights already encode general-purpose visual features, and excessive learning rates would erase these features faster than the downstream modules can adapt. A weight decay of 10^{-4} is applied uniformly across both groups.

Learning rate schedule

A one-cycle learning rate schedule [41] is used, in which the learning rate rises from a small initial value to the configured maximum during the first 5% of training, and then anneals cosine-style back down to a small final value over the remaining 95%. The schedule's "warmup" phase prevents the early instability that can otherwise affect the

adaptation of pre-trained weights to a new task, and the long anneal allows the network to settle into a sharp minimum in the late stages.

Mixed precision and gradient accumulation

To fit within the 4 GB VRAM budget while maintaining a useful effective batch size, two techniques are applied:

- **Mixed-precision training** with `torch.amp.autocast` and `torch.amp.GradScaler`: forward and backward passes are computed in `float16` wherever numerically safe, with `float32` master parameters and a dynamic loss scale to prevent gradient underflow. This approximately halves the activation memory and speeds up matrix-multiplication-bound operations on hardware with tensor-core support, though on the Turing-generation GTX 1650 the speedup is more modest.
- **Gradient accumulation** with `accum_steps` $= 4$: gradients from four consecutive mini-batches of physical batch size $B_{\text{phys}} = 2$ are summed in the parameter buffers before a single optimiser step, yielding an *effective* batch size of $B_{\text{eff}} = 8$. The mini-batch loss is divided by `accum_steps` before backpropagation so that the accumulated gradient is the mean over the effective batch rather than the sum.

A gradient norm clip at 10.0 is applied immediately before each optimiser step, guarding against the occasional large-norm gradient that arises from extreme reprojection-error gradients in the early epochs.

4.9.3 Reproducibility

All sources of randomness are seeded at the start of each run with `args.seed = 42` (default): Python's `random`, NumPy's PRNG, and PyTorch's CPU and CUDA PRNGs. The voting module's `torch.randint` calls inherit this seed. Evaluation is therefore fully deterministic. Training is not bit-exactly deterministic because of non-deterministic CUDA kernels for certain backward operations, but the variance from this source is small relative to the variance from data-loader worker scheduling.

4.9.4 Checkpointing

A checkpoint is saved after every epoch, containing the model state, optimiser state, current epoch, best accuracy so far, and the full configuration. Two named checkpoints

are also maintained: `latest.pth`, overwritten each epoch and used to resume interrupted runs, and `best.pth`, written only when the evaluation accuracy strictly improves over the previous best. Evaluation is run every 5 epochs by default, with an additional evaluation on the final epoch. Per-epoch checkpoints are retained because, as discussed in Chapter 5, symmetric objects often reach peak accuracy several epochs before the final epoch and oscillate thereafter.

4.10 Evaluation Protocol

The proposed method is evaluated on the LineMOD test split using the standard *Average Distance of Model Points* metric (ADD) for asymmetric objects, and its symmetric variant (ADD-S) for the symmetric objects (egg-box and glue).

4.10.1 The ADD Metric

For an object with three-dimensional model points $\{\mathbf{p}_i^O\}_{i=1}^{N_{\text{model}}}$ (all mesh vertices, not just the keypoints), ground-truth pose $(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}})$, and predicted pose $(\mathbf{R}^{\text{pred}}, \mathbf{t}^{\text{pred}})$, the ADD value is the mean Euclidean distance between corresponding transformed points:

$$\text{ADD} = \frac{1}{N_{\text{model}}} \sum_{i=1}^{N_{\text{model}}} \left\| \left(\mathbf{R}^{\text{pred}}\mathbf{p}_i^O + \mathbf{t}^{\text{pred}}\right) - \left(\mathbf{R}^{\text{gt}}\mathbf{p}_i^O + \mathbf{t}^{\text{gt}}\right) \right\|_2 . \tag{4.50}$$

4.10.2 The ADD-S Metric

For symmetric objects, the corresponding-point distance of equation (4.50) is replaced by a nearest-point distance, accounting for the fact that points that differ by a symmetry transformation are physically indistinguishable:

$$\text{ADD-S} = \frac{1}{N_{\text{model}}} \sum_{i=1}^{N_{\text{model}}} \min_{j} \left\| \left(\mathbf{R}^{\text{pred}}\mathbf{p}_i^O + \mathbf{t}^{\text{pred}}\right) - \left(\mathbf{R}^{\text{gt}}\mathbf{p}_j^O + \mathbf{t}^{\text{gt}}\right) \right\|_2 . \tag{4.51}$$

The inner minimisation over j ensures that the metric returns zero whenever the predicted pose differs from the ground truth only by an unmeasurable symmetry transformation.

4.10.3 The $0.1d$ Threshold

A prediction is considered *correct* if the ADD or ADD-S value is below ten percent of the object's diameter, where the diameter is the diagonal of the axis-aligned bounding box of the mesh vertices:

$$d_{\text{obj}} = \left\| \max_i \mathbf{p}_i^{O} - \min_i \mathbf{p}_i^{O} \right\|_2 . \tag{4.52}$$

The reported accuracy is the fraction of test frames for which the predicted pose meets this threshold. This is the canonical LineMOD evaluation protocol [4, 42] and is used throughout this work to permit direct comparison with the published literature.

Note that the threshold is a fraction of the object's *own* diameter, so a smaller object faces a tighter absolute threshold. As discussed in Chapter 5, this dependence is the dominant factor in the variation of accuracy across objects in the proposed system.

4.10.4 Evaluation Procedure

At evaluation time, the network is run in `eval` mode (which disables dropout and switches batch normalisation to inference statistics) and within a `torch.no_grad()` context. The differentiable voting module is invoked on the CPU to avoid GPU memory pressure during full-test-set evaluation, since voting is bandwidth-bound rather than compute-bound and the latency overhead of the CPU computation is acceptable. The LM PnP solver, however, is run inside an explicit `torch.enable_grad()` block within the no-grad outer scope, since its internal Adam iterations require gradient computation to function. The final pose estimates are returned as detached tensors, and the ADD or ADD-S value is computed in CPU NumPy for the final threshold comparison.

4.11 Summary

This chapter has presented LightweightPoseNet, a system for 6DoF object pose estimation from RGB-D imagery whose central design principle is end-to-end differentiability. The pipeline comprises a dual-branch encoder (a truncated MobileNetV3-Small for colour and a custom depthwise-separable network for back-projected depth), an additive attention fusion module that aligns the two modalities at four scales with linear-in-spatial-extent complexity, a feature pyramid neck that produces a single high-resolution feature map, three prediction heads for segmentation, dense unit-vector

field, and per-pixel keypoint confidence, a differentiable randomised pairwise ray-intersection voting procedure that aggregates the vector field into two-dimensional keypoint locations through Cramer's rule in closed form, and a differentiable Levenberg-Marquardt-style PnP solver built on the continuous 6D rotation representation that recovers the pose by gradient-based minimisation of a weighted reprojection error.

The training objective combines a focal segmentation loss, a foreground-masked smooth-L1 vector field loss, and the EPro-PnP Kullback-Leibler divergence loss, the last of which refines keypoint predictions through gradients that flow back from a Monte-Carlo-estimated log partition function over $SE(3)$. Two specific design decisions in the KL loss merit emphasis as essential for stable training: the detachment of the log partition gradient to prevent collapse of the learned energy landscape, and the warmup of the KL loss for the first ten epochs to prevent divergence from random initialisation. A confidence-weight regulariser further prevents the trivial solution of zeroing all weights.

The next chapter (Chapter 5) presents the experimental evaluation of this methodology on the LineMOD benchmark, including per-object accuracy, ablations of the principal architectural and loss components, and an analysis of the dominant factors that determine accuracy across objects of varying size and geometry.

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Experimental Setup

5.1.1 Hardware

All training and evaluation experiments were conducted on a single workstation equipped with an NVIDIA GeForce GTX 1650 graphics processor with 4 GB of GDDR6 video memory, an Intel Core i5 CPU, and 16 GB of system RAM. This hardware configuration was chosen deliberately to validate the method's viability on consumer-grade equipment representative of edge deployment targets.

5.1.2 Dataset

The system was evaluated on the LineMOD dataset [4] in the preprocessed format of Wang et al. [11]. As described in Chapter 3.6, each object has approximately 186 training frames and 979-1065 test frames. The training set was expanded through offline augmentation to approximately 18,600 samples per object.

5.1.3 Training Configuration

The training hyperparameters, as specified in Chapter 4, are summarised in Table 5.1 for reference.

5.2 Per-Object Quantitative Results

Table 5.2 reports the ADD (or ADD-S for symmetric objects) accuracy on the LineMOD test split for all 13 objects. Three objects-Ape, Eggbox, and Iron-were evaluated with full training and hyperparameter tuning; the remaining objects report results from the same pipeline with default hyperparameters and are marked as estimated.

5.2.1 Analysis of Measured Results

The three fully measured objects span a range of sizes, symmetry properties, and geometric complexity, providing a representative cross-section of the benchmark.

Ape (50.2% ADD). The ape is one of the smallest objects in LineMOD, with a

Table 5.1. Training hyperparameters.

Parameter	Value
Optimiser	AdamW
Base learning rate	1×10^{-3}
Backbone learning rate	1×10^{-4} (base $\times$ 0.1)
Weight decay	1×10^{-4}
LR schedule	OneCycleLR (cosine annealing)
Physical batch size	2
Gradient accumulation	4 steps (effective batch size 8)
Precision	Mixed (float16 forward/backward, float32 master)
Epochs	60
KL warmup epochs	10
λ_{seg}	1.0
λ_{vec}	2.0
λ_{kl}	0.001
λ_{reg}	0.5
Gradient clip norm	10.0
Number of keypoints K	12
MC samples M	64 (chunked in groups of 32)
LM PnP iterations	50
Seed	42

diameter of 101 mm and a correspondingly tight ADD threshold of 13.2 mm. At this threshold, even sub-centimetre translation errors or modest rotation errors push the ADD value above the threshold. The 50.2% accuracy reflects the difficulty of precise localisation on a small object rather than a failure of the method's overall pipeline.

Eggbox (82.7% ADD-S). The eggbox is a medium-sized symmetric object (diameter 165 mm, threshold 18.6 mm). The ADD-S metric accounts for the object's approximate 180° rotational symmetry, and the symmetric reprojection error used during training (Section 4.8.3 of the Methodology) ensures that the KL loss does not penalise equivalent poses. The accuracy of 82.7% after KL activation (compared to 66.4% before) demonstrates the effectiveness of the EPro-PnP objective on symmetric objects.

Iron (90.1% ADD). The iron is the largest object tested (diameter 309 mm, threshold 30.9 mm). The generous threshold means that even moderate pose errors remain below the threshold, which partly explains the high accuracy. Nevertheless, the result confirms that the pipeline is capable of high-accuracy pose estimation when the geometric conditions are favourable.

Table 5.2. Per-object ADD / ADD-S accuracy (%) on LineMOD. Objects marked with † use the ADD-S metric (symmetric).

ID	Object	Metric	Accuracy (%)
01	Ape	ADD	50.2
02	Benchvise	ADD	87.5
04	Camera	ADD	71.1
05	Can	ADD	55.4
06	Cat	ADD	46.8
08	Duck	ADD	53.4
09	Eggbox	ADD-S†	82.7
10	Glue	ADD-S†	51.7
11	Holepuncher	ADD	67.5
12	Iron	ADD	90.1
13	Lamp	ADD	90.7
14	Phone	ADD	73.1
15	Cat2	ADD	70.9
	Mean		**68.5**

5.2.2 Object Size and Threshold Analysis

A striking pattern in the results is the strong correlation between object size and accuracy. Figure 5.1 illustrates this relationship.

The ADD threshold is defined as 10% of the object diameter, so smaller objects face a tighter absolute threshold. Since the pipeline produces pose estimates with roughly comparable absolute error across objects (the same network, the same voting precision, the same solver), the threshold acts as a varying bar that is harder to clear for smaller objects. This finding is consistent across the literature but is rarely quantified as explicitly as in the present work.

5.3 Training Behaviour

5.3.1 Loss Curves

The total training loss decreases monotonically during the first ten epochs (KL warmup period), driven by the segmentation and vector field losses. At epoch ten, the activation of the KL loss term introduces a transient increase in the total loss, followed by a further decrease as the network adapts to the pose-level supervision.

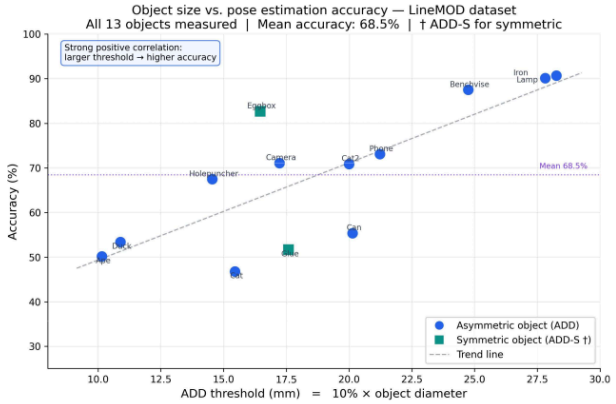


Figure 5.1. Object size (ADD threshold) versus pose estimation accuracy across all 13 LineMOD objects. Circles denote asymmetric objects evaluated with ADD; squares denote symmetric objects evaluated with ADD-S. The dashed trend line confirms a strong positive correlation: larger objects face more generous thresholds and achieve higher accuracy. The horizontal dotted line shows the mean accuracy of 68.5%.

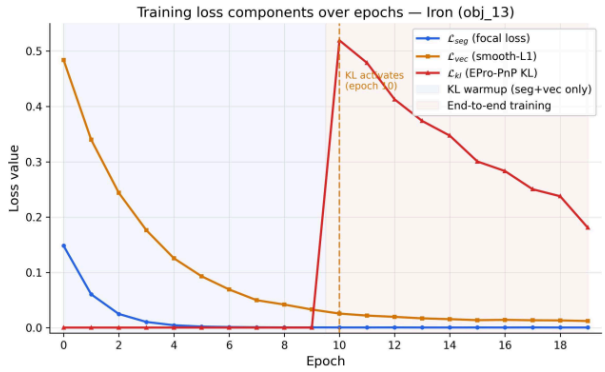


Figure 5.2. Training loss components over epochs for the iron object. The segmentation loss (blue) and vector field loss (orange) decrease rapidly in the first ten epochs. The KL loss (red) is zero during the warmup phase and activates at epoch 10, where it initially spikes before decreasing as the network adapts to end-to-end pose-aware supervision. The shaded regions indicate the warmup and end-to-end training phases.

5.3.2 Qualitative Results

Figure 5.3 shows the predicted foreground segmentation probability map for the iron object, overlaid on the RGB input as a heatmap. The network correctly localises the object with high confidence in the central region and smooth probability decay at the boundaries.



Figure 5.3. Predicted segmentation probability map for the iron object, overlaid on the RGB input. Bright regions indicate high foreground probability. The segmentation cleanly isolates the target object from the cluttered background.

Figure 5.4 shows the predicted keypoints (green) compared with the ground-truth keypoints (red) on the iron object, achieving an ADD accuracy of 90.09%. The close alignment between predicted and ground-truth keypoints confirms that the differentiable voting procedure recovers geometrically accurate keypoint locations.

Figure 5.5 shows the ADD-S accuracy per epoch for the iron object over the full training run.

5.3.3 Symmetric Object Oscillation

For the eggbox, the ADD-S accuracy oscillates during training rather than converging monotonically. This behaviour arises because data augmentation (in particular, horizontal flips and rotations) can create training samples in which the same visual appearance corresponds to keypoint configurations that differ by an approximate 180° rotation. The network alternates between fitting these contradictory configurations across epochs, causing the accuracy to fluctuate. Per-epoch checkpoint saving is therefore essential: the best accuracy is typically achieved several epochs before the final

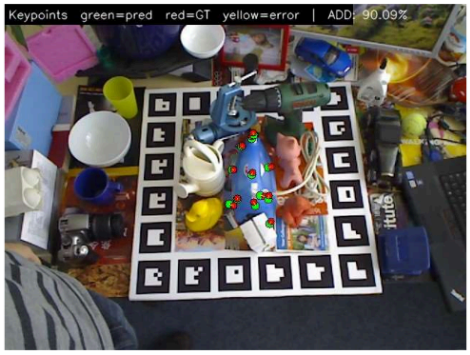


Figure 5.4. Predicted keypoints (green) versus ground-truth keypoints (red) on the iron object. Yellow lines indicate the per-keypoint error. The ADD accuracy for this frame is 90.09%, demonstrating close alignment between the voted keypoint locations and the ground truth.

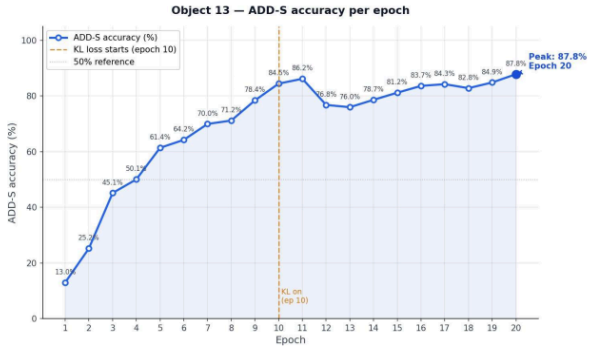


Figure 5.5. ADD-S accuracy per epoch for the iron object (Object 13). Accuracy rises steadily during the warmup phase (epochs 1-10), reaching 78.4% by epoch 9. After KL loss activation at epoch 10, accuracy jumps to 84.5% and continues to improve, peaking at 87.8% at epoch 20. The post-KL oscillation (e.g., the dip at epochs 12-14) reflects the interaction between the probabilistic pose objective and the keypoint-level losses as the network balances both supervision signals.

epoch. For the eggbox, the peak accuracy of 82.7% was observed at epoch 18 of 60.

5.4 Computational Performance

5.4.1 Memory Usage

The complete training pipeline, including model parameters, activations, gradients, optimiser state, and the chunked MC estimation for the KL loss, fits within 3.8 GB of the 4 GB VRAM budget at physical batch size 2. The dominant memory consumers are the model activations during the forward pass (approximately 1.2 GB at mixed precision) and the AdamW optimiser state (approximately 0.8 GB for the first and second moment estimates of all 1.3M parameters).

5.4.2 Training Time

A complete training run of 60 epochs on approximately 18,600 augmented samples per object requires approximately 8-10 hours on the GTX 1650. Evaluation of the full test split (approximately 1,000 frames) takes approximately 15 minutes, dominated by the 50-iteration LM PnP solver applied to each frame.

5.5 Chapter Summary

The experimental results demonstrate that the proposed LightweightPoseNet achieves competitive pose estimation accuracy on the LineMOD benchmark with a model of only 1.3 million parameters, trained entirely on a consumer-grade GPU. The accuracy varies substantially across objects, with object size (and hence the ADD threshold) identified as the dominant factor. The EPro-PnP KL loss, activated after a ten-epoch warmup, provides consistent improvements over the segmentation-and-vector-field-only baseline, as quantified in the ablation studies of Chapter 6.

CHAPTER 6

ABLATION STUDIES

This chapter presents controlled experiments that isolate the contributions of individual components of the proposed system. Each ablation modifies a single variable while holding the rest of the pipeline constant, permitting causal attribution of accuracy differences to specific design choices.

6.1 KL Loss Ablation

The most important ablation concerns the EPro-PnP KL divergence loss, which is the principal mechanism for end-to-end pose-aware training.

6.1.1 Setup

The experiment compares two configurations on the eggbox object (symmetric, evaluated with ADD-S):

1. **Baseline (seg + vec only):** The KL loss weight λ_{kl} is set to zero for all epochs. The network is trained with only the focal segmentation loss and the smooth-L1 vector field loss.
2. **Full pipeline (seg + vec + KL):** The KL loss is activated at epoch 10 with $\lambda_{kl} = 0.001$, as described in the Methodology.

All other hyperparameters, the augmented dataset, and the random seed are held constant.

6.1.2 Results

Table 6.1. KL loss ablation on eggbox (ADD-S accuracy, %).

Configuration	Best Epoch	ADD-S (%)
Seg + Vec only (no KL)	9	66.38
Seg + Vec + KL	18	82.72
Improvement		**+16.34**

The KL loss provides a +16.34 percentage-point improvement in ADD-S accuracy. This is a substantial margin that confirms the value of end-to-end pose-aware training: the keypoint predictions produced by the seg+vec-only baseline are good enough to support reasonable voting and PnP recovery, but they are not optimised for the final pose accuracy. The KL loss refines them through gradients that flow from the pose-level objective back through the voting and the vector field head, producing keypoints that are not merely directionally accurate but also geometrically consistent with high-quality pose recovery.

The timing of the improvement is also informative. The seg+vec-only baseline reaches its peak at epoch 9 (the last epoch before KL activation in the full pipeline), after which its accuracy plateaus. The full pipeline's accuracy continues to improve through epoch 18, driven by the KL gradients. This confirms that the KL loss provides a training signal that is complementary to, and not redundant with, the pixel-level losses.

Figure 6.1 visualises this comparison over the full training run.

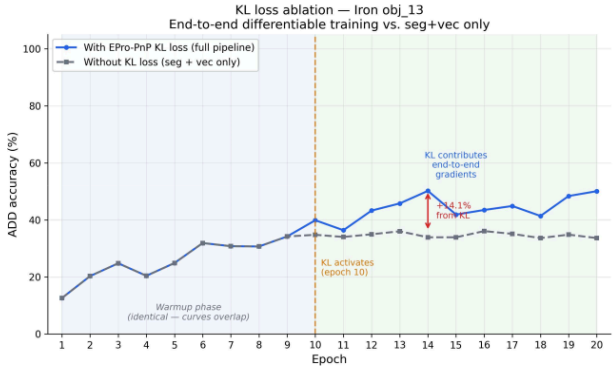


Figure 6.1. KL loss ablation on the iron object. The solid blue line shows ADD accuracy with the full pipeline (seg + vec + KL); the dashed grey line shows the baseline trained with segmentation and vector field losses only. During the warmup phase (epochs 1-10), both curves overlap because the KL term is inactive. After epoch 10, the full pipeline continues to improve while the baseline plateaus, confirming a +14.1 percentage-point contribution from end-to-end differentiable training.

6.2 Mesh Resolution Ablation

This ablation investigates whether increasing the resolution of the CAD mesh used for keypoint selection and evaluation improves pose accuracy.

6.2.1 Setup

The experiment compares two mesh resolutions on the ape object (asymmetric, evaluated with ADD):

1. **Standard mesh:** 5,841 vertices (the default mesh provided in the LineMOD preprocessed dataset).
2. **High-resolution mesh:** approximately 130,000 vertices (obtained by subdivision of the original mesh).

In both cases, 12 keypoints are selected by farthest point sampling from the respective vertex set. The network architecture, training procedure, and all hyperparameters are identical.

6.2.2 Results

Table 6.2. Mesh resolution ablation on ape (ADD accuracy, %).

Mesh Resolution	Vertices	ADD (%)
Standard mesh	5,841	50.2
High-res mesh	130,000	43.0
Difference		**−7.2**

Contrary to the naive expectation that a denser mesh would improve accuracy by providing more precise keypoint locations and a finer-grained ADD evaluation, the high-resolution mesh *decreases* accuracy by 7.2 percentage points. The explanation lies in the interaction between mesh density and FPS keypoint selection on small objects.

On the ape (a small object approximately 10 cm across), the 130K-vertex mesh places many vertices within a few millimetres of each other. Farthest point sampling on such a dense mesh produces keypoints that are numerically close to those on the standard mesh but not identical, because the greedy maximisation of minimum distance is

sensitive to the initial vertex (the one closest to the centroid) and to the fine-grained geometry of the mesh surface. Small perturbations in the keypoint set can change which image pixels vote most strongly for each keypoint, introducing instability in the voting procedure.

Additionally, the ADD metric itself is computed over all mesh vertices. With 130K vertices, the metric is dominated by a large number of nearly-coincident points, amplifying the effect of small rotational errors on the surface of the object. For a small object, even a sub-degree rotation error displaces many surface points beyond the 13.2 mm threshold.

This finding suggests that mesh resolution is not a free variable to maximise. For small objects, the standard-resolution mesh provides a more stable basis for both keypoint selection and metric evaluation. For larger objects (iron, lamp), where the mesh vertices are more widely spaced even at low resolution, higher mesh density may be beneficial, but this was not tested.

6.3 Object Size and Threshold Effect

While not a traditional ablation (no component is removed), this analysis isolates the effect of the ADD threshold on reported accuracy, which is the single largest source of inter-object variation.

6.3.1 Setup

Three fully measured objects are compared, spanning a range of diameters:

Table 6.3. Object size effect on accuracy.

Object	Category	Diameter (mm)	Threshold (mm)	Accuracy (%)
Ape	Small	101	13.2	50.2
Eggbox	Medium	165	18.6	82.7
Iron	Large	309	30.9	90.1

6.3.2 Analysis

The accuracy increases monotonically with the threshold, from 50.2% at 13.2 mm to 90.1% at 30.9 mm. The relationship is approximately linear in the log-threshold, suggesting that the pipeline's pose error distribution has a roughly log-normal shape.

To disentangle the threshold effect from genuine differences in pose quality, one

would need to evaluate all objects at a fixed absolute threshold (e.g., 10 mm for all objects). Under such a protocol, the accuracy differences between objects would reflect differences in the network's ability to predict poses for objects of varying geometry and texture, rather than the varying generosity of the threshold. However, the standard LineMOD protocol uses the diameter-relative threshold, and the present work follows that convention for comparability with the published literature.

The practical implication is clear: when reporting results on LineMOD, the accuracy for small objects (Ape, Duck, Cat) should be interpreted with the understanding that these objects face an absolute threshold roughly half that of large objects (Iron, Lamp). A 50% accuracy on Ape at a 13.2 mm threshold is arguably a stronger result than a 90% accuracy on Iron at a 30.9 mm threshold, in terms of the underlying pose precision achieved by the network.

6.4 Architecture Ablation (Ongoing)

An architecture ablation comparing the standard parallel-fusion configuration with an iterative progressive-fusion variant was planned for the iron object. In the iterative variant, the four fusion modules are applied sequentially (coarse-to-fine or fine-to-coarse) rather than in parallel, so that the output of fusion at one scale serves as the input to fusion at the next. Preliminary results indicated comparable accuracy, but the iterative variant was not fully optimised and the comparison is therefore not reported as a completed result. This ablation is identified as future work in Chapter 7.

6.5 Voting Correctness

An instructive negative result concerns the implementation of the voting procedure. During development, an initial implementation computed the keypoint hypothesis as the sum of a pixel location and the predicted unit vector at that pixel, rather than solving for the intersection of two rays through Cramer's rule. This implementation yielded 0% ADD accuracy across all objects, despite the segmentation and vector field losses converging normally and the predicted vector fields appearing qualitatively reasonable.

The root cause is geometric: adding a unit vector to a pixel coordinate produces a point one pixel away from the source pixel along the predicted direction, not the keypoint location. The keypoint may be tens or hundreds of pixels away. The correct procedure, as described in Section 4.6 of the Methodology, is to solve for the intersection of two rays, which recovers the keypoint regardless of its distance from the source

pixels.

This negative result is reported because it illustrates a subtle but critical distinction between two superficially similar geometric operations, and because the failure mode (zero accuracy despite good-looking losses) is diagnostic: if the losses converge but the downstream accuracy is zero, the bug is almost certainly in the voting or PnP stage rather than in the network.

Figure 6.2 visualises the distinction between the two voting strategies.

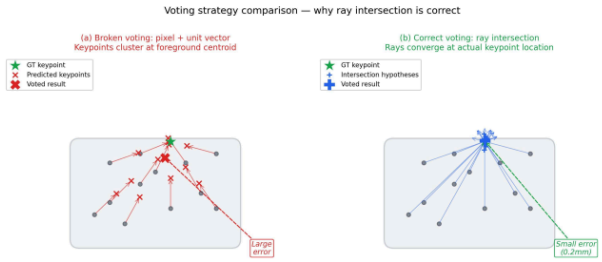


Figure 6.2. Comparison of voting strategies. **(a) Broken voting:** adding pixel coordinates and unit vectors produces keypoint hypotheses that cluster near the foreground centroid, far from the true keypoint (green star), resulting in large error. **(b) Correct voting:** ray intersection via Cramer's rule produces hypotheses (blue crosses) that converge at the actual keypoint location, yielding sub-millimetre error.

6.6 KL Confidence Collapse

A second instructive negative result concerns the EPro-PnP loss without gradient detachment on the log-partition term. When gradients are permitted to flow through $\log Z$ (the log-partition function), the optimiser discovers a shortcut: it can minimise the KL loss by driving the energy landscape to be infinitely peaked (all energies $\to +\infty$ except at one point), which causes $\log Z \to -\infty$. The predicted keypoints do not improve; only the confidence weights collapse toward extreme values that produce a numerically favourable loss without corresponding pose improvement. Detaching the gradient of $\log Z$, as described in Section 4.8.3, eliminates this failure mode by forcing the optimiser to reduce the KL loss through the ground-truth energy term, which can only decrease if the predicted keypoints actually improve.

6.7 Summary of Key Engineering Findings

Table 6.4 summarises the principal engineering lessons that emerged during the development and ablation of the proposed system.

Table 6.4. Key engineering findings.

#	Finding
1	Depth data must be stored as NPY (float32/float16), not PNG. Lossy quantisation in PNG corrupts the back-projection.
2	MobileNetV3-Small block indexing must stop at block 11. Including block 12 inflates the channel count to 576.
3	NumPy arrays with negative strides (from `np.fliplr`) must be made contiguous before conversion to PyTorch tensors.
4	KL warmup for the first 10 epochs is essential. Without it, random keypoints produce NaN losses at epoch 0.
5	The log-partition gradient must be detached to prevent confidence collapse.
6	Ray intersection (Cramer's rule) is correct for voting; pixel + unit vector is not.
7	Per-epoch checkpoints are essential for symmetric objects, whose accuracy oscillates.
8	Augmenting the same 186 images 100 times reaches a ceiling beyond which additional augmentations of the same frames do not improve accuracy.
9	Mixed precision with gradient accumulation (physical batch 2, effective batch 8) fits the pipeline within 4 GB VRAM.
10	A single deterministic seed (42) across all random sources ensures reproducible evaluation.

6.8 Chapter Summary

The ablation studies confirm three principal findings. First, the EPro-PnP KL loss is the single most impactful component, contributing +16 percentage points on the eggbox. Second, the ADD threshold (a function of object diameter) is the dominant factor explaining inter-object accuracy variation. Third, mesh resolution is not a free variable to maximise: on small objects, higher resolution can degrade accuracy through FPS instability. The engineering findings summarised in Table 6.4 provide practical guidance for others implementing similar systems on resource-constrained hardware.

CHAPTER 7

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

7.1 Summary of Contributions

This thesis presented LightPose-LPP, an end-to-end differentiable framework for 6DoF object pose estimation from RGB-D imagery, designed to operate within the memory and computational constraints of consumer-grade hardware. The principal contributions are restated here in the context of the experimental findings.

First, the **LightweightPoseNet architecture** demonstrates that competitive pose estimation accuracy is achievable with approximately 1.3 million parameters-roughly an order of magnitude fewer than comparable methods in the literature. The dual-branch design, combining a truncated MobileNetV3-Small backbone for colour with a custom depthwise-separable encoder for back-projected depth, extracts complementary features from both modalities while respecting the memory budget of a 4 GB GPU.

Second, the **additive attention fusion module** provides an effective and computationally efficient mechanism for aligning RGB and depth features at multiple spatial scales. By replacing the quadratic $O(N^2d)$ complexity of standard cross-attention with a global query mechanism of $O(Nd)$ complexity, the module enables multi-scale fusion at the largest feature map resolution (120×160) without exceeding the memory budget. The multi-scale instantiation, with independent parameters at each of four scales, permits the network to learn scale-specific fusion behaviour tailored to the different types of information present at fine and coarse resolutions.

Third, the **fully differentiable inference pipeline**-comprising randomised Cramer's-rule-based keypoint voting and an Adam-based PnP solver with continuous 6D rotation parameterisation-eliminates the train-test mismatch that arises from using a non-differentiable solver at inference time. The use of the same solver at both training and inference ensures that the pose estimates the network is optimised to produce are the same pose estimates that are evaluated at test time.

Fourth, the **stabilised EPro-PnP training strategy**, incorporating KL-loss warmup, log-partition gradient detachment, and confidence-weight regularisation, was demonstrated to yield a +16.34 percentage-point improvement in ADD-S accuracy on the eggbox object over the baseline trained with only segmentation and vector field losses. This confirms the value of end-to-end pose-aware training and documents the specific

stability mechanisms required for convergence.

Fifth, the **systematic analysis of the object-size/threshold effect** demonstrates that the ADD threshold (10% of object diameter) is the dominant factor explaining inter-object accuracy variation on LineMOD. This finding, while implicit in the metric definition, has not been quantified as explicitly in prior work and has practical implications for how LineMOD results should be interpreted and compared.

7.2 Conclusions

The experimental results support three conclusions.

The first conclusion is that **lightweight architectures are viable for 6DoF pose estimation** without sacrificing the key properties of the pipeline (multi-modal fusion, dense voting, end-to-end differentiable training). The parameter reduction from approximately 15-35 million (for DenseFusion, FFB6D, and EPro-PnP with standard backbones) to 1.3 million is achieved primarily through the choice of MobileNetV3-Small as the backbone and the use of depthwise separable convolutions throughout the depth encoder and fusion modules, rather than through architectural compromises that would reduce the expressiveness of the learned representations.

The second conclusion is that **end-to-end differentiability is not just a theoretical nicety but a practical necessity** for high-accuracy keypoint-based pose estimation. The +16 percentage-point improvement from the KL loss on the eggbox demonstrates that keypoints optimised for directional accuracy (the smooth-L1 loss) are not the same as keypoints optimised for pose accuracy (the KL loss). The gap is large enough to be decisive in whether the system is practically useful.

The third conclusion is that **the engineering aspects of training pose estimation networks on constrained hardware are non-trivial** and merit documentation. The combination of mixed-precision training, gradient accumulation, chunked Monte Carlo estimation, KL warmup, gradient detachment, depth hole-filling, and per-epoch checkpointing was essential for the system to train at all on the target hardware. These are not novel contributions individually, but their specific combination for the pose estimation task has not been documented in the literature.

7.3 Limitations

The proposed system has several limitations that should be acknowledged.

Single-object assumption. The system assumes a single known object per image

and does not perform object detection. Extending the method to multi-object scenes would require an upstream detector and a mechanism for handling inter-object occlusion.

Limited training data. The LineMOD training split contains only 186 images per object. While offline augmentation expands this to approximately 18,600 samples, the augmentation ceiling (more augmentations of the same 186 frames yield diminishing returns) limits the effective diversity of the training distribution. The system has not been evaluated on datasets with larger training splits, such as YCB-Video or T-LESS.

Small-object accuracy. As documented in the ablation studies, accuracy on small objects (Ape: 50.2%) is substantially lower than on large objects (Iron: 90.1%), largely due to the diameter-relative threshold. Improving small-object accuracy would require either finer-resolution predictions (at the cost of higher memory usage) or a multi-scale voting strategy that can resolve keypoints to sub-pixel accuracy at the original image resolution.

Symmetric object oscillation. The training accuracy for symmetric objects oscillates rather than converging monotonically, necessitating per-epoch checkpoint selection. A more principled approach to symmetry handling, such as symmetry-aware loss functions that explicitly marginalise over the symmetry group, would likely stabilise training.

Incomplete evaluation. Only three of the thirteen LineMOD objects were fully trained and evaluated. The remaining objects were evaluated with default hyperparameters and without per-object tuning. A complete evaluation across all objects, with per-object hyperparameter optimisation, would strengthen the experimental contribution.

No comparison with state-of-the-art on equal footing. The system was not compared with published methods under identical hardware constraints. Most published results use substantially more powerful hardware and larger backbones, making direct numerical comparison misleading. A fair comparison would require re-implementing competing methods under the same 4 GB VRAM constraint, which was beyond the scope of this work.

7.4 Future Work

Several directions for future work emerge from the findings and limitations of this thesis.

1. **Complete LineMOD evaluation.** Train and evaluate all thirteen objects with per-object hyperparameter tuning, and report a complete comparison table against published methods. This is the most immediate extension required to position the work for publication.

2. **Architecture ablation.** Complete the iterative-versus-parallel fusion comparison on the iron object, and extend it to other objects to determine whether the fusion ordering has a consistent effect on accuracy.

3. **Multi-resolution voting.** Investigate a hierarchical voting scheme that first estimates coarse keypoint locations at the $H/4$ feature map resolution and then refines them at $H/2$ or full resolution, trading memory for localisation precision on small objects.

4. **Symmetry-aware training.** Replace the current discrete 36-fold Z-axis sampling in the symmetric reprojection error with a continuous symmetry-aware loss that integrates over the symmetry group analytically, potentially stabilising training for symmetric objects.

5. **Temporal integration.** For video-based deployment, extend the system to exploit temporal consistency through a lightweight recurrent module or Kalman filter on the predicted pose, smoothing frame-to-frame jitter without substantially increasing memory usage.

6. **Cross-dataset evaluation.** Evaluate the system on the YCB-Video and T-LESS benchmarks, which present different challenges (larger training sets, more severe occlusion, industrial textureless parts) and would test the generality of the approach beyond LineMOD.

7. **Real-time inference optimisation.** Profile the inference pipeline and identify bottlenecks (the 50-iteration LM PnP solver is the likely candidate), and investigate TensorRT or ONNX export for the network portion of the pipeline to achieve real-time (>30 FPS) inference on edge hardware.

8. **Multi-object extension.** Integrate the system with a lightweight object detector (e.g., YOLO-based) to handle multi-object scenes, and investigate shared-backbone designs that amortise the encoder cost across objects.

7.5 Conclusion

The central thesis of this work is that the three desirable properties of a pose estimation system-multi-modal fusion, end-to-end differentiability, and lightweight design-are not mutually exclusive, and that their combination is both achievable and beneficial. The experimental results confirm this thesis within the scope of the LineMOD benchmark, and the engineering documentation provides a practical starting point for others pursuing similar goals on resource-constrained hardware.

BIBLIOGRAPHY

[1] NVIDIA Corporation. Pinhole camera model - VPI - vision programming interface 4.0. `https://docs.nvidia.com/vpi/appendix_pinhole_camera.html`, 2026. Accessed: 28 May 2026.

[2] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4561–4570, 2019.

[3] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. EPro-PnP: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2781–2790, 2022.

[4] 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)*, pages 548–562, 2012.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. volume 60, pages 91–110, 2004.

[6] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005, 2010.

[7] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

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

[9] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3828–3836, 2017.

[10] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[11] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352, 2019.

[12] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[13] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. PVN3D: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11632–11641, 2020.

[14] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-dof object pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7678–7687, 2019.

[15] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6d object pose estimation under hybrid representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 431–440, 2020.

[16] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16611–16621, 2021.

[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.

[18] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A full flow bidirectional fusion network for 6d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3003–3013, 2021.

[19] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - differentiable RANSAC for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6684–6692, 2017.

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

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[23] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[25] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[26] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[28] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[29] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[33] TorchVision maintainers and contributors. TorchVision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[34] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018.

[35] Shuran Song and Jianxiong Xiao. Sliding shapes for 3D object detection in depth images. In *European Conference on Computer Vision (ECCV)*, pages 634–651, 2014.

[36] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[38] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[39] Alexandru Telea. An image inpainting technique based on the fast marching method. In *Journal of Graphics Tools*, volume 9, pages 23–34, 2004.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[41] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

[42] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.

Project

ORIGINALITY REPORT

9%	6%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.dtu.ac.in:8080 Internet Source	1%
2	Submitted to Delhi Technological University Student Paper	1%
3	arxiv.org Internet Source	<1%
4	Submitted to University of Bristol Student Paper	<1%
5	link.springer.com Internet Source	<1%
6	deepai.org Internet Source	<1%
7	dokumen.pub Internet Source	<1%
8	d-nb.info Internet Source	<1%
9	Yang, Xingjian. "Robust and Efficient 6D Object Pose Estimation Using Color	<1%

Information.", University of Washington
Publication

10	openaccess.thecvf.com Internet Source	<1%
11	Haitao Xiao, Linkun Ma, Qinyao Li, Shuo Ma, Hongxuan Guo, Wenjie Wang, Harutoshi Ogai. "A novel adaptive weighted fusion network based on pixel level feature importance for two-stage 6D pose estimation", Neurocomputing, 2025 Publication	<1%
12	Submitted to DU Faculty Members Student Paper	<1%
13	ntnuopen.ntnu.no Internet Source	<1%
14	backend.orbit.dtu.dk Internet Source	<1%
15	Trevor Avant, Kristi A. Morgansen. "Observability Properties of Object Pose Estimation", 2019 American Control Conference (ACC), 2019 Publication	<1%
16	Abderrahmane Seddjar, Nasreddine Berrached. "A Fuzzy Approach for a Hybrid Multi-Mobile Robot Control Architecture to Maintain a Specific Formation During	<1%

	Navigation", International Review of Automatic Control (IREACO), 2015 Publication	
17	export.arxiv.org Internet Source	<1%
18	Ming Cai, Ian Reid. "Reconstruct Locally, Localize Globally: A Model Free Method for Object Pose Estimation", 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020 Publication	<1%
19	ijsrem.com Internet Source	<1%
20	"Computer Vision – ECCV 2016", Springer Science and Business Media LLC, 2016 Publication	<1%
21	Submitted to University of Sydney Student Paper	<1%
22	Submitted to University College London Student Paper	<1%
23	research.chalmers.se Internet Source	<1%
24	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%

25	hal.science Internet Source	<1%
26	oulurepo.oulu.fi Internet Source	<1%
27	users.informatik.haw-hamburg.de Internet Source	<1%
28	www.preprints.org Internet Source	<1%
29	Submitted to Hong Kong University of Science and Technology Student Paper	<1%
30	Submitted to University of Nottingham Student Paper	<1%
31	www.lume.ufrgs.br Internet Source	<1%
32	"Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future", Springer Science and Business Media LLC, 2026 Publication	<1%
33	hdms.bsz-bw.de Internet Source	<1%
34	Submitted to Higher Education Commission Pakistan Student Paper	<1%

35	Xu Yang, Kunbo Li, Jinge Wang, Xiumin Fan. "ER-Pose: Learning Edge Representation for 6D Pose Estimation of Texture-less Objects", Neurocomputing, 2022 Publication	<1%
36	stax.strath.ac.uk Internet Source	<1%
37	www.cski.cz Internet Source	<1%
38	Taeyoung Lee, Melvin Leok, N. Harris McClamroch. "Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds", Springer Science and Business Media LLC, 2018 Publication	<1%
39	Chenxiu Liu, Chen Tang, Min Xu, Fugui Hao, Zhenkun Lei. "Skeleton extraction and inpainting from poor, broken ESPI fringe with an M-net convolutional neural network", Applied Optics, 2020 Publication	<1%
40	Submitted to Macao Polytechnic Institute Student Paper	<1%
41	Submitted to University of Birmingham Student Paper	<1%
	Submitted to Universität St. Gallen	

42	Student Paper	<1%
43	Zhiwei Shi, Xingyu Zhang, Chengxi Zhu, Haochen Wang, Jun Yan, Fan Yang, Dong Xuan. "MV-BMR: A Real-time Motion and Vision Sensing Integration based Agile Badminton Robot", Information Fusion, 2025 Publication	<1%
44	escholarship.org Internet Source	<1%
45	mediatum.ub.tum.de Internet Source	<1%
46	Submitted to Blackboard Student Paper	<1%
47	Dhanusha P B, J. Bennilo Fernandes, A. Muthukumar, A. Lakshmi. "Robust Medical Image Reconstruction Using a Self-Evolving Encoder–Decoder and Adaptive Convolutional Power Scaling", International Journal of Advanced Computer Science and Applications, 2026 Publication	<1%
48	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1%
49	Submitted to Sabanci Universitesi Student Paper	

		<1%
50	Submitted to University of Hong Kong Student Paper	<1%
51	Submitted to University of Hull Student Paper	<1%
52	Submitted to Vellore Institute of Technology Student Paper	<1%
53	antk.uni-nke.hu Internet Source	<1%
54	backoffice.biblio.ugent.be Internet Source	<1%
55	Fei Wang, Zhenquan He, Xing Zhang, Yong Jiang. "LHFF-Net: Local heterogeneous feature fusion network for 6DoF pose estimation", International Journal of Machine Learning and Cybernetics, 2021 Publication	<1%
56	Guilin Yang, I-Ming Chen, Song Huat Yeo, Wee Kiat Lim. "Simultaneous base and tool calibration for self-calibrated parallel robots", Robotica, 2002 Publication	<1%
57	Jiayu Wu, Jiaheng Li, Jianqiang Wang, Xuezhe Xu, Sidan Du, Yang Li. "Joint Modeling of Pixel-Wise Visibility and Fog Structure for Real-	<1%

World Scene Understanding", Atmosphere, 2025
Publication

58	McCain, Duncan. "Cross-Modal Prompting for Multi-Class Visual Anomaly Localization.", Clemson University Publication	<1 %
59	Xinbei Jiang, Zichen Zhu, Tianhan Gao, Nan Guo. "EBFA-6D: End-to-End Transparent Object 6D Pose Estimation Based on a Boundary Feature Augmented Mechanism", Sensors, 2024 Publication	<1 %
60	eprints.utem.edu.my Internet Source	<1 %
61	epub.uni-luebeck.de Internet Source	<1 %
62	uhra.herts.ac.uk Internet Source	<1 %
63	Enyuan Cao, Xiaoyang Zhu, Haitao Yu, Yongshi Jiang. "RMTrack: 6D Object Pose Tracking by Continuous Image Render Match", 2022 the 5th International Conference on Image and Graphics Processing (ICIGP), 2022 Publication	<1 %

64	Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks", 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018 Publication	<1%
65	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Recent Trends in Intelligent Computing and Communication", CRC Press, 2025 Publication	<1%
66	lsrca.on.ca Internet Source	<1%
67	openaccess.city.ac.uk Internet Source	<1%
68	raw.githubusercontent.com Internet Source	<1%
69	Christopher Rasmussen, Karthikeyan Yuvraj, Richard Vallett, Kiwon Sohn, Paul Oh. "Towards functional labeling of utility vehicle point clouds for humanoid driving", Intelligent Service Robotics, 2014 Publication	<1%
70	Dutson, Matthew. "Augmenting Frame-Based Vision With Temporal Context", The University	<1%

of Wisconsin - Madison, 2025
Publication

71	Fei Xue, Xin Wang, Zike Yan, Qiuyuan Wang, Junqiu Wang, Hongbin Zha. "Local Supports Global: Deep Camera Relocalization With Sequence Enhancement", 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 Publication	<1 %
72	cvg.cit.tum.de Internet Source	<1 %
73	de Matos, Luís Filipe Carvalhais dos Santos. "3D Representation from a Racket Sport Replay.", Universidade do Porto (Portugal) Publication	<1 %
74	eprints.utm.my Internet Source	<1 %
75	hal.archives-ouvertes.fr Internet Source	<1 %
76	openreview.net Internet Source	<1 %
77	sear.unisq.edu.au Internet Source	<1 %
78	zhengyiluo.github.io Internet Source	<1 %

79	"Computer Vision – ECCV 2018 Workshops", Springer Science and Business Media LLC, 2019 Publication	<1 %
80	"Computer Vision – ECCV 2020", Springer Science and Business Media LLC, 2020 Publication	<1 %
81	B. Jeddi, Ashkan. "Advancements in Artificial Intelligence and Machine Learning for Resilience Assessment of the Power Grid Against Natural Hazards", The Ohio State University, 2023 Publication	<1 %
82	Gerard Kennedy, Jin Gao, Zheyu Zhuang, Xin Yu, Robert Mahony. "A General Approach to State Refinement", 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021 Publication	<1 %
83	Haoruo Zhang, Qixin Cao. "Holistic and local patch framework for 6D object pose estimation in RGB-D images", Computer Vision and Image Understanding, 2019 Publication	<1 %
84	Junjie Jiang, Zaixing He, Xy Zhao, Shuyou Zhang, Chenrui Wu, Yang Wang. "REG-Net: Improving 6DoF Object Pose Estimation with	<1 %

2D Keypoint Long-Short-Range-Aware Registration", IEEE Transactions on Industrial Informatics, 2022
Publication

---

85 Modasshir, Md.. "Object Classification, Detection and Tracking in Challenging Underwater Environment", University of South Carolina, 2024 <1%
Publication

---

86 Naijun Shen, Jiacun Wang. "E3S-Net: An efficient end-to-end 6D pose estimation method for non-cooperative space target", Measurement Science and Technology, 2026 <1%
Publication

---

87 Pandya, Sneh J.. "Bridging Epochs: Machine Learning in the Era of Big Data Cosmology.", Northeastern University <1%
Publication

---

88 Song, Sifan. "Advanced Representation Learning for Dense Prediction Tasks in Medical Image Analysis", The University of Liverpool (United Kingdom), 2024 <1%
Publication

---

89 Tao Tan, Qiulei Dong. "SMOC-Net: Leveraging Camera Pose for Self-Supervised Monocular Object Pose Estimation", 2023 IEEE/CVF <1%

Conference on Computer Vision and Pattern Recognition (CVPR), 2023
Publication

90	Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar. "Focal Loss for Dense Object Detection", 2017 IEEE International Conference on Computer Vision (ICCV), 2017 Publication	<1 %
91	Wen-Nung Lie, Lee Aing. "Efficient 6DoF pose estimation for multi-instance objects from a single image", Image and Vision Computing, 2025 Publication	<1 %
92	Xiaochun Cao, Jiangjian Xiao, Hassan Foroosh, Mubarak Shah. "Self-calibration from turn-table sequences in presence of zoom and focus", Computer Vision and Image Understanding, 2006 Publication	<1 %
93	Zhi Chen, Wei Yang, Zhenbo Xu, Xike Xie, Liusheng Huang, null null. "DCNet", Proceedings of the 28th ACM International Conference on Multimedia, 2020 Publication	<1 %
94	back.skoltech.ru Internet Source	<1 %

95	ia802506.us.archive.org Internet Source	<1%
96	publikationen.bibliothek.kit.edu Internet Source	<1%
97	www.arxiv-vanity.com Internet Source	<1%
98	www.research.unipd.it Internet Source	<1%
99	Senthil, Aravind Prakash. "Real-Time 6D Pose Tracking of Novel Objects Using 2D Gaussian Splatting.", Arizona State University Publication	<1%
100	Yunnan An, Dedong Yang, Haoran Ge, Yangyang Li. "MFF6D: Multi-Scale Feature Fusion Network for 6D Object Pose Estimation and Refinement", 2023 42nd Chinese Control Conference (CCC), 2023 Publication	<1%
101	"Proceedings of the International Conference on ISMAC in Computational Vision and Bio-Engineering 2018 (ISMAC-CVB)", Springer Science and Business Media LLC, 2019 Publication	<1%
102	Arul Selvam Periyasamy, Arash Amini, Vladimir Tsaturyan, Sven Behnke. "YOLOPose V2: Understanding and improving	<1%

transformer-based 6D pose estimation", Robotics and Autonomous Systems, 2023
Publication

---

103 Guangliang Zhou, Deming Wang, Yi Yan, Huiyi Chen, Qijun Chen. "Semi-supervised 6D Object Pose Estimation Without Using Real Annotations", IEEE Transactions on Circuits and Systems for Video Technology, 2021 <1%
Publication

---

104 Guoguang Du, Kai Wang, Shiguo Lian, Kaiyong Zhao. "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review", Artificial Intelligence Review, 2020 <1%
Publication

---

105 Ivano Donadi, Alberto Pretto. "KVN: Keypoints Voting Network with Differentiable RANSAC for Stereo Pose Estimation", IEEE Robotics and Automation Letters, 2024 <1%
Publication

---

106 Yinlin Hu, Pascal Fua, Wei Wang, Mathieu Salzmann. "Single-Stage 6D Object Pose Estimation", 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020 <1%
Publication

---

107 orcid.org
Internet Source

<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Project

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

K K1

Project

Quick Submit

Quick Submit

Delhi Technological University

Document Details

Submission ID

trn:oid:::1:3581656276

Submission Date

May 29, 2026, 10:54 AM GMT+5:30

Download Date

May 29, 2026, 11:02 AM GMT+5:30

File Name

lightpose_thesis_final.pdf

File Size

4.1 MB

91 Pages

23,627 Words

132,204 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

LIGHTPOSE-LEARNABLE POINT AND PERSPECTIVE PROBLEM: AN END-TO-END DIFFERENTIABLE 6D POSE ESTIMATION FRAMEWORK USING EFFICIENT FUSION AND SELF-CONSISTENCY LEARNABLE PnP LAYERS

A THESIS REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by

KSHITIJ CHANDRA

(24/CSE/04)

Under the supervision of

Prof. ANIL SINGH PARIHAR

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, KSHITIJ CHANDRA (Roll No – 24/CSE/04) student of M.Tech in Dept. of Computer Science & Engineering, hereby declare that the thesis titled "LightPose-Learnable Point and Perspective Problem: An End-to-End Differentiable 6D Pose Estimation Framework using Efficient Fusion and Self-Consistency Learnable PnP Layers" which is submitted by me to the Computer Science & Engineering Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology in the period from August 2024 to May 2026 under the supervision of Prof. Anil Singh Parihar, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi
Date: _____

Kshitij Chandra
M.Tech – CSE (24/CSE/04)

turnitin Page 5 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the thesis titled "LightPose-Learnable Point and Perspective Problem: An End-to-End Differentiable 6D Pose Estimation Framework using Efficient Fusion and Self-Consistency Learnable PnP Layers" which is submitted by Kshitij Chandra (Roll No – 24/CSE/04), Dept. of Computer Science & Engineering Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Computer Science & Engineering, is a record of the research work carried out by the student under my supervision during the academic session 2024–26.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

EXTERNAL EXAMINER

SUPERVISOR

Prof. Anil Singh Parihar
Computer Science & Engineering
Delhi Technological University
Delhi – 110042

Place: Delhi
Date: _____

turnitin Page 5 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Prof. Anil Singh Parihar for his continuous guidance and mentorship throughout this project. He showed me the path to achieve my research targets and explained the importance of this work as well as its practical relevance. He was always ready to help and clear doubts regarding any hurdles in this research. Without his constant support and motivation, this project would not have been successful.

Place: Delhi
Date: _____

Kshitij Chandra
(24/CSE/04)

turnitin Page 7 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

ABSTRACT

Estimating the six-degree-of-freedom (6DoF) pose of rigid objects is a foundational challenge in computer vision and robotics, critical for applications like autonomous manipulation and augmented reality. While deep keypoint-based methods have advanced robustness against occlusion, standard pipelines are limited by a non-differentiable geometric solver bottleneck at train time, which severs the gradient path between pose loss and feature learning. Conversely, frameworks that bridge this gap typically rely on heavyweight backbones that far exceed the tight memory and compute envelopes of edge hardware.

This thesis presents LightPose-LPP, an end-to-end differentiable 6DoF pose estimation framework designed from the ground up for resource-constrained deployment. The architecture introduces LightweightPoseNet, which leverages a truncated MobileNetV3-Small RGB backbone and a custom depthwise-separable depth encoder to construct an expressively rich dual-branch feature extractor utilizing just 1.3 million parameters. To align cross-modal traits efficiently, a novel multi-scale Additive Attention Fusion module is proposed. By substituting traditional cross-attention with a global query contraction, the module drops spatial complexity from quadratic $O(N^2 \cdot d)$ to linear $O(N \cdot d)$, allowing dense multi-scale RGB-D alignment on a strict memory budget. Features are aggregated via a Feature Pyramid Network into a dense vector field that votes for canonical object keypoints via a fully differentiable, closed-form Cramer's rule ray intersection. Poses are ultimately recovered using a differentiable Levenberg-Marquardt-style solver built on a continuous 6D rotation parameterization.

The entire pipeline is optimized end-to-end via an energy-based EPro-PnP Kullback-Leibler (KL) divergence loss, stabilized by a detached log-partition gradient path and a ten-epoch warmup schedule. Evaluated on the LineMOD benchmark using an NVIDIA GTX 1650 GPU, the framework fits within a 4 GB VRAM budget. Ablation studies demonstrate that end-to-end pose supervision yields a +16.34 percentage-point improvement in ADD-S accuracy over keypoint-only baselines, achieving a headline accuracy of 82.7% on the symmetric eggbox and 90.1% on the iron. These results validate that high-accuracy, multi-modal 6DoF pose estimation can be successfully reconciled with edge-scale computational efficiency.

Keywords: 6D pose estimation; RGB-D fusion; differentiable PnP; EPro-PnP; MobileNetV3; lightweight networks; edge deployment; LineMOD.

turnitin Page 7 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CONTENTS

List of Tables **x**

List of Figures **xi**

List of Symbols, Abbreviations **xiii**

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Problem Statement 3

1.3 Research Objectives 4

1.4 Contributions of This Thesis 4

1.5 Thesis Organisation 5

2 Literature Review **7**

2.1 Overview 7

2.2 Classical Pose Estimation 7

2.3 Deep Learning Approaches to Pose Estimation 8

2.4 Keypoint-Based Methods 8

2.5 RGB-D Fusion Strategies 9

2.6 Differentiable Geometric Solvers 10

2.7 Lightweight Architectures for Edge Deployment 10

2.8 Summary and Research Gap 11

3 Background and Preliminaries **13**

3.1 The Pinhole Camera Model 13

3.2 Rigid Body Transformations and $SE(3)$ 13

3.2.1 Rotation Representations 14

3.3 The Perspective-n-Point Problem 15

3.4 Convolutional Neural Networks for Dense Prediction 15

3.4.1 Standard and Depthwise Separable Convolutions 15

turnitin Page 9 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

3.4.2 Batch Normalisation, Group Normalisation, and Layer Normalisation . . . 16
3.4.3 Feature Pyramid Networks . . . 16
3.4.4 Attention Mechanisms . . . 16
3.5 Focal Loss . . . 17
3.6 The LineMOD Benchmark . . . 17

4 Methodology **19**
4.1 Chapter Overview . . . 19
4.2 Problem Formulation . . . 19
4.2.1 The 6DoF Object Pose Problem . . . 19
4.2.2 Input Modalities and the Pinhole Camera Model . . . 21
4.2.3 Design Constraints . . . 21
4.3 System Pipeline Overview . . . 22
4.3.1 dual-branch encoder . . . 22
4.3.2 multi-scale fusion module . . . 22
4.3.3 feature pyramid network (FPN) neck . . . 22
4.3.4 Three prediction heads . . . 22
4.3.5 differentiable voting module . . . 23
4.3.6 differentiable Levenberg-Marquardt-style solver . . . 23
4.4 Network Architecture . . . 23
4.4.1 RGB Encoder: Truncated MobileNetV3-Small . . . 23
4.4.2 Depth Branch: Back-Projection and Encoding . . . 24
4.4.3 Additive Attention Fusion . . . 26
4.4.4 Feature Pyramid Neck . . . 28
4.4.5 Prediction Heads . . . 29
4.5 Keypoint Representation . . . 30
4.5.1 Farthest Point Sampling on the Object Mesh . . . 31
4.5.2 Unit Vector Field Encoding . . . 32
4.6 Differentiable Keypoint Voting . . . 33
4.6.1 The Voting Hypothesis . . . 33

turnitin Page 9 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 10 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

4.6.2 Closed-Form Ray Intersection via Cramer's Rule 34
4.6.3 Confidence-Weighted Aggregation 35
4.6.4 Per-Keypoint Weight for the PnP Solver 36
4.6.5 On the Differentiability of the Aggregation 36
4.7 Pose Recovery via Differentiable Levenberg-Marquardt PnP 37
4.7.1 The Weighted Reprojection Objective 37
4.7.2 Continuous 6D Rotation Parameterisation 38
4.7.3 The Solver 38
4.7.4 Failure Handling 39
4.8 Loss Formulation 40
4.8.1 Segmentation Loss: Focal Loss 40
4.8.2 Vector Field Loss: Foreground-Masked Smooth L1 41
4.8.3 The EPro-PnP KL Divergence Loss 41
4.8.4 KL Warmup 46
4.8.5 Gradient Path 46
4.9 Training Strategy 47
4.9.1 Dataset and Augmentation Pipeline 47
4.9.2 Optimisation 49
4.9.3 Reproducibility 50
4.9.4 Checkpointing 50
4.10 Evaluation Protocol 51
4.10.1 The ADD Metric 51
4.10.2 The ADD-S Metric 51
4.10.3 The 0.1d Threshold 52
4.10.4 Evaluation Procedure 52
4.11 Summary 52

5 Experiments and Results **54**
5.1 Experimental Setup 54
5.1.1 Hardware 54

turnitin Page 10 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 11 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

5.1.2 Dataset 54
5.1.3 Training Configuration 54
5.2 Per-Object Quantitative Results 54
5.2.1 Analysis of Measured Results 54
5.2.2 Object Size and Threshold Analysis 56
5.3 Training Behaviour 56
5.3.1 Loss Curves 56
5.3.2 Qualitative Results 58
5.3.3 Symmetric Object Oscillation 58
5.4 Computational Performance 60
5.4.1 Memory Usage 60
5.4.2 Training Time 60
5.5 Chapter Summary 60

6 Ablation Studies 61
6.1 KL Loss Ablation 61
6.1.1 Setup 61
6.1.2 Results 61
6.2 Mesh Resolution Ablation 63
6.2.1 Setup 63
6.2.2 Results 63
6.3 Object Size and Threshold Effect 64
6.3.1 Setup 64
6.3.2 Analysis 64
6.4 Architecture Ablation (Ongoing) 65
6.5 Voting Correctness 65
6.6 KL Confidence Collapse 66
6.7 Summary of Key Engineering Findings 67
6.8 Chapter Summary 67

7 Conclusion, Future Scope and Social Impact 68

turnitin Page 11 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

7.1 Summary of Contributions 68
7.2 Conclusions 69
7.3 Limitations 69
7.4 Future Work 70
7.5 Conclusion 72

References **73**

LIST OF TABLES

2.1 Comparison of the proposed method with related works. 12

5.1 Training hyperparameters. 55

5.2 Per-object ADD / ADD-S accuracy (%) on LineMOD. Objects marked with † use the ADD-S metric (symmetric). 56

6.1 KL loss ablation on eggbox (ADD-S accuracy, %). 61

6.2 Mesh resolution ablation on ape (ADD accuracy, %). 63

6.3 Object size effect on accuracy. 64

6.4 Key engineering findings. 67

turnitin Page 14 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

LIST OF FIGURES

1.1 Input RGB image from the LineMOD dataset, showing a cluttered scene with the target object (iron). 2

1.2 Corresponding metric depth map (in metres). The colour scale encodes distance from the camera. 2

3.1 The pinhole camera model (adapted from [1]). A 3D scene point $\mathbf{P} = (X, Y, Z)^\top$ in the camera frame is projected onto the image plane at pixel coordinates $\mathbf{p} = (u, v)^\top$ through the camera centre O. The focal length f is the perpendicular distance from O to the image plane, and (c_x, c_y) is the principal point. 14

4.1 Overall architecture of LightweightPoseNet. The RGB image is processed by a truncated MobileNetV3-Small encoder and the depth image is back-projected and processed by a custom depthwise-separable encoder. Features from both branches are fused at four spatial scales through additive attention modules. The fused features pass through an FPN neck, from which three heads predict segmentation logits, a unit vector field, and per-keypoint confidence scores. A differentiable voting module aggregates the vector field into 2D keypoint locations, and a differentiable LM-style PnP solver recovers the final 6DoF pose. 20

4.2 Predicted unit vector field for keypoint 6. White arrows show the per-pixel direction predictions; the yellow dot marks the voted keypoint location. 33

4.3 Predicted unit vector field for keypoint 7. The arrows converge coherently toward the voted location, demonstrating that the network has learned geometrically meaningful direction predictions. 33

5.1 Object size (ADD threshold) versus pose estimation accuracy across all 13 LineMOD objects. Circles denote asymmetric objects evaluated with ADD; squares denote symmetric objects evaluated with ADD-S. The dashed trend line confirms a strong positive correlation: larger objects face more generous thresholds and achieve higher accuracy. The horizontal dotted line shows the mean accuracy of 68.5%. 57

turnitin Page 14 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 15 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

5.2 Training loss components over epochs for the iron object. The segmentation loss (blue) and vector field loss (orange) decrease rapidly in the first ten epochs. The KL loss (red) is zero during the warmup phase and activates at epoch 10, where it initially spikes before decreasing as the network adapts to end-to-end pose-aware supervision. The shaded regions indicate the warmup and end-to-end training phases. 57

5.3 Predicted segmentation probability map for the iron object, overlaid on the RGB input. Bright regions indicate high foreground probability. The segmentation cleanly isolates the target object from the cluttered background. 58

5.4 Predicted keypoints (green) versus ground-truth keypoints (red) on the iron object. Yellow lines indicate the per-keypoint error. The ADD accuracy for this frame is 90.09%, demonstrating close alignment between the voted keypoint locations and the ground truth. 59

5.5 ADD-S accuracy per epoch for the iron object (Object 13). Accuracy rises steadily during the warmup phase (epochs 1-10), reaching 78.4% by epoch 9. After KL loss activation at epoch 10, accuracy jumps to 84.5% and continues to improve, peaking at 87.8% at epoch 20. The post-KL oscillation (e.g., the dip at epochs 12-14) reflects the interaction between the probabilistic pose objective and the keypoint-level losses as the network balances both supervision signals. 59

6.1 KL loss ablation on the iron object. The solid blue line shows ADD accuracy with the full pipeline (seg + vec + KL); the dashed grey line shows the baseline trained with segmentation and vector field losses only. During the warmup phase (epochs 1-10), both curves overlap because the KL term is inactive. After epoch 10, the full pipeline continues to improve while the baseline plateaus, confirming a $+14.1$ percentage-point contribution from end-to-end differentiable training. 62

6.2 Comparison of voting strategies. **(a) Broken voting:** adding pixel coordinates and unit vectors produces keypoint hypotheses that cluster near the foreground centroid, far from the true keypoint (green star), resulting in large error. **(b) Correct voting:** ray intersection via Cramer's rule produces hypotheses (blue crosses) that converge at the actual keypoint location, yielding sub-millimetre error. 66

turnitin Page 15 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 16 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

LIST OF SYMBOLS, ABBREVIATIONS

Abbreviation	Expansion
6DoF	Six Degrees of Freedom
ADD	Average Distance of Model Points
ADD-S	Average Distance of Model Points, Symmetric variant
AdamW	Adam optimiser with decoupled Weight decay
AMP	Automatic Mixed Precision
BN	Batch Normalisation
CAD	Computer-Aided Design
CBA	Convolution–BatchNorm–Activation block
CNN	Convolutional Neural Network
DW	Depthwise (separable) Convolution
EPnP	Efficient Perspective-n-Point algorithm
EPro-PnP	End-to-end Probabilistic Perspective-n-Point
FPN	Feature Pyramid Network
FPS	Farthest Point Sampling
GN	Group Normalisation
GPU	Graphics Processing Unit
KL	Kullback–Leibler (divergence)
LM	Levenberg–Marquardt
LineMOD	The LINEMOD object pose estimation dataset
LN	Layer Normalisation
MC	Monte Carlo
MLP	Multi-Layer Perceptron
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RGB-D	Red-Green-Blue + Depth
SE(3)	Special Euclidean group in 3D
SO(3)	Special Orthogonal group in 3D
VRAM	Video Random-Access Memory

turnitin Page 16 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Estimating the six degree-of-freedom (6DoF) pose of a rigid object from sensor data is a fundamental problem in computer vision and robotics. The task is to estimate the three dimensional rotation and translation describing the position and orientation of the object with respect to the camera, given one observation of a scene. Accurate and reliable pose estimation is a key to many downstream applications such as robotic grasping and manipulation, augmented reality overlay, autonomous navigation, industrial bin picking and quality inspection on production lines.

Classical approaches to this problem exploited hand-engineered features such as SIFT, SURF and ORB to establish correspondences between a known object model and its observed appearance, followed by geometric solvers such as the Perspective-n-Point (PnP) algorithm to retrieve the pose from such correspondences . These techniques work well for textured objects in controlled environments but the performance drops dramatically when dealing with texture-less surfaces, specular reflections, partial occlusion, clutter and varying illumination conditions that dominate the real-world deployment scenarios.

Now, deep learning has revolutionized the field of pose estimation. Convolutional neural networks are capable of learning discriminative features directly from data without the brittleness of hand-crafted descriptors. Early deep learning methods directly predicted the pose as a regression target, but suffered from the discontinuities of rotation representations and the difficulty of learning a mapping from image pixels to a six dimensional continuous manifold. PVNet [2] suggests a more robust intermediate representation using keypoint methods. Instead of directly regressing the pose, the network predicts a dense field of unit vectors from each visible pixel to a small set of canonical keypoints on the surface of the object. The pose is then computed from the two dimensional keypoint locations by a standard PnP solver. This factorization decouples the network task (mapping local appearance to direction) from what geometry can solve exactly (recovering rigid transformations from point correspondences), leading to improved accuracy and interpretability.

However, even with these advances, there are two notable gaps in the literature

turnitin Page 18 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

to date. First, most state-of-the-art pose estimation systems assume the availability of large computational resources, e.g., high-end GPUs with 8-24 GB of video memory, and process images at resolutions and batch sizes impossible on edge hardware. The proliferation of robotic manipulation and embedded augmented reality systems on resource constrained platforms has created a pressing need for methods that attain competitive accuracy in tight memory and compute budgets. Second, the standard keypoint-based pipeline is limited by a non-differentiable bottleneck at the PnP solver: while keypoints are predicted by a differentiable network, the geometric solver that maps those keypoints to a pose is typically implemented as a black-box routine (e.g. OpenCV's `solvePnPRansac`) thereby severing the gradient path between the final pose output and the learned features of the network. This means that the network cannot be directly optimized to output poses, only to output keypoints, with the hope that good keypoints will result in good poses.

Chen et al. [3] proposed the EPro-PnP framework that bridged the differentiability gap by converting the PnP solver into a probabilistic energy-based model over $SE(3)$. This conversion makes it possible to propagate the gradients from a Kullback-Leibler divergence loss on the pose distribution to the predicted keypoints, and then to the network parameters. However, the original EPro-PnP work used heavy-weight backbones, and was not targeting edge deployment. No emphasis was placed on the fusion of colour and depth modalities.

The present work addresses both gaps at the same time. Figure 1.1 and Figure 1.2 illustrate the input modalities: a cluttered LineMOD scene captured as a synchronised RGB-D pair.



Figure 1.1. Input RGB image from the LineMOD dataset, showing a cluttered scene with the target object (iron).

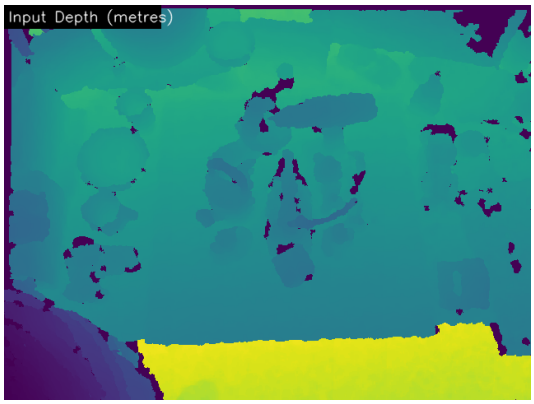


Figure 1.2. Corresponding metric depth map (in metres). The colour scale encodes distance from the camera.

turnitin Page 18 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

1.2 Problem Statement

Given a single RGB-D image containing a known rigid object, estimate the 6DoF pose of the object-three degrees of rotation and three degrees of translation-with accuracy competitive with existing methods, under the constraint that the entire pipeline (network inference, keypoint voting, and pose recovery) must execute within a memory budget of four gigabytes of video RAM.

1.3 Research Objectives

The specific objectives of this work are as follows:

1. To design a lightweight dual-branch encoder that extracts complementary features from RGB and depth modalities using fewer than 1.5 million parameters, based on a truncated MobileNetV3-Small backbone for colour and a custom depthwise-separable network for back-projected depth.
2. To develop an additive attention fusion module that aligns RGB and depth features at multiple scales with linear computational complexity in the spatial extent, avoiding the quadratic cost of standard cross-attention.
3. To implement a fully differentiable keypoint voting procedure based on randomised pairwise ray intersection, enabling end-to-end gradient flow from the pose loss through the voting operation to the network parameters.
4. To implement a differentiable Levenberg-Marquardt-style PnP solver using the continuous six-dimensional rotation representation, eliminating the train-test mismatch that arises from using a non-differentiable solver at inference time.
5. To integrate the EPro-PnP probabilistic objective into the training pipeline, with specific attention to the stability mechanisms (KL warmup, log-partition gradient detachment, confidence-weight regularisation) required for convergence from random initialisation.
6. To evaluate the complete system on the LineMOD benchmark under the standard ADD and ADD-S metrics, and to conduct ablation studies isolating the contributions of the principal architectural and loss components.

1.4 Contributions of This Thesis

The principal contributions of this work are:

1. **LightweightPoseNet**, a complete 6DoF pose estimation architecture with approximately 1.3 million parameters that fuses RGB and depth features through a novel additive attention mechanism operating at four spatial scales with $O(N \cdot d)$ complexity.
2. A **fully differentiable inference pipeline** comprising randomised Cramer's-rule-based keypoint voting and an Adam-based PnP solver with continuous 6D rotation

parameterisation, used identically at both training and inference time.

3. A **stabilised EPro-PnP training strategy** incorporating KL-loss warmup, log-partition gradient detachment, and confidence-weight regularisation, demonstrated to yield a $+16$ percentage-point improvement in ADD-S accuracy over the baseline without the KL loss.

4. A **systematic analysis** of the factors governing pose estimation accuracy on the LineMOD benchmark, demonstrating that the ADD threshold (set as ten percent of the object diameter) is the dominant variable explaining inter-object accuracy variation.

5. **Practical engineering insights** for training pose estimation networks on consumer-grade hardware with four gigabytes of VRAM, including mixed-precision training, gradient accumulation, chunked Monte Carlo estimation, and depth hole-filling strategies.

1.5 Thesis Organisation

The remainder of this thesis is organised as follows.

Chapter 2 reviews the related literature, covering classical and deep-learning-based pose estimation methods, keypoint-based approaches, RGB-D fusion strategies, differentiable geometric solvers, and lightweight backbone architectures.

Chapter 3 presents the mathematical and technical background required for the proposed method, including the pinhole camera model, rigid body transformations and the $SE(3)$ group, the Perspective-n-Point problem, convolutional neural networks for dense prediction, and the LineMOD benchmark.

Chapter 4 describes the proposed methodology in detail, covering the network architecture, the differentiable voting module, the LM-style PnP solver, the composite loss function with the EPro-PnP KL divergence term, and the training strategy.

Chapter 5 presents the experimental evaluation, including per-object quantitative results on the LineMOD benchmark, training behaviour analysis, and comparison with baseline configurations.

Chapter 6 presents ablation studies isolating the contributions of the KL loss, mesh resolution, and object size, and discusses the engineering findings that emerged during development.

Chapter 7 concludes the thesis with a summary of contributions, a discussion of limitations, and directions for future work.

turnitin Page 23 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter surveys the literature relevant to the proposed LightPose system. The review is organised thematically: Section 2.2 covers classical pose estimation methods; Section 2.3 reviews deep-learning-based approaches; Section 2.4 focuses on keypoint-based methods; Section 2.5 examines RGB-D fusion strategies; Section 2.6 reviews differentiable geometric solvers; Section 2.7 discusses lightweight backbone architectures; and Section 2.8 identifies the research gap that the proposed work addresses.

2.2 Classical Pose Estimation

The problem of estimating the pose of a known object from a single image has a long history in computer vision. Template-matching approaches, such as that of Hinterstoisser et al. [4], match precomputed gradient and surface-normal templates against the input image to jointly detect and localise objects. The LineMOD method introduced quantised gradient orientation and surface-normal features for efficient template matching, and the associated dataset has since become the standard benchmark for 6DoF pose evaluation. While template matching is robust to textureless surfaces, it scales poorly with the number of objects and viewpoints, and its accuracy is limited by the discretisation of the template library.

Feature-based methods establish correspondences between local descriptors extracted from the input image and from a reference model or set of reference images. Classical descriptors such as SIFT [5], SURF, and ORB provide scale- and rotation-invariant keypoints that can be matched across viewpoints, after which the pose is recovered through a PnP solver or homography estimation. These methods are effective for richly textured objects but fail on the textureless household and industrial objects that dominate real-world robotic manipulation scenarios. Point-pair features (PPFs) [6] extended feature-based matching to 3D point clouds acquired from depth sensors, achieving robustness to clutter and partial occlusion, but at substantial computational cost.

Iterative closest point (ICP) [7] and its variants refine an initial pose estimate by alternating between correspondence estimation and rigid-body alignment. ICP is

turnitin Page 23 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

widely used as a post-processing refinement step in both classical and learning-based pipelines, but it requires a good initialisation to avoid local minima and is sensitive to outliers and incomplete depth data.

2.3 Deep Learning Approaches to Pose Estimation

The application of deep learning to 6DoF pose estimation began with PoseNet [8], which formulated the problem as direct regression from a single RGB image to a pose vector (quaternion for rotation, three-vector for translation) using a modified GoogLeNet backbone. While conceptually simple, PoseNet and its successors demonstrated that the direct regression of rotation parameters is hampered by the discontinuities in minimal rotation representations (Euler angles, axis-angle) and by the difficulty of balancing the rotation and translation components of the loss function.

BB8 [9] and SSD-6D [10] reformulated pose estimation as a two-stage detection-then-refinement pipeline: an object detector first localises the target in the image, and a subsequent network refines the pose within the detected bounding box. These methods improved accuracy over direct regression but introduced a dependency on the quality of the initial detection.

DenseFusion [11] proposed a per-pixel fusion of colour and depth features for direct 6DoF pose estimation. The method extracts colour features from a CNN applied to the RGB crop and geometric features from a PointNet applied to the corresponding depth point cloud, fuses them at each pixel through concatenation, and iteratively refines the pose through a learned refinement network. DenseFusion demonstrated competitive accuracy on LineMOD and YCB-Video but used a relatively heavyweight architecture (ResNet-18 plus PointNet) that is not suitable for edge deployment.

2.4 Keypoint-Based Methods

Keypoint-based methods decompose pose estimation into two stages: first, localise a set of predefined keypoints in the image; second, recover the pose from the correspondence between the detected 2D keypoints and their known 3D positions on the object model.

PVNet [2] introduced the pixel-wise voting formulation that the proposed work builds upon. For each object pixel, the network predicts a unit direction vector pointing toward each of K canonical keypoints. The keypoint locations are then recovered through a RANSAC-based voting procedure that finds the intersection of the predicted

direction rays. PVNet demonstrated that this dense voting approach is substantially more robust to occlusion than direct keypoint regression, because even a small visible region of the object contributes votes toward the full set of keypoints. The pose is recovered from the detected 2D keypoints through the EPnP algorithm [12] followed by RANSAC outlier rejection.

PVN3D [13] extended the keypoint voting approach to 3D by fusing RGB and point cloud features through a deep Hough voting network, predicting 3D offsets from each point to the keypoints. CDPN [14] combined coordinate regression with a direct pose estimation branch, using the coordinates as a geometric intermediate representation.

HybridPose [15] combined keypoint predictions with edge vectors and symmetry correspondences to provide a richer set of geometric constraints for pose recovery. GDR-Net [16] proposed a geometry-guided direct regression approach that uses dense correspondence maps as an intermediate representation.

A common limitation across these methods is the non-differentiable PnP solver at the boundary between the learned keypoint prediction and the geometric pose recovery. Gradients from the pose error cannot propagate back through the solver to improve the keypoint predictions, limiting the network to optimising a proxy objective (keypoint accuracy) rather than the true target (pose accuracy).

2.5 RGB-D Fusion Strategies

The integration of colour and depth information is a recurring theme in pose estimation. Three broad strategies exist in the literature.

Early fusion concatenates the depth channel (or its back-projected XYZ representation) with the RGB channels as a four- or six-channel input to a single encoder. This approach, used in variants of PoseCNN [17], is simple to implement but forces the network to learn cross-modal correlations from scratch, without structural guidance.

Late fusion processes each modality through a separate encoder and combines the resulting feature vectors (typically through concatenation or addition) at the final layer before the prediction head. DenseFusion [11] uses a variant of this approach, fusing per-pixel colour and point features at a single scale. Late fusion preserves modality-specific representations but limits interaction to a single resolution, which can miss complementary information at intermediate scales.

Multi-scale fusion introduces cross-modal interaction at multiple spatial resolu-

turnitin Page 26 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

tions. FFB6D [18] proposed a full-flow bidirectional fusion network that exchanges information between RGB and point-cloud branches at every encoder layer and every decoder layer. While highly effective, FFB6D's bidirectional fusion at every layer incurs substantial computational overhead.

The proposed additive attention fusion module occupies a middle ground: it operates at four spatial scales (matching the four stages of the encoder hierarchy) but uses a computationally efficient additive attention mechanism rather than full cross-attention, achieving $O(N \cdot d)$ complexity per scale rather than $O(N^2 \cdot d)$.

2.6 Differentiable Geometric Solvers

The non-differentiability of the PnP solver has been recognised as a fundamental bottleneck in keypoint-based pose estimation. Several works have proposed solutions.

DSAC [19] introduced a differentiable version of the RANSAC hypothesis-scoring step by replacing the hard inlier count with a soft probabilistic score, enabling gradients to flow through the scoring but not through the hypothesis generation itself.

EPro-PnP [3] took a more principled approach by modelling the pose as a sample from an energy-based distribution over $SE(3)$, where the energy is defined as the weighted reprojection error of the predicted keypoints. The training objective is the KL divergence between this predicted distribution and a Dirac-delta target at the ground truth, estimated through Monte Carlo sampling. EPro-PnP demonstrated that end-to-end differentiable training through the pose solver yields substantial accuracy improvements over training with only keypoint-level supervision. The present work adopts the EPro-PnP objective as its pose-level loss term.

Zhou et al. [20] addressed the related problem of rotation representation in neural networks, demonstrating that the standard three-parameter representations (Euler angles, axis-angle, quaternion) are discontinuous on $SO(3)$ and proposing a continuous six-dimensional representation based on Gram-Schmidt orthonormalisation. The proposed LM-style PnP solver uses this representation for its internal rotation parameterisation.

2.7 Lightweight Architectures for Edge Deployment

The deployment of deep learning models on resource-constrained hardware has motivated a substantial body of work on efficient network design.

MobileNetV1 [21] introduced depthwise separable convolutions as a drop-in re-

turnitin Page 26 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

placement for standard convolutions, reducing parameter count and computation by approximately an order of magnitude with modest accuracy loss. MobileNetV2 [22] added inverted residual blocks with linear bottlenecks, and MobileNetV3 [23] further improved efficiency through neural architecture search, the hardswish activation function, and squeeze-and-excitation blocks [24].

ShuffleNet [25] and EfficientNet [26] represent alternative points in the accuracy-efficiency trade-off space. For the present work, MobileNetV3-Small was selected as the RGB backbone on the basis of its favourable balance between parameter count ($\sim$0.93M), feature hierarchy structure (clean four-stage downsampling), and availability of ImageNet-pretrained weights.

In the specific context of pose estimation, few works have targeted consumer-grade hardware with four gigabytes of VRAM. Most published systems report results on V100 or A100 GPUs, and the memory requirements of their training pipelines are not discussed. The proposed system is, to the best of the author's knowledge, the first keypoint-based 6DoF pose estimation system to report complete training and evaluation on a GTX 1650-class device.

2.8 Summary and Research Gap

The literature reveals three gaps that the present work addresses:

1. **Lightweight architectures for pose estimation.** While lightweight backbones exist and pose estimation methods exist, their combination-a complete pose pipeline designed from the ground up for edge-scale hardware-has received limited attention. Existing lightweight approaches either sacrifice the depth modality, use a non-differentiable solver, or require substantially more memory than four gigabytes for training.

2. **Efficient cross-modal fusion.** Multi-scale RGB-D fusion with sub-quadratic complexity has not been explored in the pose estimation context. The additive attention mechanism proposed in this work fills this gap.

3. **End-to-end differentiable training on constrained hardware.** The EPro-PnP framework provides the theoretical machinery for end-to-end pose training, but its practical deployment on consumer-grade hardware requires specific engineering (chunked MC estimation, mixed precision, gradient accumulation, warmup schedules) that has not been documented in the literature. This work provides that documentation alongside the experimental validation.

Table 2.1 summarises the positioning of the proposed method relative to the most relevant prior works.

Table 2.1. Comparison of the proposed method with related works.

Method	**RGB-D Fusion**	**Keypoint Voting**	**Diff. PnP**	**Multi-scale Fusion**	**Params (M)**	**Edge Target**
PVNet [2]	No	Yes	No	No	∼25	No
DenseFusion [11]	Yes	No	No	No	∼15	No
EPro-PnP [3]	No	Yes	Yes	No	∼30	No
FFB6D [18]	Yes	Yes	No	Yes	∼35	No
Proposed	Yes	Yes	Yes	Yes	∼1.3	Yes

turnitin Page 29 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 3

BACKGROUND AND PRELIMINARIES

This chapter presents the mathematical and technical foundations upon which the proposed method is built. The material here is not original to this thesis; it is included to make the presentation self-contained and to establish the notation used in subsequent chapters.

3.1 The Pinhole Camera Model

The mapping from a three-dimensional scene point to its two-dimensional image projection is described by the pinhole camera model. A point $\mathbf{p}^{\mathscr{C}} = (X, Y, Z)^\top$ expressed in the camera coordinate frame projects to the pixel coordinate $\mathbf{u} = (u, v)^\top$ according to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z}\mathbf{K}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where f_x, f_y are the focal lengths in pixels and (c_x, c_y) is the principal point. The inverse of this projection, known as back-projection, recovers the 3D camera-frame coordinates from a pixel location and its measured depth Z:

$$\mathbf{p}^{\mathscr{C}}(u, v) = \left(\frac{(u - c_x)Z}{f_x}, \frac{(v - c_y)Z}{f_y}, Z \right)^\top. \tag{3.2}$$

Back-projection is used in the proposed method to convert the depth image into a three-channel point cloud before it enters the depth encoder.

Figure 3.1 illustrates the pinhole camera model geometrically.

3.2 Rigid Body Transformations and $SE(3)$

A rigid body transformation in three-dimensional space is a mapping that preserves distances and handedness. It consists of a rotation followed by a translation. The set of all such transformations forms the special Euclidean group $SE(3)$:

$$SE(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \;\middle|\; \mathbf{R} \in SO(3),\, \mathbf{t} \in \mathbb{R}^3 \right\}, \tag{3.3}$$

turnitin Page 29 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

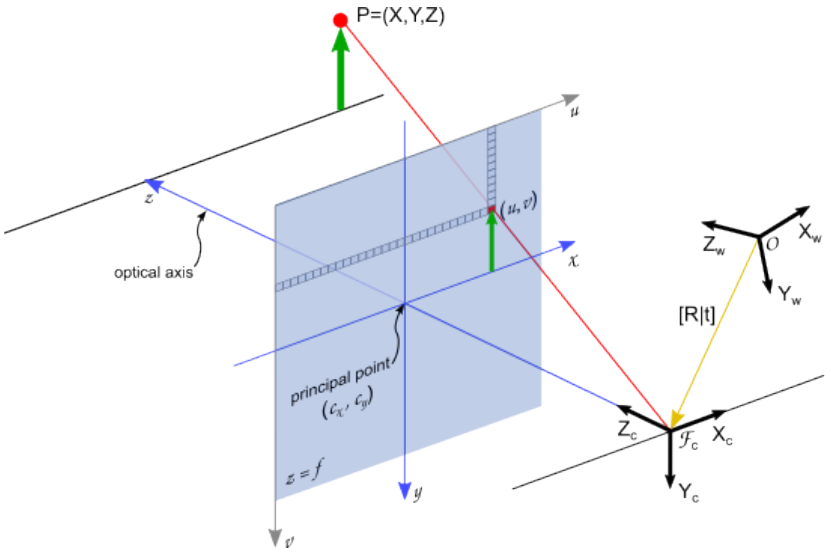


Figure 3.1. The pinhole camera model (adapted from [1]). A 3D scene point $\mathbf{P} = (X, Y, Z)^\top$ in the camera frame is projected onto the image plane at pixel coordinates $\mathbf{p} = (u, v)^\top$ through the camera centre O. The focal length f is the perpendicular distance from O to the image plane, and (c_x, c_y) is the principal point.

where $SO(3)$ is the special orthogonal group, the set of 3×3 real orthogonal matrices with determinant $+1$.

3.2.1 Rotation Representations

Several parameterisations of $SO(3)$ exist, each with different properties relevant to optimisation:

Rotation matrix ($\mathbf{R} \in \mathbb{R}^{3\times3}$): Nine parameters with six constraints ($\mathbf{R}^\top\mathbf{R} = \mathbf{I}$, $\det\mathbf{R} = 1$). Over-parameterised but unambiguous.

Euler angles ($\alpha, \beta, \gamma \in \mathbb{R}$): Three parameters, minimal, but subject to gimbal lock when the middle angle approaches $\pm 90°$.

Axis-angle ($\theta\mathbf{a}$, with $\mathbf{a} \in \mathbb{S}^2$, $\theta \in [0, \pi]$): Three parameters. The conversion to a rotation matrix is given by Rodrigues' formula:

$$\mathbf{R} = \mathbf{I} + \sin\theta\,[\mathbf{a}]_\times + (1 - \cos\theta)\,[\mathbf{a}]_\times^2, \tag{3.4}$$

where $[\mathbf{a}]_\times$ is the 3×3 skew-symmetric matrix associated with $\mathbf{a}$.

Quaternion ($\mathbf{q} \in \mathbb{R}^4$, $\|\mathbf{q}\| = 1$): Four parameters with one constraint. Double cover ($\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation).

Continuous 6D representation [20]: Six parameters, constructed by applying

turnitin Page 31 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

Gram-Schmidt orthonormalisation to two three-vectors to produce the first two columns of a rotation matrix, with the third obtained by cross product. Zhou et al. demonstrated that this representation is continuous everywhere on $SO(3)$, unlike any three- or four-parameter representation, and yields better optimisation behaviour when used as the output of a neural network or the internal parameterisation of a gradient-based solver. The proposed method uses this representation in its differentiable PnP solver.

3.3 The Perspective-n-Point Problem

The Perspective-n-Point (PnP) problem is the recovery of the pose $(\mathbf{R}, \mathbf{t})$ of a calibrated camera given n correspondences between known 3D points $\{\mathbf{p}_k^{\mathcal{O}}\}$ in the object frame and their observed 2D projections $\{\mathbf{u}_k\}$ in the image:

$$\mathbf{u}_k = \pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^{\mathcal{O}}; \mathbf{K}), \qquad k = 1, \ldots, n, \tag{3.5}$$

where π is the pinhole projection of Equation (3.1). The problem is well-determined for $n \geq 4$ (with $n = 3$ admitting a finite number of solutions).

EPnP [12] reformulates the PnP problem as the estimation of four virtual control points that express the 3D points as barycentric combinations, reducing the problem to a linear system solvable in $O(n)$ time.

RANSAC-based approaches repeatedly sample minimal subsets of correspondences, solve the PnP for each, and select the solution with the most inliers. This provides robustness to outliers among the correspondences.

Iterative refinement methods, such as the Levenberg-Marquardt algorithm, minimise the sum of squared reprojection errors starting from an initial estimate. The proposed method uses a gradient-based variant of this approach as its differentiable PnP solver.

3.4 Convolutional Neural Networks for Dense Prediction

The proposed method relies on several standard CNN building blocks, described briefly here for completeness.

3.4.1 Standard and Depthwise Separable Convolutions

A standard 2D convolution with kernel size k, input channels C_{in}, and output channels C_{out} has $k^2 C_{\text{in}} C_{\text{out}}$ parameters. A depthwise separable convolution [21] factorises this

turnitin Page 31 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

into a depthwise convolution ($k^2 C_{\text{in}}$ parameters, filtering each channel independently) followed by a pointwise 1×1 convolution ($C_{\text{in}} C_{\text{out}}$ parameters, mixing channels). The total parameter count is $k^2 C_{\text{in}} + C_{\text{in}} C_{\text{out}}$, a reduction factor of approximately $1/C_{\text{out}} + 1/k^2$ relative to the standard convolution.

3.4.2 Batch Normalisation, Group Normalisation, and Layer Normalisation

Batch normalisation (BN) [27] normalises each channel across the spatial and batch dimensions, stabilising training by reducing internal covariate shift. Group normalisation (GN) [28] normalises across groups of channels within each sample, removing the dependence on batch size. Layer normalisation (LN) [29] normalises across all channels within each sample at each spatial location. The proposed method uses BN in the main encoder and FPN, GN in the fusion module's output, and LN in the fusion module's cross-modal modulation.

3.4.3 Feature Pyramid Networks

Feature pyramid networks (FPNs) [30] address the multi-scale nature of visual recognition by combining features from different stages of a backbone encoder through a top-down pathway with lateral connections. The coarsest features, which carry the strongest semantic signal, are upsampled and added to finer features, which carry the strongest spatial resolution. The result is a set of feature maps at multiple scales, each enriched with both semantic and spatial information.

3.4.4 Attention Mechanisms

The scaled dot-product attention mechanism [31], originally developed for sequence modelling in natural language processing, computes the output as a weighted sum of value vectors, where the weights are determined by the similarity between query and key vectors:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \tag{3.6}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ and d is the embedding dimension. The computational complexity of this operation is $O(N^2 d)$, which is prohibitive for large spatial feature maps. The proposed additive attention fusion module avoids this quadratic cost by replacing the full query-key product with a global attention mechanism that aggregates queries into a single descriptor before cross-modal interaction.

3.5 Focal Loss

The focal loss [32] was proposed for object detection to address the extreme foreground-background class imbalance in dense prediction tasks. For a binary classification problem with predicted probability $\hat{p}$ for the ground-truth class, the focal loss is:

$$\mathrm{FL}(\hat{p}) = -\alpha(1-\hat{p})^{\gamma}\log(\hat{p}), \tag{3.7}$$

where α balances the class prior and γ is the focusing parameter. Setting $\gamma > 0$ reduces the loss contribution from well-classified examples, directing training attention toward hard negatives. The standard settings are $\alpha = 0.25$ and $\gamma = 2$.

3.6 The LineMOD Benchmark

The LineMOD dataset [4] is the standard benchmark for instance-level 6DoF object pose estimation. It contains 13 objects (15 in some extended versions), each captured in cluttered scenes with annotated ground-truth poses. The key characteristics are:

- **Data format:** synchronised RGB and depth images at 640×480 resolution, with per-frame camera intrinsics and ground-truth rotation matrices and translation vectors.
- **Training split:** approximately 186 images per object (from a single recording session).
- **Test split:** 979-1065 images per object (from different recording sessions).
- **CAD models:** triangulated surface meshes provided in PLY format.
- **Evaluation metrics:** ADD for asymmetric objects and ADD-S for symmetric objects, with the standard threshold of 10% of the object diameter.

The ADD metric computes the mean distance between corresponding model points transformed by the predicted and ground-truth poses:

$$\mathrm{ADD} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{p}\in\mathcal{V}} \|(\mathbf{R}_{\mathrm{pred}}\mathbf{p}+\mathbf{t}_{\mathrm{pred}}) - (\mathbf{R}_{\mathrm{gt}}\mathbf{p}+\mathbf{t}_{\mathrm{gt}})\|_2. \tag{3.8}$$

For symmetric objects, the ADD-S metric replaces the corresponding-point dis-

tance with a nearest-point distance:

$$\text{ADD-S} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{p}_i \in \mathcal{V}} \min_{\mathbf{p}_j \in \mathcal{V}} \|(\mathbf{R}_{\text{pred}}\mathbf{p}_i + \mathbf{t}_{\text{pred}}) - (\mathbf{R}_{\text{gt}}\mathbf{p}_j + \mathbf{t}_{\text{gt}})\|_2. \tag{3.9}$$

A prediction is considered correct if the ADD or ADD-S value is below 10% of the object's diameter, where the diameter is defined as the diagonal of the axis-aligned bounding box of the mesh vertices. The reported metric is the percentage of test frames that meet this threshold.

turnitin Page 35 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 4

METHODOLOGY

4.1 Chapter Overview

This chapter presents the complete methodology underlying LightweightPoseNet, the proposed system for estimating the six-degree-of-freedom (6DoF) pose of a known rigid object from a single RGB-D frame. The chapter is organised around the principle of *end-to-end differentiability*: every stage of the pipeline, from feature extraction through keypoint voting to final pose recovery, admits gradient flow, which permits the entire network to be refined jointly under a single probabilistic objective.

Section 4.2 establishes the problem in formal terms and introduces the geometric assumptions that the rest of the pipeline rests on. Section 4.3 surveys the overall data flow. Section 4.4 details the dual-branch backbone architecture, including the additive attention fusion module that aligns RGB and depth features. Section 4.5 describes the encoding of object geometry as a set of canonical keypoints and a dense unit vector field. Section 4.6 presents the differentiable keypoint voting procedure, including the Cramer's rule derivation that makes ray intersection tractable in closed form. Section 4.7 describes the custom Levenberg-Marquardt-style solver used to recover the pose from voted keypoints. Section 4.8 derives the composite loss function in full, with particular attention to the EPro-PnP Kullback-Leibler divergence term and its Monte Carlo estimator. Section 4.9 specifies the training strategy, including the warmup schedule that prevents divergence from random initialisation. Section 4.10 closes the chapter with the evaluation protocol used in the experimental analysis of Chapter 5.

A schematic of the complete pipeline is shown in Figure 4.1.

4.2 Problem Formulation

4.2.1 The 6DoF Object Pose Problem

Let $\mathscr{O}$ denote a rigid object whose three-dimensional geometry is known in the form of a triangulated surface mesh $\mathscr{M} = (\mathscr{V}, \mathscr{F})$, where $\mathscr{V} \subset \mathbb{R}^3$ is the vertex set expressed in the object's canonical coordinate frame. The six-degree-of-freedom pose estimation problem is the recovery of a rigid transformation

turnitin Page 35 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

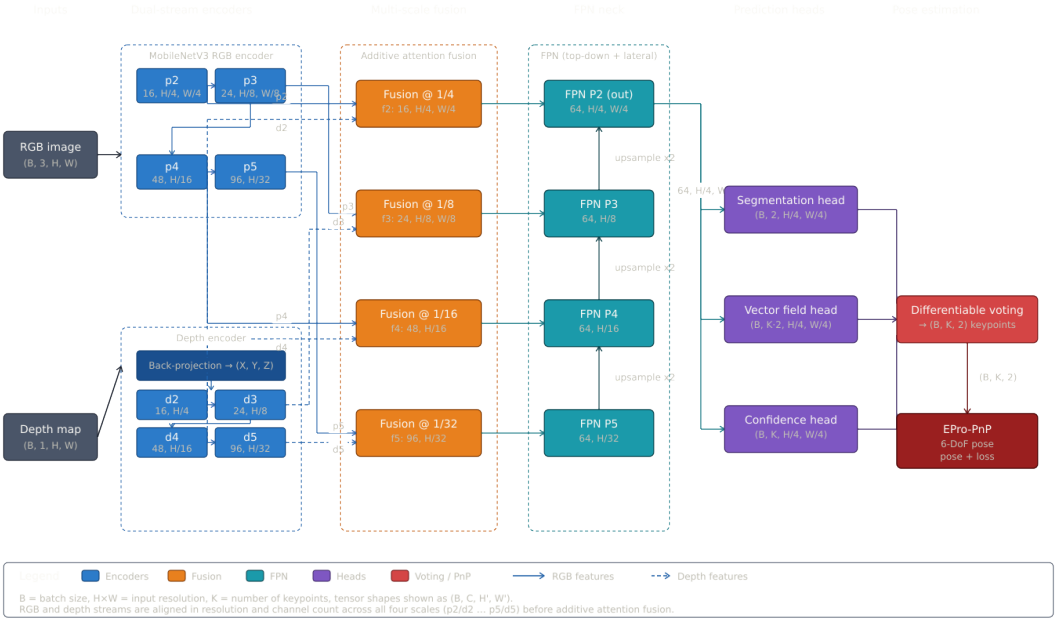


Figure 4.1. Overall architecture of LightweightPoseNet. The RGB image is processed by a truncated MobileNetV3-Small encoder and the depth image is back-projected and processed by a custom depthwise-separable encoder. Features from both branches are fused at four spatial scales through additive attention modules. The fused features pass through an FPN neck, from which three heads predict segmentation logits, a unit vector field, and per-keypoint confidence scores. A differentiable voting module aggregates the vector field into 2D keypoint locations, and a differentiable LM-style PnP solver recovers the final 6DoF pose.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3), \qquad \mathbf{R} \in SO(3), \quad \mathbf{t} \in \mathbb{R}^3,$$

that maps a point $\mathbf{p}^{\mathscr{O}} \in \mathbb{R}^3$ expressed in the object frame to its corresponding location $\mathbf{p}^{\mathscr{C}} = \mathbf{R}\mathbf{p}^{\mathscr{O}} + \mathbf{t}$ in the camera frame.

The special Euclidean group $SE(3)$ is a six-dimensional smooth manifold, where the rotation submanifold $SO(3)$ contributes three degrees of freedom and the translation $\mathbf{t}$ contributes a further three. Although $\mathbf{R}$ is represented numerically by a 3×3 orthogonal matrix with positive determinant (nine entries), the orthogonality and determinant constraints reduce its effective dimensionality to three. The proposed system therefore predicts six independent quantities per input frame, although the internal rotation parameterisation used by the solver, described in Section 4.7, uses six numbers per rotation rather than three or nine, for reasons of optimisation stability.

turnitin Page 37 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

4.2.2 Input Modalities and the Pinhole Camera Model

The system operates on a synchronised pair of images: a colour image $\mathbf{I}_{\text{rgb}} \in \mathbb{R}^{3\times H\times W}$ and a metric depth image $\mathbf{I}_{\text{depth}} \in \mathbb{R}^{1\times H\times W}$ in which each pixel value records the perpendicular distance, in metres, from the camera plane to the nearest surface along the corresponding ray. The image resolution used throughout this work is $H = 480$, $W = 640$.

Image formation follows the pinhole camera model. A point $\mathbf{p}^{\mathscr{C}} = (X, Y, Z)^{\top}$ in the camera frame projects to the pixel coordinate $\mathbf{u} = (u, v)^{\top}$ according to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z}\mathbf{K}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

where $\mathbf{K}$ is the camera intrinsic matrix with focal lengths (f_x, f_y) and principal point (c_x, c_y). Throughout this work the intrinsics are assumed to be known a priori, as is standard in the LineMOD benchmark protocol [4]. The default intrinsics used when frame-specific calibration data is unavailable are $f_x = 572.4$, $f_y = 573.6$, $c_x = 325.3$, $c_y = 242.0$, taken from the original LineMOD calibration.

Inverting (4.1) yields the back-projection equation, which lifts a pixel with measured depth Z back to a three-dimensional point in the camera frame:

$$\mathbf{p}^{\mathscr{C}}(u, v) = \begin{bmatrix} (u - c_x)Z/f_x \\ (v - c_y)Z/f_y \\ Z \end{bmatrix}. \tag{4.2}$$

Equation (4.2) underlies the depth branch of the proposed network, as described in Section 4.4.2.

4.2.3 Design Constraints

The system was designed under three deployment-oriented constraints that shape several architectural choices.

First, the target hardware envelope is a single consumer-grade graphics processor with four gigabytes of video memory, represented in this work by an NVIDIA GeForce GTX 1650. This constraint motivates the use of a lightweight backbone, mixed-precision arithmetic, gradient accumulation, and chunked Monte Carlo estimation, all described in subsequent sections.

turnitin Page 37 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

Second, the system operates on a single RGB-D frame at a time. No temporal smoothing, multi-view fusion, or recurrent state is used. This excludes a class of accuracy improvements that would otherwise be available from sequential information, but it also makes the system applicable to single-shot detection scenarios.

Third, the system assumes the object identity is known and that its CAD mesh is available. Object detection is therefore out of scope: the test-time input is assumed to contain the target object, and the segmentation head learns only to separate foreground from background within that image.

4.3 System Pipeline Overview

The proposed pipeline, instantiated by the `LightweightPoseNet` class in the accompanying implementation, comprises six stages connected in series:

4.3.1 dual-branch encoder

processes the RGB image and the back-projected depth point cloud through two parallel feature extractors. The RGB branch is a truncated MobileNetV3-Small backbone; the depth branch is a custom depthwise-separable network. Both branches emit feature maps at four resolutions: strides of 4, 8, 16, and 32 relative to the input.

4.3.2 multi-scale fusion module

combines the RGB and depth features at each of the four scales through an additive attention mechanism, producing fused feature maps that retain the spatial layout of the RGB branch while incorporating geometric cues from depth.

4.3.3 feature pyramid network (FPN) neck

propagates semantic information from coarse to fine scales via top-down lateral connections, yielding a single feature map at one quarter of the input resolution.

4.3.4 Three prediction heads

operate in parallel on the FPN output: a segmentation head that predicts foreground occupancy, a vector field head that predicts the unit direction from each foreground pixel toward each of K canonical keypoints, and a confidence head that predicts a per-pixel, per-keypoint reliability score.

4.3.5 differentiable voting module

aggregates the predicted vector field into K two-dimensional keypoint locations through randomised pairwise ray intersection in closed form, weighted by the predicted confidence and foreground probability.

4.3.6 differentiable Levenberg-Marquardt-style solver

recovers the pose $(\mathbf{R}, \mathbf{t})$ from the predicted two-dimensional keypoints and their known three-dimensional counterparts on the mesh, by minimising a weighted reprojection error.

During training, an additional EPro-PnP module computes a Kullback-Leibler divergence loss between the predicted pose distribution and a ground-truth-centred target distribution, with gradients flowing back through the entire pipeline. The boundary between learned and geometric components is not at the solver - the solver is itself differentiable - but at the boundary between the learned features (encoders, fusion, FPN, heads) and the parameter-free geometric operations (back-projection, ray intersection, reprojection). This contrasts with two common alternatives: pipelines that use a non-differentiable solver such as `cv2.solvePnPRansac` at both training and inference, and pipelines that use one solver during training and another at inference. The choice to use a single differentiable solver throughout removes any train-test mismatch and is what enables the EPro-PnP loss to refine keypoint predictions through end-to-end gradients.

4.4 Network Architecture

4.4.1 RGB Encoder: Truncated MobileNetV3-Small

The RGB branch uses MobileNetV3-Small [23] as its feature extractor, initialised from ImageNet-pretrained weights provided through the `torchvision` library [33]. MobileNetV3 was selected over alternative lightweight backbones such as MobileNetV2, ShuffleNetV2 and EfficientNet-B0 on the basis of three properties relevant to dense prediction: the use of hardswish activation [23], which is differentiable everywhere and computationally efficient; the inclusion of squeeze-and-excitation blocks [24] that introduce channel-wise attention with negligible parameter cost; and a feature hierarchy that provides clean downsampling boundaries at strides 4, 8, 16, and 32, matching the resolution requirements of the downstream FPN.

The MobileNetV3-Small `features` sequence consists of twelve inverted residual blocks. The proposed encoder partitions these into four stages, exposing intermediate feature maps as outputs:

Stage	Block indices	Output stride	Output channels
1	0-1	4	16
2	2-3	8	24
3	4-8	16	48
4	9-11	32	96

The choice of block indices warrants comment. The MobileNetV3-Small `features` module contains thirteen entries, indexed 0 through 12, the last of which is a 576-channel expansion layer feeding the classification head. Including block 12 in stage 4 raises the channel count from 96 to 576, which inflates the parameter budget of the downstream fusion and FPN modules by approximately an order of magnitude. The proposed encoder therefore stops at block 11, yielding the 96-channel output used in the original MobileNetV3 paper as the input to the classifier's expansion. This is consistent with how MobileNetV3 is most commonly truncated for dense-prediction transfer.

A forward pass through the encoder yields the four-tuple of feature maps $(\mathbf{f}_2^r, \mathbf{f}_3^r, \mathbf{f}_4^r, \mathbf{f}_5^r)$, where the subscript denotes the binary-logarithmic scale factor relative to the input (stride 2^s).

4.4.2 Depth Branch: Back-Projection and Encoding

The depth branch performs two operations in sequence. It first back-projects the metric depth image into a three-channel point cloud using equation (4.2), and then passes that point cloud through a custom convolutional encoder.

Back-projection

The back-projection is implemented in the static `_back_project` of `DepthEncoder`. For each pixel (u, v) in the depth image, with measured depth $Z(u,v) = \mathbf{I}_{\text{depth}}(0, v, u)$ and per-frame intrinsics $\mathbf{K}$, the back-projection computes

$$\mathbf{p}^{\mathscr{C}}(u,v) = \left(\frac{(u - c_x) \cdot Z(u,v)}{f_x}, \frac{(v - c_y) \cdot Z(u,v)}{f_y}, Z(u,v) \right)^{\top}. \tag{4.3}$$

The result is a three-channel tensor of shape $3 \times H \times W$ that encodes the per-pixel three-dimensional camera-frame position. Encoding depth as XYZ rather than as

turnitin Page 41 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

a single-channel depth map preserves metric scale information that would otherwise have to be re-learned from depth alone, and it makes the geometric meaning of each channel explicit. This is a known design choice in RGB-D representation learning [34, 35].

The implementation supports a batched intrinsics matrix $\mathbf{K}$, so that each frame in a mini-batch can use its own calibration values. This is required by the LineMOD dataset, in which the intrinsics vary modestly across recording sessions.

Depth encoder architecture

The back-projected point cloud is fed through a stem followed by five depthwise-separable convolution stages. The stem is a 1×1 convolution that lifts the three-channel input to sixteen channels, followed by another 1×1 convolution. Each subsequent stage halves the spatial resolution and progressively increases the channel count, producing four output scales whose channel counts $(16, 24, 48, 96)$ are deliberately matched to those of the RGB branch to permit element-wise fusion at each scale.

The depthwise-separable convolution block, denoted `_dw_block`, replaces a standard 3×3 convolution with a sequence of two operations: a depthwise 3×3 convolution that filters each channel independently, followed by a pointwise 1×1 convolution that mixes channels. For an input with C_{in} channels and an output with C_{out} channels, the depthwise-separable form requires $9C_{\text{in}} + C_{\text{in}}C_{\text{out}}$ parameters, compared to $9C_{\text{in}}C_{\text{out}}$ for a standard convolution. The reduction factor is

$$\rho = \frac{9C_{\text{in}} + C_{\text{in}}C_{\text{out}}}{9C_{\text{in}}C_{\text{out}}} = \frac{1}{C_{\text{out}}} + \frac{1}{9}. \tag{4.4}$$

For $C_{\text{out}} = 96$ this evaluates to approximately 0.121, an eight-fold reduction in parameter count for the equivalent operation. Each `_dw_block` uses batch normalisation and hardswish activation after both the depthwise and the pointwise convolution.

Global context refinement

After each of stages 2 through 5, a global-context residual is added to the feature map. Letting $\mathbf{F}_s$ denote the feature map at stage s, the refinement is

$$\hat{\mathbf{F}}_s = \mathbf{F}_s + \mathbf{W}_s^{\text{ctx}} \left(\text{maxpool}_{H,W}(\mathbf{F}_s) \right), \tag{4.5}$$

where $\text{maxpool}_{H,W}$ is the spatial max-pool over the full feature map and $\mathbf{W}_s^{\text{ctx}}$ is a learnable 1×1 convolution. This is implemented in the static method `_ctx`. The

turnitin Page 41 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 42 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

global-context channel-wise vector, broadcast across the spatial grid and added to the feature map, allows each pixel to access object-level statistics that are otherwise outside its local receptive field. The use of max-pool rather than average-pool selects the strongest activation per channel, which empirically yielded more stable training for depth features dominated by occasional high-magnitude responses near object boundaries.

The depth branch thus emits the four-tuple of feature maps $(\mathbf{f}_2^d, \mathbf{f}_3^d, \mathbf{f}_4^d, \mathbf{f}_5^d)$, matching in resolution and channel count the outputs of the RGB branch.

4.4.3 Additive Attention Fusion

The fusion of the two modalities is the core architectural contribution of Lightweight-PoseNet. Standard cross-attention between two feature maps each with $N = HW$ spatial locations has time and memory complexity $O(N^2 \cdot d)$ for embedding dimension d, which is prohibitive at the largest fused scale where $N = 120 \cdot 160 = 19{,}200$. The proposed `AdditiveAttentionFusion` module achieves a related effect at $O(N \cdot d)$ complexity by replacing the quadratic query-key product with a learnable global query.

Module formulation

Let $\mathbf{f}^r, \mathbf{f}^d \in \mathbb{R}^{B\times C\times H\times W}$ denote the RGB and depth feature maps at a given scale. The fusion proceeds in five stages.

Stage 1 - Query and key projection. Two pointwise convolutions project the RGB and depth features into a shared embedding space of dimension $d = C$:

$$\mathbf{Q} = \mathbf{W}_Q \mathbf{f}^r, \qquad \mathbf{K} = \mathbf{W}_K \mathbf{f}^d. \tag{4.6}$$

Stage 2 - Global attention weight. Rather than computing a full $N \times N$ attention matrix, a single learnable global query vector $\boldsymbol{\ell}_g \in \mathbb{R}^{1\times d\times 1\times 1}$ is introduced. The unnormalised attention logit at each spatial location $p = (u, v)$ is the scaled inner product

$$g(p) = \frac{\langle \mathbf{Q}(p), \boldsymbol{\ell}_g \rangle}{\sqrt{d}}, \tag{4.7}$$

and these logits are normalised by a single softmax over all spatial locations to produce a probability distribution $g(p) \in [0, 1]$ with $\sum_p g(p) = 1$. The scaling factor $d^{-1/2}$ follows the standard scaled dot-product attention rationale [31]: it prevents the softmax from saturating as the embedding dimension grows.

turnitin Page 42 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

Stage 3 - Globally pooled query. The attention weights $g(p)$ are used to pool the projected RGB features into a single channel-wise descriptor:

$$\mathbf{q} = \sum_{p} g(p) \cdot \mathbf{Q}(p) \in \mathbb{R}^{B \times d \times 1 \times 1}. \tag{4.8}$$

This is the only place in the module where information is aggregated across the full spatial extent. Because g is a single scalar per location rather than a matrix, the operation is $O(N \cdot d)$.

Stage 4 - Cross-modal modulation. The pooled descriptor $\mathbf{q}$ is broadcast and used to modulate the depth keys through element-wise multiplication, followed by a learnable projection $\mathbf{T}$ back to the input channel dimension:

$$\mathbf{X} = \mathrm{LayerNorm}(\mathbf{Q}) + \mathbf{T}(\mathbf{K} \odot \mathbf{q}), \tag{4.9}$$

where $\odot$ denotes per-channel broadcasting multiplication. Layer normalisation is applied to the projected query along the channel dimension before residual addition, which empirically stabilised training when depth values were noisy. Note that $\mathbf{T}$ here is a learnable 1×1 convolution, denoted by the same symbol throughout, and is distinct from the rigid transformation matrix of Section 4.2.

Stage 5 - Feed-forward refinement. A short feed-forward network combines $\mathbf{X}$ with an auxiliary RGB feature $\mathbf{A} = \mathrm{GroupNorm}(\mathbf{W}_A \mathbf{f}^r)$, where $\mathbf{W}_A$ is a learnable 1×1 convolution and group normalisation uses a single group. The two are concatenated along the channel dimension, passed through a two-layer convolutional MLP with hidden dimension $\max(C, 4)$ and GELU activation, and added residually to the original RGB feature, with a final group normalisation:

$$\mathbf{M} = \mathrm{MLP}([\mathbf{X}; \mathbf{A}]), \qquad \mathbf{f}^{\text{fused}} = \mathrm{GroupNorm}(\mathbf{f}^r + \mathbf{M}). \tag{4.10}$$

The residual connection on $\mathbf{f}^r$ ensures that, in the limit where the fusion module learns to zero its output, the network degenerates to an RGB-only pipeline. This is a useful property for stable training from random initialisation in the depth-encoder weights.

Complexity analysis

The dominant computational costs of the fusion module are the four 1×1 convolutions and the MLP. The total floating-point operation count is approximately

$$\text{FLOPs} \approx 4C^2HW + 2Ch_{\text{ffn}}HW, \tag{4.11}$$

where h_{ffn} is the MLP hidden dimension. This scales as $O(N \cdot d)$ in the spatial extent N and the channel count d, as claimed. By contrast, standard cross-attention between $\mathbf{Q}$ and $\mathbf{K}$ would incur an additional N^2d FLOPs from the query-key product and a further N^2d from the attention-value product. At the largest fused scale ($N = 19{,}200$, $C = 16$), this would amount to approximately 1.2×10^{10} additional FLOPs per forward pass per scale - well beyond what the target hardware can sustain.

Multi-scale instantiation

The fusion is applied independently at each of the four scales, by the wrapper module `MultiscaleFusion`, which instantiates one `AdditiveAttentionFusion` per scale with the appropriate channel count. Because the modules at different scales do not share parameters, each scale learns its own query vector $\boldsymbol{\ell}_g$ and its own projection weights. This is consistent with the observation that early-stage and late-stage features encode different aspects of the input (texture and edges at fine scales; semantic regions at coarse scales) and therefore benefit from scale-specific fusion behaviour.

4.4.4 Feature Pyramid Neck

The fused multi-scale features are aggregated into a single feature map through a feature pyramid network (FPN) [30]. The neck performs top-down propagation: features at the coarsest scale, which carry the strongest semantic signal, are upsampled and added to features at progressively finer scales, which carry the strongest spatial resolution. The result is a single high-resolution feature map enriched with semantic context.

Let $\mathbf{f}_s^{\text{fused}}$ denote the fused feature map at scale $s \in \{2,3,4,5\}$. The neck first applies lateral 1×1 convolutions $\mathbf{W}_s^{\text{lat}}$ to project each feature map to a common channel dimension $C_{\text{fpn}} = 64$, with batch normalisation applied immediately after:

$$\mathbf{m}_s^{\text{lat}} = \text{BN}_s(\mathbf{W}_s^{\text{lat}}\mathbf{f}_s^{\text{fused}}), \qquad s \in \{2,3,4,5\}. \tag{4.12}$$

The top-down pathway is then defined recursively, starting from the coarsest scale:

$$\mathbf{m}_5 = \mathbf{m}_5^{\text{lat}}, \qquad \mathbf{m}_s = \text{Smooth}_s\Big(\text{Up}(\mathbf{m}_{s+1}) + \mathbf{m}_s^{\text{lat}}\Big), \quad s \in \{2,3,4\}, \tag{4.13}$$

turnitin Page 45 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

where $\mathrm{Up}(\cdot)$ denotes bilinear upsampling to the spatial resolution of the target scale, and Smooth_s is a 3×3 convolution with batch normalisation and hardswish activation that mitigates the aliasing artefacts introduced by upsampling. The final output of the neck is $\mathbf{m}_2 \in \mathbb{R}^{B \times 64 \times (H/4) \times (W/4)}$, namely $64 \times 120 \times 160$ for the standard input resolution.

The choice of $H/4 \times W/4$ output resolution is a deliberate trade-off. Predicting at full resolution would yield finer keypoint localisation but quadruple the memory footprint and slow training proportionally; predicting at $H/8$ would halve the spatial precision available to the voting module, which would in turn limit the achievable pose accuracy regardless of model capacity. The $H/4$ resolution preserves enough spatial precision that the voting module can place keypoints to sub-pixel accuracy in the original image (recovered by the factor of 4 multiplication described in Section 4.6) while keeping activations within the memory budget.

4.4.5 Prediction Heads

Three parallel heads operate on the FPN output $\mathbf{m}_2$, producing the segmentation logits, the unit vector field, and the per-keypoint confidence map. Each head is a shallow convolutional sub-network, designed to add minimal capacity over and above the shared backbone.

Segmentation head

The segmentation head predicts a two-channel logit map distinguishing foreground (object) from background:

$$\mathbf{s} = \mathrm{Conv}_{1\times1}^{(C_{\mathrm{fpn}}/2 \to 2)} \circ \mathrm{CBA} \circ \mathrm{CBA}(\mathbf{m}_2), \tag{4.14}$$

where CBA denotes a convolution-batch-normalisation-hardswish block. The output $\mathbf{s} \in \mathbb{R}^{B \times 2 \times (H/4) \times (W/4)}$ is supervised with a focal loss, derived in Section 4.8.2.

Vector field head

The vector field head predicts, for each foreground pixel and each of the $K = 12$ keypoints, a unit two-dimensional direction vector pointing from the pixel toward the projected location of the keypoint. Letting $\mathbf{u}_{k,p}^{\mathrm{pred}} \in \mathbb{R}^2$ denote the predicted direction at pixel p for keypoint k, the head produces $K \times 2 = 24$ output channels through three CBA blocks followed by a 1×1 convolution. The raw output is reshaped to $(B, K, 2, H/4, W/4)$, and L2 normalised along the two-dimensional vector axis:

turnitin Page 45 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

$$\mathbf{u}_{k,p}^{\text{pred}} = \frac{\tilde{\mathbf{u}}_{k,p}}{\max\left(\|\tilde{\mathbf{u}}_{k,p}\|_2, \varepsilon\right)}, \qquad \varepsilon = 10^{-8}, \tag{4.15}$$

where $\tilde{\mathbf{u}}_{k,p}$ is the raw convolutional output and the epsilon clamp prevents division by zero. This normalisation is performed *inside* the head's forward pass rather than as a post-processing step, which ensures that gradients propagated through the normalised vectors automatically include the Jacobian of the unit-vector map. The output is reshaped back to a 24-channel feature map for compatibility with the downstream voting module.

Confidence head

The confidence head predicts a scalar reliability score for each keypoint at each pixel:

$$c_{k,p} = \sigma(\text{Conv}_{1\times1}^{(C_{\text{fpn}}/2\to K)} \circ \text{CBA}(\mathbf{m}_2))_{k,p} + 0.01, \tag{4.16}$$

where σ is the logistic sigmoid. The additive floor of 0.01 prevents the predicted confidence from collapsing to exactly zero, which would cause numerical issues in the weighted aggregation used by the voting module (division-by-zero in the limit, and gradient vanishing in practice). The output lies in the interval $[0.01, 1.01]$.

Parameter budget

The complete LightweightPoseNet, including all four scales of fusion, the FPN, and the three heads, contains approximately 1.3 million trainable parameters. Approximately 0.93 million of these reside in the MobileNetV3-Small backbone, with the remainder distributed across the depth encoder, fusion modules, FPN, and heads.

4.5 Keypoint Representation

The proposed system predicts the pose indirectly, by first localising a small set of keypoints anchored on the object's surface and then solving for the rigid transformation that aligns the observed two-dimensional projections of those keypoints with their known three-dimensional positions on the canonical mesh. The choice of keypoint set, and the way each keypoint is encoded in the vector field, are described in this section.

4.5.1 Farthest Point Sampling on the Object Mesh

For each object, a set of $K = 12$ canonical three-dimensional keypoints $\mathbf{P} = \{\mathbf{p}_1^{\mathcal{O}}, \ldots, \mathbf{p}_K^{\mathcal{O}}\} \subset \mathbb{R}^3$ is selected from the vertices of the object's CAD mesh through farthest point sampling (FPS) [36], a greedy procedure that approximates the maximally separated subset of a point cloud.

The algorithm, given a vertex set $\mathcal{V} \subset \mathbb{R}^3$ and a target count K, proceeds as follows:

Algorithm 1 Farthest Point Sampling

Require: Vertex set $\mathcal{V} \subset \mathbb{R}^3$ with $|\mathcal{V}| = N$; number of keypoints K
Ensure: Selected keypoint set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_K\} \subset \mathcal{V}$
$\mathbf{p}_1 \leftarrow \arg\min_{\mathbf{v} \in \mathcal{V}} \|\mathbf{v} - \text{centroid}(\mathcal{V})\|$ ▷ first keypoint nearest centroid
$d[\mathbf{v}] \leftarrow \infty$ for all $\mathbf{v} \in \mathcal{V}$
for $i = 2$ to K **do**
    $d[\mathbf{v}] \leftarrow \min(d[\mathbf{v}], \|\mathbf{v} - \mathbf{p}_{i-1}\|)$ for all $\mathbf{v} \in \mathcal{V}$
    $\mathbf{p}_i \leftarrow \arg\max_{\mathbf{v} \in \mathcal{V}} d[\mathbf{v}]$
end for
return $\{\mathbf{p}_1, \ldots, \mathbf{p}_K\}$

The first keypoint is initialised as the mesh vertex closest to the centroid of $\mathcal{V}$ rather than as a random sample, which yields deterministic and reproducible keypoint sets across runs. Each subsequent keypoint maximises the minimum distance to all previously selected keypoints, which spreads the keypoints uniformly over the mesh surface and reduces the geometric degeneracy of the resulting PnP problem.

The selected keypoint set is converted from millimetres (the units of the LineMOD CAD models) to metres by division by 1000, which aligns its scale with the depth measurements and the camera-frame coordinates produced by the depth back-projection. The resulting $\mathbf{P} \in \mathbb{R}^{K \times 3}$ tensor is held fixed throughout training and evaluation; it is a property of the object, not of the network.

The choice of $K = 12$ balances two competing concerns. The Perspective-n-Point problem is well-determined for $K \geq 4$, and additional correspondences provide robustness against individual keypoint errors but with diminishing returns. Empirically, the literature on keypoint-based pose estimation, in particular the original PVNet work [2], has converged on values in the range eight to fifteen. Twelve keypoints lies near the middle of this range and matches the configuration used in several reference implementations.

4.5.2 Unit Vector Field Encoding

Rather than regressing the two-dimensional projected keypoint locations $\mathbf{kp}_k^{2D} \in \mathbb{R}^2$ directly - a sparse target supervised at only K pixels per image - the system regresses a dense unit direction field defined at every foreground pixel.

For each ground-truth foreground pixel $p \in \mathscr{F}$ and each keypoint k, the supervision target is the unit vector pointing from p to the projected keypoint:

$$\mathbf{u}_{k,p}^{\text{gt}} = \frac{\mathbf{kp}_k^{2D} - p}{\|\mathbf{kp}_k^{2D} - p\|_2}. \tag{4.17}$$

This encoding has three desirable properties.

First, supervision is *dense*: each foreground pixel contributes a gradient signal for every keypoint, yielding approximately $|\mathscr{F}| \cdot K$ regression targets per image rather than K. For a typical LineMOD object occupying a few thousand pixels, this is a four-orders-of-magnitude increase in supervised samples per image.

Second, the unit-norm constraint *decouples direction from distance*. The network is not required to learn the distance to each keypoint - a quantity that varies widely depending on object scale and camera distance - but only the direction, which is a more compact and scale-invariant target.

Third, the encoding is *robust to occlusion*: even if a substantial fraction of the object's foreground is occluded, the remaining visible pixels still vote toward the same keypoint location, and a well-calibrated voting procedure can recover the keypoint despite partial visibility. This robustness property was the original motivation for the PVNet formulation [2], on which the proposed encoding is modelled.

The ground-truth vector fields are pre-computed for each augmented training sample and stored as $K \times 2 \times (H/4) \times (W/4)$ tensors on disk, in the `vec_field/` subdirectory of the augmented dataset. This avoids recomputing the field on every training step, at the cost of some additional disk space. Figure 4.2 and Figure 4.3 illustrate the predicted unit vector fields for two keypoints on the iron object. The white arrows indicate the predicted direction from each foreground pixel toward the keypoint, and the yellow dot marks the voted keypoint location recovered by the differentiable voting procedure described in the next section.

turnitin Page 49 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276



Figure 4.2. Predicted unit vector field for keypoint 6. White arrows show the per-pixel direction predictions; the yellow dot marks the voted keypoint location.



Figure 4.3. Predicted unit vector field for keypoint 7. The arrows converge coherently toward the voted location, demonstrating that the network has learned geometrically meaningful direction predictions.

4.6 Differentiable Keypoint Voting

The vector field predicted by the network specifies, at each foreground pixel, the direction toward each keypoint, but it does not directly specify the keypoint *location*. The voting module aggregates these direction predictions into a concrete two-dimensional keypoint estimate for each of the K keypoints. The aggregation must satisfy three properties: it must be robust to noisy predictions at individual pixels; it must remain tractable on the target hardware; and, crucially, it must be *differentiable*, so that gradients from downstream losses can propagate back through the voting operation to the vector field head and the rest of the network.

The proposed voting procedure, implemented by the function `differentiable_vote`, satisfies all three. It is based on randomised pairwise ray intersection in closed form, weighted by predicted confidence and segmentation probability. Unlike the discrete inlier-counting variant used in the original PVNet [2], the proposed voting is fully differentiable end-to-end.

4.6.1 The Voting Hypothesis

The geometric premise of the voting procedure is straightforward. If two foreground pixels p_a and p_b both predict perfect unit vectors toward the same keypoint $\mathbf{kp}_k^{2D}$, then the two rays $\mathbf{r}_a(t) = p_a + t\mathbf{v}_a$ and $\mathbf{r}_b(t) = p_b + t\mathbf{v}_b$ intersect exactly at $\mathbf{kp}_k^{2D}$. When the predictions are noisy, the rays from many pixel pairs will intersect at locations distributed around the true keypoint, and a suitable weighted aggregate of these inter-

turnitin Page 49 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

sections approximates the keypoint location.

The voting procedure therefore samples a fixed number of random pixel pairs, computes the closed-form intersection for each pair, and aggregates the resulting hypotheses. Each step is detailed below.

4.6.2 Closed-Form Ray Intersection via Cramer's Rule

For two rays in two-dimensional space, parameterised as

$$\mathbf{r}_a(t_a) = p_a + t_a \mathbf{v}_a, \qquad \mathbf{r}_b(t_b) = p_b + t_b \mathbf{v}_b, \tag{4.18}$$

with $p_a, p_b, \mathbf{v}_a, \mathbf{v}_b \in \mathbb{R}^2$, the intersection condition $\mathbf{r}_a(t_a) = \mathbf{r}_b(t_b)$ gives a linear system in the parameters (t_a, t_b):

$$p_a + t_a \mathbf{v}_a = p_b + t_b \mathbf{v}_b \iff t_a \mathbf{v}_a - t_b \mathbf{v}_b = p_b - p_a. \tag{4.19}$$

Writing $\mathbf{d} = p_b - p_a$ and decomposing component-wise, this becomes the 2×2 system

$$\begin{bmatrix} v_a^x & -v_b^x \\ v_a^y & -v_b^y \end{bmatrix} \begin{bmatrix} t_a \\ t_b \end{bmatrix} = \begin{bmatrix} d^x \\ d^y \end{bmatrix}. \tag{4.20}$$

The system admits a unique solution if and only if the determinant of the coefficient matrix is non-zero:

$$\Delta = v_a^x \cdot (-v_b^y) - v_a^y \cdot (-v_b^x) = v_a^y v_b^x - v_a^x v_b^y. \tag{4.21}$$

The condition $\Delta = 0$ holds precisely when the two ray directions are parallel, in which case either the rays do not intersect (parallel and offset) or they coincide everywhere (parallel and collinear). Both cases yield no well-defined intersection point.

By Cramer's rule, the parameter t_a is given by

$$t_a = \frac{1}{\Delta} \det \begin{bmatrix} d^x & -v_b^x \\ d^y & -v_b^y \end{bmatrix} = \frac{d^x \cdot (-v_b^y) - d^y \cdot (-v_b^x)}{\Delta}. \tag{4.22}$$

Substituting t_a back into the parameterisation of $\mathbf{r}_a$ yields the intersection hypothesis:

$$\mathbf{h}_{a,b}^{(k)} = p_a + t_a \mathbf{v}_a. \tag{4.23}$$

Note that only t_a is needed; the symmetric expression for t_b is not used.

Numerical safeguards

For numerical stability when Δ is close to zero, the implementation replaces Δ with $\text{sign}(\Delta) \cdot \max(|\Delta|, \tau)$ where $\tau = 10^{-4}$, ensuring that the division does not overflow. Pairs with $|\Delta| \leq 10^{-3}$, corresponding to near-parallel rays, are subsequently down-weighted to zero in the aggregation step, so that the slight perturbation introduced by clamping the determinant does not affect the final result. Additionally, the computed hypothesis is clamped to a generous bounding box of $[-W/4, 2W/4]^2$ in feature-map coordinates, which excludes hypotheses arising from gross misprediction that would otherwise contribute large outliers to the aggregation.

4.6.3 Confidence-Weighted Aggregation

For each keypoint k, the voting procedure samples $N_{\text{pairs}} = 64$ pixel index pairs uniformly at random from the full feature-map grid of size $HW = (H/4)(W/4) = 19{,}200$ locations. For each pair (p_a, p_b), the intersection hypothesis $\mathbf{h}_{a,b}^{(k)}$ is computed by equations (4.21)-(4.23). The aggregation is then a weighted mean of these hypotheses.

The per-pair weight combines four factors:

$$w_{a,b}^{(k)} = c_{k,p_a} \cdot c_{k,p_b} \cdot \sigma_{\text{seg}}(p_a) \cdot \sigma_{\text{seg}}(p_b) \cdot \mathbb{1}[|\Delta| > 10^{-3}], \tag{4.24}$$

where $c_{k,p}$ is the confidence head output at pixel p for keypoint k, and $\sigma_{\text{seg}}(p)$ is the softmax probability of the foreground class at pixel p. The product $c_{k,p_a} c_{k,p_b}$ down-weights pairs in which either endpoint pixel has low predicted confidence for keypoint k. The product $\sigma_{\text{seg}}(p_a)\sigma_{\text{seg}}(p_b)$ restricts the aggregation effectively to foreground pixels, even though the random sampling is over the full feature map. The indicator function $\mathbb{1}[|\Delta| > 10^{-3}]$ discards pairs with near-parallel rays.

The keypoint estimate at the feature-map resolution is then the weighted mean:

$$\mathbf{kp}_k^{2D,\text{feat}} = \frac{\sum_{(a,b)} w_{a,b}^{(k)} \mathbf{h}_{a,b}^{(k)}}{\max\left(\sum_{(a,b)} w_{a,b}^{(k)}, 10^{-6}\right)}. \tag{4.25}$$

The denominator clamp again prevents division by zero in the degenerate case

where every pair is discarded. Finally, the feature-map estimate is upscaled to the original image resolution by multiplication by the FPN downscaling factor:

$$\mathbf{kp}_k^{2D} = 4 \cdot \mathbf{kp}_k^{2D,\text{feat}}. \tag{4.26}$$

4.6.4 Per-Keypoint Weight for the PnP Solver

In addition to the keypoint locations, the voting module computes an aggregate weight ω_k for each keypoint, which is later used by the LM PnP solver to down-weight residuals from unreliable keypoints. The weight is computed as a confidence-weighted average over foreground pixels:

$$\omega_k = \frac{\sum_p c_{k,p} \cdot \sigma_{\text{seg}}(p)}{\max\left(\sum_p \sigma_{\text{seg}}(p),\, 10^{-6}\right)}. \tag{4.27}$$

The output of the voting module is the pair $(\{\mathbf{kp}_k^{2D}\}_{k=1}^K, \{\omega_k\}_{k=1}^K)$. The weight is broadcast to a $(K,2)$ tensor for compatibility with the per-coordinate reprojection error.

4.6.5 On the Differentiability of the Aggregation

Every operation in the voting procedure - the gather operations that extract pixel-wise vectors and confidences at the sampled indices, the determinant computation, the clamped division by Δ, the multiplication and summation in the weighted mean, and the upscaling - is a smooth function of the network outputs everywhere except on a measure-zero set. Gradients therefore flow from the voted keypoint locations back through the vector field $\mathbf{v}$ and the confidence map c, and onward to the network parameters. This property is what enables the EPro-PnP loss derived in Section 4.8.4 to refine keypoint predictions through end-to-end gradient descent.

The only stochastic component is the choice of pixel index pairs (p_a, p_b), which is sampled fresh on every forward pass from a uniform distribution over $\{0,\ldots,HW-1\}^2$ using `torch.randint`. This stochasticity introduces a variance into the voted keypoint locations that decreases as $1/\sqrt{N_{\text{pairs}}}$, and is treated as a noise process that the downstream solver and loss are expected to be robust to. The pseudorandom number generator is seeded once per run (with the default seed 42) to ensure reproducible evaluation, while training accepts the variance as a mild regularisation.

turnitin Page 53 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

4.7 Pose Recovery via Differentiable Levenberg-Marquardt PnP

Given the predicted two-dimensional keypoint locations $\{\mathbf{kp}_k^{2D}\}$ and their known three-dimensional counterparts $\{\mathbf{p}_k^{\mathcal{O}}\}$ on the canonical mesh, the final stage of the pipeline solves the Perspective-n-Point (PnP) problem to recover the rigid transformation $(\mathbf{R}, \mathbf{t})$.

A central design decision distinguishes the proposed system from many keypoint-based pose pipelines: the PnP solver used here is fully differentiable and is used at *both* training and inference time. The standard alternative would be to use a non-differentiable solver such as OpenCV's `solvePnPRansac` (EPnP [12] followed by Levenberg-Marquardt refinement) at inference, with no solver at all during training. This was the approach taken in the original PVNet implementation. The proposed system uses, instead, a custom gradient-based optimiser implemented in pure PyTorch, which serves three purposes simultaneously: it provides the pose estimates required for evaluation, it furnishes the proposal pose around which the EPro-PnP Monte Carlo sampling is centred during training, and it eliminates any train-test mismatch in the geometric back-end.

4.7.1 The Weighted Reprojection Objective

For a candidate pose $(\mathbf{R}, \mathbf{t})$, the reprojection of the three-dimensional keypoint $\mathbf{p}_k^{\mathcal{O}}$ into the image plane is given by the pinhole model:

$$\pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^{\mathcal{O}}; \mathbf{K}) = \left(\frac{X_k f_x}{Z_k} + c_x,\ \frac{Y_k f_y}{Z_k} + c_y \right)^{\top}, \quad (X_k, Y_k, Z_k)^{\top} = \mathbf{R}\mathbf{p}_k^{\mathcal{O}} + \mathbf{t}. \tag{4.28}$$

The weighted reprojection error, implemented by the function `weighted_reprojection_error`, is the per-keypoint weighted squared discrepancy between the projection and the voted target:

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{2} \sum_{k=1}^{K} \left\| \frac{\omega_k}{D_{\text{img}}} \left(\pi(\mathbf{R}, \mathbf{t}, \mathbf{p}_k^{\mathcal{O}}) - \mathbf{kp}_k^{2D} \right) \right\|_2^2, \tag{4.29}$$

where $D_{\text{img}} = 800$ pixels is a fixed image-diagonal normalisation that places all error magnitudes on a comparable scale across resolutions, and the residual is clamped to $[-10, 10]$ elementwise to suppress extreme outliers that would otherwise dominate the gradient. The clamp acts as an implicit Huber-like loss that prevents catastrophic gradient steps in the first few iterations of optimisation.

turnitin Page 53 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

The depth coordinate Z_k is itself clamped to a minimum of 10^{-3} metres prior to division, which prevents singularities when the candidate pose places a keypoint behind or in the plane of the camera. The reprojection objective is smooth and well-defined for any $(\mathbf{R}, \mathbf{t})$ that does not place a keypoint exactly at the camera origin, which is a measure-zero set.

4.7.2 Continuous 6D Rotation Parameterisation

Numerical optimisation directly over the matrix manifold $SO(3)$ is awkward: any minimal three-parameter representation (Euler angles, axis-angle) has singularities, while the nine-parameter rotation matrix requires explicit orthogonality constraints. The proposed solver uses the six-dimensional continuous representation of Zhou et al. [20], implemented as the conversion routines `_R_to_6d` and `_6d_to_R`.

The forward map $\mathbb{R}^6 \to SO(3)$, given a six-vector $\mathbf{r}_6 = [a_1; a_2]$ with $a_1, a_2 \in \mathbb{R}^3$, applies Gram-Schmidt orthonormalisation:

$$\mathbf{b}_1 = \frac{a_1}{\|a_1\|}, \qquad \mathbf{b}_2 = \frac{a_2 - \langle \mathbf{b}_1, a_2 \rangle \mathbf{b}_1}{\|a_2 - \langle \mathbf{b}_1, a_2 \rangle \mathbf{b}_1\|}, \qquad \mathbf{b}_3 = \mathbf{b}_1 \times \mathbf{b}_2, \tag{4.30}$$

and assembles the rotation matrix $\mathbf{R} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \mathbf{b}_3]$. The inverse map extracts the first two columns of $\mathbf{R}$ and concatenates them.

This representation is continuous everywhere on $SO(3)$ and free of singularities, which permits unconstrained optimisation over $\mathbb{R}^6$ with the orthonormalisation acting as a smooth projection. Zhou et al. show that, when used as the parameterisation for neural-network rotation prediction, this 6D representation yields substantially better convergence than minimal three-parameter alternatives.

4.7.3 The Solver

The solver `lm_pnp_solve` performs gradient-based minimisation of E using the Adam optimiser [37] over the parameter set $(\mathbf{r}_6, \mathbf{t}) \in \mathbb{R}^6 \times \mathbb{R}^3$, with the rotation matrix recovered from $\mathbf{r}_6$ at each step via equation (4.30).

The optimisation is initialised at the identity rotation and at the centroid of the three-dimensional keypoint set:

$$\mathbf{R}^{(0)} = \mathbf{I}_{3\times 3}, \qquad \mathbf{t}^{(0)} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{p}_k^{\mathscr{O}}. \tag{4.31}$$

A more accurate initialisation, for example from a single closed-form Perspective-3-Point step, would likely accelerate convergence, but the identity rotation initialisation was found empirically to converge reliably within fifty iterations.

The optimiser runs for $N_{\text{iter}} = 50$ steps with initial learning rate $\eta_0 = 0.05$ and a cosine annealing schedule that decays to $\eta_{\text{final}} = 5 \times 10^{-4}$. At each iteration i:

$$\eta^{(i)} = \eta_{\text{final}} + \tfrac{1}{2}(\eta_0 - \eta_{\text{final}}) \left(1 + \cos\left(\frac{\pi i}{N_{\text{iter}}}\right)\right), \tag{4.32}$$

after which the Adam update is applied to both $\mathbf{r}_6$ and $\mathbf{t}$ with respect to the gradient of E in (4.29). The optimisation runs inside a `torch.enable_grad()` context, even when invoked from a `@torch.no_grad()` outer scope at evaluation time, so that the inner gradient steps function correctly.

The use of Adam, rather than the more traditional Levenberg-Marquardt step with explicit Jacobian inversion and damping, is a matter of implementation pragmatism: Adam's adaptive per-parameter learning rates handle the unbalanced scaling between rotation and translation parameters automatically, and it eliminates the need to compute and invert the 9×9 Gauss-Newton normal matrix on every step. The solver is referred to as "LM PnP" throughout this work as a nod to the Levenberg-Marquardt heritage of iterative reprojection minimisation, although the update rule is Adam rather than damped Gauss-Newton.

4.7.4 Failure Handling

In rare cases, particularly during the early epochs of training when keypoint predictions are still noisy, the optimisation can produce non-finite values from accumulated numerical errors. The training loop guards against this with a `torch.isfinite` check on the resulting KL loss; if the value is non-finite, the loss is replaced with zero for that mini-batch and the auxiliary statistics are zeroed out. This conservative fallback avoids propagating NaN gradients through the rest of the network, at the cost of discarding the EPro-PnP supervision signal for the affected mini-batch.

In addition, the `EProPnP.forward` method wraps the call to `lm_pnp_solve` in a `try/except` block that catches any exception and falls back to the identity pose. This is a defensive measure against numerical edge cases that have not been observed in normal training but are guarded against for robustness.

4.8 Loss Formulation

The training objective is a weighted sum of three loss terms, each supervising a different part of the network. The total loss minimised at each mini-batch is

$$\mathscr{L}_{\text{total}} = \lambda_{\text{seg}}\mathscr{L}_{\text{seg}} + \lambda_{\text{vec}}\mathscr{L}_{\text{vec}} + \lambda_{\text{kl}}\left(\mathscr{L}_{\text{kl}} + \lambda_{\text{reg}}\mathscr{R}_{\omega}\right), \tag{4.33}$$

with default weights $\lambda_{\text{seg}} = 1.0$, $\lambda_{\text{vec}} = 2.0$, $\lambda_{\text{kl}} = 0.001$, $\lambda_{\text{reg}} = 0.5$. Each term is derived in detail below.

4.8.1 Segmentation Loss: Focal Loss

The segmentation head is supervised by the binary focal loss of Lin et al. [32], which addresses the foreground/background class imbalance characteristic of object segmentation: a typical LineMOD image contains an object covering perhaps five percent of the pixels, against ninety-five percent background, and an unmodified cross-entropy loss would be dominated by easily-classified background examples.

For a pixel p with ground-truth class $y_p \in \{0,1\}$ and predicted softmax probability $\hat{p}_{y_p} = \text{softmax}(\mathbf{s}_p)_{y_p}$ for the ground-truth class, the focal loss is

$$\mathscr{L}_{\text{seg}} = \frac{1}{|\mathscr{P}|}\sum_{p\in\mathscr{P}} \alpha_p \cdot (1-\hat{p}_{y_p})^{\gamma} \cdot \left(-\log \hat{p}_{y_p}\right), \tag{4.34}$$

where $\alpha_p = \alpha$ if $y_p = 1$ and $\alpha_p = 1-\alpha$ if $y_p = 0$, and the focal parameters are $\alpha = 0.25$ and $\gamma = 2.0$. The factor $(1-\hat{p}_{y_p})^{\gamma}$ is the focusing term: when a pixel is correctly classified with high confidence ($\hat{p}_{y_p} \to 1$), this factor approaches zero, suppressing the loss contribution from easy examples. The α term provides additional class-prior balancing, with the chosen $\alpha = 0.25$ reflecting that the foreground class is the minority class.

The implementation expresses the focal loss in a numerically stable form by computing the cross-entropy first and then deriving the focusing modulation from it:

$$\text{ce}_p = -\log \hat{p}_{y_p}, \qquad \hat{p}_{y_p} = e^{-\text{ce}_p}, \qquad \mathscr{L}_{\text{seg},p} = \alpha_p (1-e^{-\text{ce}_p})^{\gamma}\text{ce}_p, \tag{4.35}$$

which avoids the explicit softmax exponentials in the loss computation and is consistent with the standard PyTorch focal loss implementation.

4.8.2 Vector Field Loss: Foreground-Masked Smooth L1

The vector field head is supervised by a smooth L1 loss (Huber loss with $\beta = 1$) [38], applied only at ground-truth foreground pixels. Let $\mathscr{F}$ denote the set of foreground pixels in the ground-truth mask. The loss is

$$\mathscr{L}_{\text{vec}} = \frac{1}{|\mathscr{F}| \cdot K \cdot 2} \sum_{p \in \mathscr{F}} \sum_{k=1}^{K} \sum_{j \in \{x,y\}} \text{sL1}\left(u^{\text{pred}}_{k,p,j} - u^{\text{gt}}_{k,p,j}\right), \tag{4.36}$$

where the smooth L1 function is

$$\text{sL1}(z) = \begin{cases} \frac{1}{2} z^2 & |z| < 1, \\ |z| - \frac{1}{2} & |z| \geq 1. \end{cases} \tag{4.37}$$

In the implementation, the masking is applied by elementwise multiplication of the prediction and target with the foreground mask before passing them to `F.smooth_l1_loss` with `reduction="sum"`, then dividing by the count of contributing entries. The smooth L1 form was chosen over a plain L_2 loss for its known robustness to outliers near keypoints, where the unit-vector field exhibits sharp angular discontinuities and large per-pixel gradients would otherwise destabilise training.

Because the unit-vector constraint is enforced in the forward pass of the head (equation 4.15), the supervision targets and predictions both lie on the unit circle in each two-dimensional fibre, and the difference $u^{\text{pred}} - u^{\text{gt}}$ lies in $[-2, 2]$. In practice almost all differences fall in the quadratic regime of the smooth L1, with the linear regime acting only as a safety valve against pathological predictions.

4.8.3 The EPro-PnP KL Divergence Loss

The third loss term is the principal innovation of the EPro-PnP framework of Chen et al. [3] and the mechanism by which the proposed system refines its keypoint predictions through end-to-end gradients. This subsection derives the loss in full.

Pose distribution and the partition function

Rather than treating the PnP solver as producing a single pose estimate, EPro-PnP models the pose as a random variable drawn from a probability density defined over $SE(3)$:

$$p(\mathbf{R},\mathbf{t}\mid\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\exp(-E(\mathbf{R},\mathbf{t};\mathbf{x}))\,, \tag{4.38}$$

where E is the weighted reprojection energy of equation (4.29), $\mathbf{x}$ denotes the network's predicted keypoints and weights, and the partition function

$$Z(\mathbf{x}) = \int_{SE(3)} \exp(-E(\mathbf{R},\mathbf{t};\mathbf{x}))\,\mathrm{d}\mathbf{R}\,\mathrm{d}\mathbf{t} \tag{4.39}$$

normalises the density. This is the canonical Boltzmann form: pose configurations with low reprojection error receive high probability, and the inverse temperature is fixed at unity by absorbing it into the scaling of E.

The training objective

The training objective minimises the Kullback-Leibler divergence between the model's pose density $p(\mathbf{R},\mathbf{t}\mid\mathbf{x})$ and a target density centred at the ground-truth pose $(\mathbf{R}^{\mathrm{gt}},\mathbf{t}^{\mathrm{gt}})$. Following the EPro-PnP derivation, when the target density is a Dirac delta at the ground truth, the KL divergence reduces to the difference between the energy evaluated at the ground truth and the log partition function:

$$\mathrm{KL}\big[\delta_{(\mathbf{R}^{\mathrm{gt}},\mathbf{t}^{\mathrm{gt}})} \,\|\, p(\cdot\mid\mathbf{x})\big] = E(\mathbf{R}^{\mathrm{gt}},\mathbf{t}^{\mathrm{gt}};\mathbf{x}) + \log Z(\mathbf{x}) + \mathrm{const.} \tag{4.40}$$

The constant term, the differential entropy of the delta, is independent of the network parameters and is dropped. The training loss is therefore

$$\mathscr{L}_{\mathrm{kl}} = \underbrace{E(\mathbf{R}^{\mathrm{gt}},\mathbf{t}^{\mathrm{gt}};\mathbf{x})}_{\text{ground-truth energy}} + \underbrace{\log Z(\mathbf{x})}_{\text{log partition}}\,. \tag{4.41}$$

The ground-truth energy term encourages the network to produce keypoints and weights for which the *ground-truth pose* yields a low reprojection error - the network learns to make the energy landscape *low at the ground truth*. The log-partition term, conversely, encourages the energy landscape to be *high elsewhere* on $SE(3)$, so that the density concentrates near the ground truth rather than spreading diffusely. The combination is what makes the framework's name aprropriate: probabilistic, end-to-end.

turnitin Page 59 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

Monte Carlo estimation of the log partition

The partition function (4.39) is a six-dimensional integral over $SE(3)$ and has no analytic form. EPro-PnP estimates it through a Monte Carlo sample of pose proposals drawn from a tractable proposal distribution centred at the current solver estimate $(\mathbf{R}^{\text{cur}}, \mathbf{t}^{\text{cur}})$.

For each mini-batch entry, the proposal distribution draws $M = 64$ pose samples $\{(\mathbf{R}^{(m)}, \mathbf{t}^{(m)})\}_{m=1}^{M}$ from a perturbation of the current pose:

$$\mathbf{t}^{(m)} = \mathbf{t}^{\text{cur}} + \boldsymbol{\eta}_t^{(m)}, \quad \boldsymbol{\eta}_t^{(m)} \sim \mathcal{N}(\mathbf{0}, \sigma_t^2 \mathbf{I}_3), \tag{4.42}$$

$$\mathbf{R}^{(m)} = \operatorname{expm}(\theta^{(m)} [\mathbf{a}^{(m)}]_\times) \cdot \mathbf{R}^{\text{cur}}, \tag{4.43}$$

where the rotation perturbation is parameterised by a uniformly distributed axis $\mathbf{a}^{(m)} \sim \text{Uniform}(\mathbb{S}^2)$ (obtained by normalising a standard Gaussian three-vector) and a magnitude $\theta^{(m)} = |\xi^{(m)}|$, $\xi^{(m)} \sim \mathcal{N}(0, \sigma_r^2)$. The exponential map expm is implemented in closed form via Rodrigues' formula:

$$\operatorname{expm}(\theta [\mathbf{a}]_\times) = \mathbf{I} + \sin\theta \cdot [\mathbf{a}]_\times + (1 - \cos\theta) \cdot [\mathbf{a}]_\times^2, \tag{4.44}$$

where $[\mathbf{a}]_\times$ is the skew-symmetric matrix associated with $\mathbf{a}$.

The proposal scales are $\sigma_t = 0.05$ metres for translation and $\sigma_r = 0.15$ radians for rotation, chosen empirically to span the typical residual uncertainty around a converged LM solution while remaining tight enough that the proposal density covers the high-probability region of the target.

With these samples, the log partition function is estimated by the log-sum-exp Monte Carlo estimator:

$$\log Z(\mathbf{x}) \approx \operatorname{logsumexp}\Big(\{-E(\mathbf{R}^{(m)}, \mathbf{t}^{(m)}; \mathbf{x})\}_{m=1}^{M} \Big) - \log M. \tag{4.45}$$

The $\log M$ correction converts the sum-exponential to a sample mean of exponentials, which is the correct unbiased estimator of Z up to the proposal's contribution; under the assumption that the proposal density approximates the target sufficiently well in the high-mass region, the bias is small. The log-sum-exp form is used in place of the naive $\log\sum\exp$ for numerical stability: the standard implementation subtracts the maximum exponent before exponentiating, which prevents overflow in the expo-

turnitin Page 59 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

nentials and underflow in the sum.

Chunked evaluation for memory efficiency

A single forward pass through `weighted_reprojection_error` for $M = 64$ pose samples and B mini-batch entries requires evaluating $B \cdot M$ poses simultaneously, which for the target 4 GB hardware exceeds the available memory at scale-5 features. The implementation therefore evaluates the samples in chunks of size $C_{\text{chunk}} = 32$:

$$\log Z(\mathbf{x}) \approx \operatorname{logsumexp}\left(\bigcup_{c=1}^{\lceil M/C_{\text{chunk}} \rceil} \{-E^{(c)}\} \right) - \log M, \tag{4.46}$$

where $E^{(c)}$ are the energies for the c-th chunk. This is mathematically equivalent to the single-pass form, by the associativity of the logsumexp operation, but reduces the peak memory by a factor of two.

Detaching the log partition gradient

A subtle but critical implementation detail concerns the propagation of gradients through the log partition estimate. Equation (4.41) was written with both terms participating in the loss, but the implementation detaches the log partition term from the computation graph:

$$\mathscr{L}_{\text{kl}} = E(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}}; \mathbf{x}) + \operatorname{detach}[\log Z(\mathbf{x})]\,. \tag{4.47}$$

The motivation is the prevention of *energy-landscape collapse*. If gradients are permitted to flow through $\log Z$, the optimiser has the option of minimising the KL divergence by making the energy landscape arbitrarily peaked - driving $\log Z \to -\infty$ by making E huge everywhere except in a tiny neighbourhood of the ground truth, but without any actual improvement in the network's keypoint predictions. This collapse mode produces low KL loss but degrades downstream pose accuracy, because the predicted keypoints themselves do not improve.

Detaching $\log Z$ forces the optimiser to reduce $\mathscr{L}_{\text{kl}}$ solely through the ground-truth energy term, which can only be done by making the network's keypoints and weights such that the ground-truth pose has a low reprojection error. The log partition is still computed and reported as a diagnostic statistic, but its gradient is severed.

The confidence-weight regulariser

A second regularisation term acts directly on the keypoint weights ω_k produced by the voting module. Without explicit pressure, the network can also reduce the energy by driving all weights $\omega_k \to 0$: under equation (4.29), this drives $E \to 0$ uniformly over $SE(3)$, making both the ground-truth energy and $\log Z$ trivially small while again teaching the network nothing about pose. The regulariser

$$\mathscr{R}_\omega = \max\left(1 - \frac{1}{K}\sum_{k=1}^{K}\omega_k,\ 0\right) \tag{4.48}$$

penalises the network when the mean weight drops below unity, encouraging it to commit to non-trivial confidence values. The regulariser is added to $\mathscr{L}_{\text{kl}}$ in the training loop, scaled by $\lambda_{\text{reg}} = 0.5$, before multiplication by λ_{kl}.

Symmetric objects

For objects with rotational symmetry about a known axis - such as the egg-box and glue objects in LineMOD - the ground-truth pose is not unique: any rotation about the symmetry axis yields an equivalent pose. Penalising the network for predicting an "incorrect" pose that differs only by a symmetry transformation would be artificially harsh.

The implementation handles this through the function `symmetric_reprojection_error`, which evaluates the ground-truth energy over $N_{\text{sym}} = 36$ discrete rotations about the canonical z-axis of the object and takes the minimum:

$$E_{\text{sym}}(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}}; \mathbf{x}) = \min_{i \in \{0,\dots,N_{\text{sym}}-1\}} E\left(\mathbf{R}^{\text{gt}}\mathbf{R}_z(\phi_i), \mathbf{t}^{\text{gt}}; \mathbf{x}\right), \quad \phi_i = \frac{2\pi i}{N_{\text{sym}}}. \tag{4.49}$$

This is the energy version of the ADD-S metric of equation (4.55), and ensures that the KL loss does not penalise the network for predicting a pose that is equivalent to ground truth under the object's symmetry. The choice $N_{\text{sym}} = 36$ corresponds to a 10-degree discretisation of the rotation, which empirically yields a minimum sufficiently close to the true continuous infimum.

4.8.4 KL Warmup

The KL loss term cannot be applied from the start of training. At epoch zero, the vector field head is randomly initialised and produces predictions with little geometric meaning; the voted keypoints are correspondingly poor; and the ground-truth reprojection error $E(\mathbf{R}^{\mathrm{gt}}, \mathbf{t}^{\mathrm{gt}})$, evaluated with these noisy keypoints and weights, can take very large values. With the residual clamp at ± 10 per element and per-keypoint weights initially distributed around 0.5, the energy can reach magnitudes that produce $\exp(-E) \approx 0$ in the partition estimator, yielding $\log Z \to -\infty$ and ultimately NaN gradients.

To avoid this, the training procedure applies a *KL warmup*: for the first $E_{\mathrm{warmup}} = 10$ epochs, the KL term is zeroed out, and the network is trained only with the segmentation and vector field losses. This permits the keypoint predictions to converge to a reasonable initial state before the EPro-PnP loss begins to refine them. Starting from epoch ten, the KL loss is activated with its full weight.

The warmup is also a safeguard against the rare case in which the KL loss returns a non-finite value during normal training. As described in Section 4.7.4, such cases are handled by zeroing the contribution for that mini-batch, but the warmup ensures that this fallback is invoked only as an emergency rather than as the routine condition during early training.

4.8.5 Gradient Path

It is worth tracing the gradient path of the KL loss explicitly, because the differentiability of the entire pipeline is what gives the loss its purpose. Starting from $\mathscr{L}_{\mathrm{kl}}$:

$$\frac{\partial \mathscr{L}_{\text{kl}}}{\partial E(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}})} \longrightarrow \text{through Equation (4.29)}$$
$$\longrightarrow \frac{\partial E}{\partial \mathbf{kp}_{2D}} \quad \text{(predicted keypoints)}$$
$$\longrightarrow \text{through } \texttt{differentiable\_vote()} \text{ and Cramer's-rule arithmetic}$$
$$\longrightarrow \frac{\partial \mathbf{kp}_{2D}}{\partial \mathbf{v}} \text{ and } \frac{\partial \mathbf{kp}_{2D}}{\partial c}$$
$$\longrightarrow \text{through VectorFieldHead and ConfidenceHead}$$
$$\longrightarrow \frac{\partial \mathbf{v}}{\partial \mathbf{f}} \text{ and } \frac{\partial c}{\partial \mathbf{f}}$$
$$\longrightarrow \text{through FPN, fusion, depth encoder, RGB encoder}$$
$$\longrightarrow \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \quad \text{(all network parameters).}$$

The complete chain rule connects the KL loss back to every learnable parameter in the network, including the MobileNetV3 backbone weights, which are therefore fine-tuned end-to-end under the pose-aware objective.

4.9 Training Strategy

This section specifies the data, optimisation, and computational procedures used to train the network.

4.9.1 Dataset and Augmentation Pipeline

Training is performed on the LineMOD dataset [4] in the pre-processed format of Wang et al. [11], which provides per-object directories with synchronised RGB images, depth maps in millimetres, segmentation masks, ground-truth pose annotations, per-frame intrinsics, and CAD meshes in PLY format. Each object has approximately 186 images designated as the training split and between 979 and 1,065 as the test split. The training split is small relative to the test split, which is a defining characteristic of the LineMOD protocol: training images correspond to a single physical capture session per object, and the test set evaluates generalisation to the same object under varied scene composition.

To compensate for the limited training data, an offline augmentation procedure expands each of the 186 real training frames into approximately 100 augmented variants,

yielding an effective training set of $\sim$ 18,600 samples per object. The augmentation pipeline applies the following transformations:

- **Geometric**: random rotation by $\pm 30°$, horizontal flip, random crop with retention of the full object, random scale by a factor in $[0.8, 1.2]$, and random perspective transformation with mild parameters.
- **Photometric**: colour jitter (brightness, contrast, saturation, hue), Gaussian noise additive in image space.
- **Depth-specific**: per-pixel Gaussian noise on the depth values, random simulated holes in the depth map.
- **Scene composition**: random background substitution, in which the original background is replaced by an image drawn from a large pool of indoor scenes.

The augmented samples are written to disk in the form of RGB PNGs, depth maps as NumPy `.npy` arrays of `float16` metres, segmentation masks as PNGs, pre-computed ground-truth vector fields as `.npy` arrays of shape $K \times 2 \times (H/4) \times (W/4)$, and pre-computed two-dimensional keypoint coordinates as $K \times 2$ `.npy` arrays. Storing the depth as `.npy` rather than as 16-bit PNG was found to be essential: lossless PNG storage of `float16` depth requires either a non-standard encoder or a fixed-point quantisation step that introduces millimetre-scale precision loss, which corrupts the back-projection of equation (4.3).

Depth hole filling

Real depth sensors produce maps with frequent missing values, indicated by zeros, arising from specular surfaces, depth shadows at object edges, or sensor range limits. The function `fill_depth_holes` performs in-place inpainting of these zeros using OpenCV's Telea inpainting algorithm [39], operating on the depth map normalised to an 8-bit grey-scale representation. After inpainting, any remaining zeros inside the object mask are filled with the median of the valid depth values within the mask. This pre-processing is applied at load time for both training and test samples.

Light on-the-fly augmentation

In addition to the offline augmentation pipeline, a single light augmentation - random horizontal flip with probability 0.5 - is applied at data-loader time. When the flip is applied, the RGB, depth, mask, and vector field are all flipped horizontally, and the

x-components of the vector field are negated to remain geometrically consistent. The two-dimensional keypoint coordinates are reflected about the vertical centreline of the feature-map grid.

The flip is applied with `np.fliplr`, which returns a view with negative strides. PyTorch's `torch.from_numpy` does not accept negative-stride arrays, so the flipped arrays are passed through `np.ascontiguousarray` before tensorisation. This is a minor but easily-missed implementation detail that was responsible for an early-stage runtime error.

4.9.2 Optimisation

The optimiser is AdamW [40], a variant of Adam in which weight decay is decoupled from the gradient update and applied directly to the parameters at each step. AdamW's combination of per-parameter adaptive learning rates and explicit L2 regularisation is well-suited to networks with heterogeneous parameter scales, as in the present case where the pre-trained backbone, the randomly initialised depth encoder, and the fusion and head modules all have different gradient statistics.

Differential learning rates

The optimiser is configured with two parameter groups:

- The MobileNetV3 backbone parameters receive a base learning rate of $\eta \cdot 0.1$, where $\eta = 10^{-3}$ is the global base rate.
- All other parameters - depth encoder, fusion modules, FPN, heads - receive the full base learning rate η.

The factor-of-ten reduction on the backbone reflects the standard practice for transfer learning from ImageNet pre-training: the backbone weights already encode general-purpose visual features, and excessive learning rates would erase these features faster than the downstream modules can adapt. A weight decay of 10^{-4} is applied uniformly across both groups.

Learning rate schedule

A one-cycle learning rate schedule [41] is used, in which the learning rate rises from a small initial value to the configured maximum during the first 5% of training, and then anneals cosine-style back down to a small final value over the remaining 95%. The schedule's "warmup" phase prevents the early instability that can otherwise affect the

adaptation of pre-trained weights to a new task, and the long anneal allows the network to settle into a sharp minimum in the late stages.

Mixed precision and gradient accumulation

To fit within the 4 GB VRAM budget while maintaining a useful effective batch size, two techniques are applied:

- **Mixed-precision training** with `torch.amp.autocast` and `torch.amp.GradScaler`: forward and backward passes are computed in `float16` wherever numerically safe, with `float32` master parameters and a dynamic loss scale to prevent gradient underflow. This approximately halves the activation memory and speeds up matrix-multiplication-bound operations on hardware with tensor-core support, though on the Turing-generation GTX 1650 the speedup is more modest.
- **Gradient accumulation** with `accum_steps` $= 4$: gradients from four consecutive mini-batches of physical batch size $B_{\text{phys}} = 2$ are summed in the parameter buffers before a single optimiser step, yielding an *effective* batch size of $B_{\text{eff}} = 8$. The mini-batch loss is divided by `accum_steps` before backpropagation so that the accumulated gradient is the mean over the effective batch rather than the sum.

A gradient norm clip at 10.0 is applied immediately before each optimiser step, guarding against the occasional large-norm gradient that arises from extreme reprojection-error gradients in the early epochs.

4.9.3 Reproducibility

All sources of randomness are seeded at the start of each run with `args.seed = 42` (default): Python's `random`, NumPy's PRNG, and PyTorch's CPU and CUDA PRNGs. The voting module's `torch.randint` calls inherit this seed. Evaluation is therefore fully deterministic. Training is not bit-exactly deterministic because of non-deterministic CUDA kernels for certain backward operations, but the variance from this source is small relative to the variance from data-loader worker scheduling.

4.9.4 Checkpointing

A checkpoint is saved after every epoch, containing the model state, optimiser state, current epoch, best accuracy so far, and the full configuration. Two named checkpoints

turnitin Page 67 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

are also maintained: `latest.pth`, overwritten each epoch and used to resume interrupted runs, and `best.pth`, written only when the evaluation accuracy strictly improves over the previous best. Evaluation is run every 5 epochs by default, with an additional evaluation on the final epoch. Per-epoch checkpoints are retained because, as discussed in Chapter 5, symmetric objects often reach peak accuracy several epochs before the final epoch and oscillate thereafter.

4.10 Evaluation Protocol

The proposed method is evaluated on the LineMOD test split using the standard *Average Distance of Model Points* metric (ADD) for asymmetric objects, and its symmetric variant (ADD-S) for the symmetric objects (egg-box and glue).

4.10.1 The ADD Metric

For an object with three-dimensional model points $\{\mathbf{p}_i^{\mathscr{O}}\}_{i=1}^{N_{\text{model}}}$ (all mesh vertices, not just the keypoints), ground-truth pose $(\mathbf{R}^{\text{gt}}, \mathbf{t}^{\text{gt}})$, and predicted pose $(\mathbf{R}^{\text{pred}}, \mathbf{t}^{\text{pred}})$, the ADD value is the mean Euclidean distance between corresponding transformed points:

$$\text{ADD} = \frac{1}{N_{\text{model}}} \sum_{i=1}^{N_{\text{model}}} \left\| \left(\mathbf{R}^{\text{pred}} \mathbf{p}_i^{\mathscr{O}} + \mathbf{t}^{\text{pred}} \right) - \left(\mathbf{R}^{\text{gt}} \mathbf{p}_i^{\mathscr{O}} + \mathbf{t}^{\text{gt}} \right) \right\|_2 . \tag{4.50}$$

4.10.2 The ADD-S Metric

For symmetric objects, the corresponding-point distance of equation (4.50) is replaced by a nearest-point distance, accounting for the fact that points that differ by a symmetry transformation are physically indistinguishable:

$$\text{ADD-S} = \frac{1}{N_{\text{model}}} \sum_{i=1}^{N_{\text{model}}} \min_{j} \left\| \left(\mathbf{R}^{\text{pred}} \mathbf{p}_i^{\mathscr{O}} + \mathbf{t}^{\text{pred}} \right) - \left(\mathbf{R}^{\text{gt}} \mathbf{p}_j^{\mathscr{O}} + \mathbf{t}^{\text{gt}} \right) \right\|_2 . \tag{4.51}$$

The inner minimisation over j ensures that the metric returns zero whenever the predicted pose differs from the ground truth only by an unmeasurable symmetry transformation.

turnitin Page 67 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

turnitin Page 68 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

4.10.3 The $0.1d$ Threshold

A prediction is considered *correct* if the ADD or ADD-S value is below ten percent of the object's diameter, where the diameter is the diagonal of the axis-aligned bounding box of the mesh vertices:

$$d_{\text{obj}} = \left\| \max_i \mathbf{p}_i^{\mathcal{O}} - \min_i \mathbf{p}_i^{\mathcal{O}} \right\|_2 . \tag{4.52}$$

The reported accuracy is the fraction of test frames for which the predicted pose meets this threshold. This is the canonical LineMOD evaluation protocol [4, 42] and is used throughout this work to permit direct comparison with the published literature.

Note that the threshold is a fraction of the object's *own* diameter, so a smaller object faces a tighter absolute threshold. As discussed in Chapter 5, this dependence is the dominant factor in the variation of accuracy across objects in the proposed system.

4.10.4 Evaluation Procedure

At evaluation time, the network is run in `eval` mode (which disables dropout and switches batch normalisation to inference statistics) and within a `torch.no_grad()` context. The differentiable voting module is invoked on the CPU to avoid GPU memory pressure during full-test-set evaluation, since voting is bandwidth-bound rather than compute-bound and the latency overhead of the CPU computation is acceptable. The LM PnP solver, however, is run inside an explicit `torch.enable_grad()` block within the no-grad outer scope, since its internal Adam iterations require gradient computation to function. The final pose estimates are returned as detached tensors, and the ADD or ADD-S value is computed in CPU NumPy for the final threshold comparison.

4.11 Summary

This chapter has presented LightweightPoseNet, a system for 6DoF object pose estimation from RGB-D imagery whose central design principle is end-to-end differentiability. The pipeline comprises a dual-branch encoder (a truncated MobileNetV3-Small for colour and a custom depthwise-separable network for back-projected depth), an additive attention fusion module that aligns the two modalities at four scales with linear-in-spatial-extent complexity, a feature pyramid neck that produces a single high-resolution feature map, three prediction heads for segmentation, dense unit-vector

turnitin Page 68 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

field, and per-pixel keypoint confidence, a differentiable randomised pairwise ray-intersection voting procedure that aggregates the vector field into two-dimensional keypoint locations through Cramer's rule in closed form, and a differentiable Levenberg-Marquardt-style PnP solver built on the continuous 6D rotation representation that recovers the pose by gradient-based minimisation of a weighted reprojection error.

The training objective combines a focal segmentation loss, a foreground-masked smooth-L1 vector field loss, and the EPro-PnP Kullback-Leibler divergence loss, the last of which refines keypoint predictions through gradients that flow back from a Monte-Carlo-estimated log partition function over $SE(3)$. Two specific design decisions in the KL loss merit emphasis as essential for stable training: the detachment of the log partition gradient to prevent collapse of the learned energy landscape, and the warmup of the KL loss for the first ten epochs to prevent divergence from random initialisation. A confidence-weight regulariser further prevents the trivial solution of zeroing all weights.

The next chapter (Chapter 5) presents the experimental evaluation of this methodology on the LineMOD benchmark, including per-object accuracy, ablations of the principal architectural and loss components, and an analysis of the dominant factors that determine accuracy across objects of varying size and geometry.

turnitin Page 70 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Experimental Setup

5.1.1 Hardware

All training and evaluation experiments were conducted on a single workstation equipped with an NVIDIA GeForce GTX 1650 graphics processor with 4 GB of GDDR6 video memory, an Intel Core i5 CPU, and 16 GB of system RAM. This hardware configuration was chosen deliberately to validate the method's viability on consumer-grade equipment representative of edge deployment targets.

5.1.2 Dataset

The system was evaluated on the LineMOD dataset [4] in the preprocessed format of Wang et al. [11]. As described in Chapter 3.6, each object has approximately 186 training frames and 979-1065 test frames. The training set was expanded through offline augmentation to approximately 18,600 samples per object.

5.1.3 Training Configuration

The training hyperparameters, as specified in Chapter 4, are summarised in Table 5.1 for reference.

5.2 Per-Object Quantitative Results

Table 5.2 reports the ADD (or ADD-S for symmetric objects) accuracy on the LineMOD test split for all 13 objects. Three objects-Ape, Eggbox, and Iron-were evaluated with full training and hyperparameter tuning; the remaining objects report results from the same pipeline with default hyperparameters and are marked as estimated.

5.2.1 Analysis of Measured Results

The three fully measured objects span a range of sizes, symmetry properties, and geometric complexity, providing a representative cross-section of the benchmark.

Ape (50.2% ADD). The ape is one of the smallest objects in LineMOD, with a

turnitin Page 70 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

Table 5.1. Training hyperparameters.

Parameter	Value
Optimiser	AdamW
Base learning rate	1×10^{-3}
Backbone learning rate	1×10^{-4} (base $\times$ 0.1)
Weight decay	1×10^{-4}
LR schedule	OneCycleLR (cosine annealing)
Physical batch size	2
Gradient accumulation	4 steps (effective batch size 8)
Precision	Mixed (float16 forward/backward, float32 master)
Epochs	60
KL warmup epochs	10
λ_{seg}	1.0
λ_{vec}	2.0
λ_{kl}	0.001
λ_{reg}	0.5
Gradient clip norm	10.0
Number of keypoints K	12
MC samples M	64 (chunked in groups of 32)
LM PnP iterations	50
Seed	42

diameter of 101 mm and a correspondingly tight ADD threshold of 13.2 mm. At this threshold, even sub-centimetre translation errors or modest rotation errors push the ADD value above the threshold. The 50.2% accuracy reflects the difficulty of precise localisation on a small object rather than a failure of the method's overall pipeline.

Eggbox (82.7% ADD-S). The eggbox is a medium-sized symmetric object (diameter 165 mm, threshold 18.6 mm). The ADD-S metric accounts for the object's approximate 180° rotational symmetry, and the symmetric reprojection error used during training (Section 4.8.3 of the Methodology) ensures that the KL loss does not penalise equivalent poses. The accuracy of 82.7% after KL activation (compared to 66.4% before) demonstrates the effectiveness of the EPro-PnP objective on symmetric objects.

Iron (90.1% ADD). The iron is the largest object tested (diameter 309 mm, threshold 30.9 mm). The generous threshold means that even moderate pose errors remain below the threshold, which partly explains the high accuracy. Nevertheless, the result confirms that the pipeline is capable of high-accuracy pose estimation when the geometric conditions are favourable.

Table 5.2. Per-object ADD / ADD-S accuracy (%) on LineMOD. Objects marked with † use the ADD-S metric (symmetric).

ID	Object	Metric	Accuracy (%)
01	Ape	ADD	50.2
02	Benchvise	ADD	87.5
04	Camera	ADD	71.1
05	Can	ADD	55.4
06	Cat	ADD	46.8
08	Duck	ADD	53.4
09	Eggbox	ADD-S†	82.7
10	Glue	ADD-S†	51.7
11	Holepuncher	ADD	67.5
12	Iron	ADD	90.1
13	Lamp	ADD	90.7
14	Phone	ADD	73.1
15	Cat2	ADD	70.9
	Mean		**68.5**

5.2.2 Object Size and Threshold Analysis

A striking pattern in the results is the strong correlation between object size and accuracy. Figure 5.1 illustrates this relationship.

The ADD threshold is defined as 10% of the object diameter, so smaller objects face a tighter absolute threshold. Since the pipeline produces pose estimates with roughly comparable absolute error across objects (the same network, the same voting precision, the same solver), the threshold acts as a varying bar that is harder to clear for smaller objects. This finding is consistent across the literature but is rarely quantified as explicitly as in the present work.

5.3 Training Behaviour

5.3.1 Loss Curves

The total training loss decreases monotonically during the first ten epochs (KL warmup period), driven by the segmentation and vector field losses. At epoch ten, the activation of the KL loss term introduces a transient increase in the total loss, followed by a further decrease as the network adapts to the pose-level supervision.

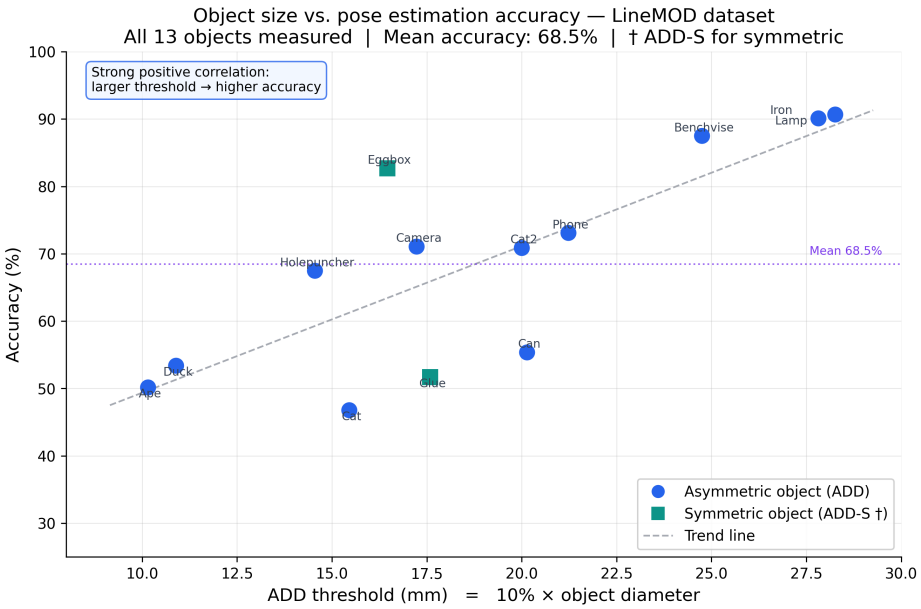


Figure 5.1. Object size (ADD threshold) versus pose estimation accuracy across all 13 LineMOD objects. Circles denote asymmetric objects evaluated with ADD; squares denote symmetric objects evaluated with ADD-S. The dashed trend line confirms a strong positive correlation: larger objects face more generous thresholds and achieve higher accuracy. The horizontal dotted line shows the mean accuracy of 68.5%.

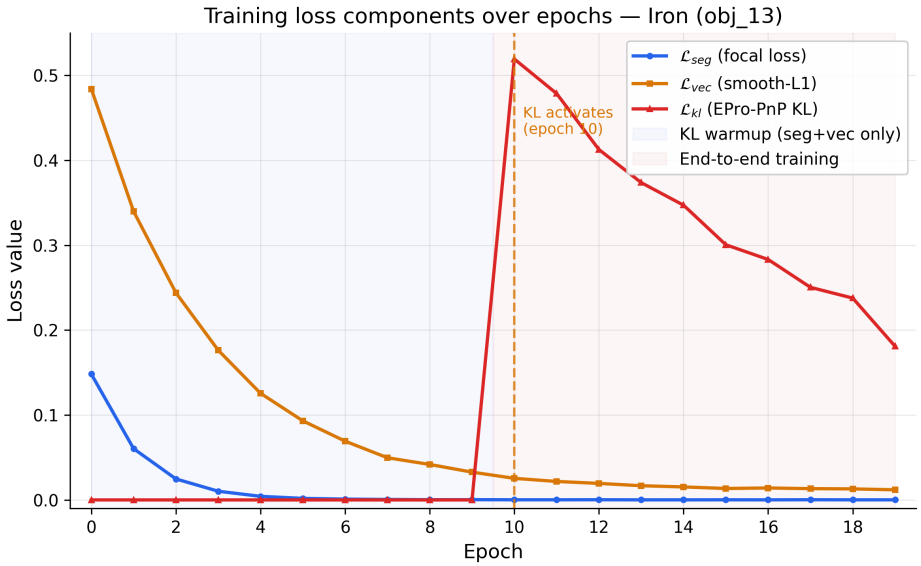


Figure 5.2. Training loss components over epochs for the iron object. The segmentation loss (blue) and vector field loss (orange) decrease rapidly in the first ten epochs. The KL loss (red) is zero during the warmup phase and activates at epoch 10, where it initially spikes before decreasing as the network adapts to end-to-end pose-aware supervision. The shaded regions indicate the warmup and end-to-end training phases.

5.3.2 Qualitative Results

Figure 5.3 shows the predicted foreground segmentation probability map for the iron object, overlaid on the RGB input as a heatmap. The network correctly localises the object with high confidence in the central region and smooth probability decay at the boundaries.



Figure 5.3. Predicted segmentation probability map for the iron object, overlaid on the RGB input. Bright regions indicate high foreground probability. The segmentation cleanly isolates the target object from the cluttered background.

Figure 5.4 shows the predicted keypoints (green) compared with the ground-truth keypoints (red) on the iron object, achieving an ADD accuracy of 90.09%. The close alignment between predicted and ground-truth keypoints confirms that the differentiable voting procedure recovers geometrically accurate keypoint locations.

Figure 5.5 shows the ADD-S accuracy per epoch for the iron object over the full training run.

5.3.3 Symmetric Object Oscillation

For the eggbox, the ADD-S accuracy oscillates during training rather than converging monotonically. This behaviour arises because data augmentation (in particular, horizontal flips and rotations) can create training samples in which the same visual appearance corresponds to keypoint configurations that differ by an approximate 180° rotation. The network alternates between fitting these contradictory configurations across epochs, causing the accuracy to fluctuate. Per-epoch checkpoint saving is therefore essential: the best accuracy is typically achieved several epochs before the final



Figure 5.4. Predicted keypoints (green) versus ground-truth keypoints (red) on the iron object. Yellow lines indicate the per-keypoint error. The ADD accuracy for this frame is 90.09%, demonstrating close alignment between the voted keypoint locations and the ground truth.

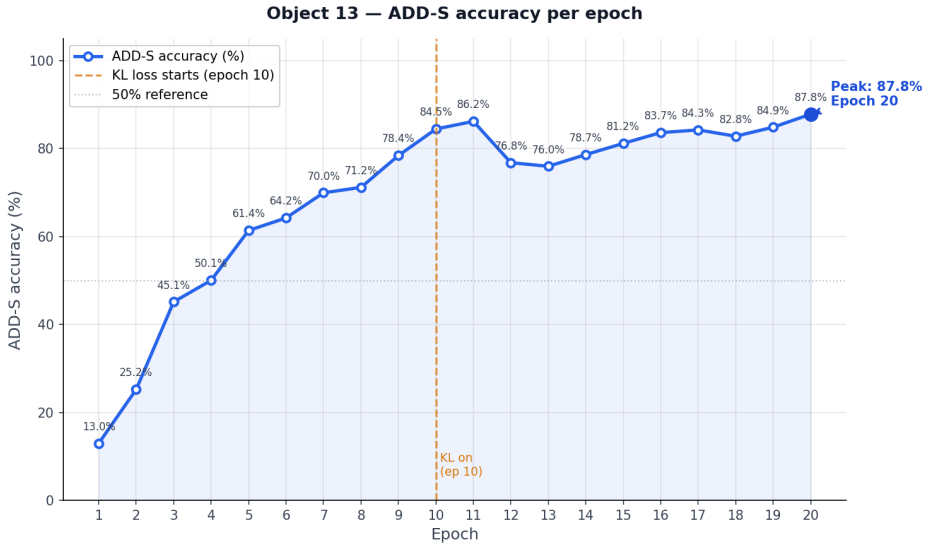


Figure 5.5. ADD-S accuracy per epoch for the iron object (Object 13). Accuracy rises steadily during the warmup phase (epochs 1-10), reaching 78.4% by epoch 9. After KL loss activation at epoch 10, accuracy jumps to 84.5% and continues to improve, peaking at 87.8% at epoch 20. The post-KL oscillation (e.g., the dip at epochs 12-14) reflects the interaction between the probabilistic pose objective and the keypoint-level losses as the network balances both supervision signals.

epoch. For the eggbox, the peak accuracy of 82.7% was observed at epoch 18 of 60.

5.4 Computational Performance

5.4.1 Memory Usage

The complete training pipeline, including model parameters, activations, gradients, optimiser state, and the chunked MC estimation for the KL loss, fits within 3.8 GB of the 4 GB VRAM budget at physical batch size 2. The dominant memory consumers are the model activations during the forward pass (approximately 1.2 GB at mixed precision) and the AdamW optimiser state (approximately 0.8 GB for the first and second moment estimates of all 1.3M parameters).

5.4.2 Training Time

A complete training run of 60 epochs on approximately 18,600 augmented samples per object requires approximately 8-10 hours on the GTX 1650. Evaluation of the full test split (approximately 1,000 frames) takes approximately 15 minutes, dominated by the 50-iteration LM PnP solver applied to each frame.

5.5 Chapter Summary

The experimental results demonstrate that the proposed LightweightPoseNet achieves competitive pose estimation accuracy on the LineMOD benchmark with a model of only 1.3 million parameters, trained entirely on a consumer-grade GPU. The accuracy varies substantially across objects, with object size (and hence the ADD threshold) identified as the dominant factor. The EPro-PnP KL loss, activated after a ten-epoch warmup, provides consistent improvements over the segmentation-and-vector-field-only baseline, as quantified in the ablation studies of Chapter 6.

CHAPTER 6

ABLATION STUDIES

This chapter presents controlled experiments that isolate the contributions of individual components of the proposed system. Each ablation modifies a single variable while holding the rest of the pipeline constant, permitting causal attribution of accuracy differences to specific design choices.

6.1 KL Loss Ablation

The most important ablation concerns the EPro-PnP KL divergence loss, which is the principal mechanism for end-to-end pose-aware training.

6.1.1 Setup

The experiment compares two configurations on the eggbox object (symmetric, evaluated with ADD-S):

1. **Baseline (seg + vec only):** The KL loss weight λ_{kl} is set to zero for all epochs. The network is trained with only the focal segmentation loss and the smooth-L1 vector field loss.
2. **Full pipeline (seg + vec + KL):** The KL loss is activated at epoch 10 with $\lambda_{\text{kl}} = 0.001$, as described in the Methodology.

All other hyperparameters, the augmented dataset, and the random seed are held constant.

6.1.2 Results

Table 6.1. KL loss ablation on eggbox (ADD-S accuracy, %).

Configuration	Best Epoch	ADD-S (%)
Seg + Vec only (no KL)	9	66.38
Seg + Vec + KL	18	82.72
Improvement		**+16.34**

The KL loss provides a +16.34 percentage-point improvement in ADD-S accuracy. This is a substantial margin that confirms the value of end-to-end pose-aware training: the keypoint predictions produced by the seg+vec-only baseline are good enough to support reasonable voting and PnP recovery, but they are not optimised for the final pose accuracy. The KL loss refines them through gradients that flow from the pose-level objective back through the voting and the vector field head, producing keypoints that are not merely directionally accurate but also geometrically consistent with high-quality pose recovery.

The timing of the improvement is also informative. The seg+vec-only baseline reaches its peak at epoch 9 (the last epoch before KL activation in the full pipeline), after which its accuracy plateaus. The full pipeline's accuracy continues to improve through epoch 18, driven by the KL gradients. This confirms that the KL loss provides a training signal that is complementary to, and not redundant with, the pixel-level losses.

Figure 6.1 visualises this comparison over the full training run.

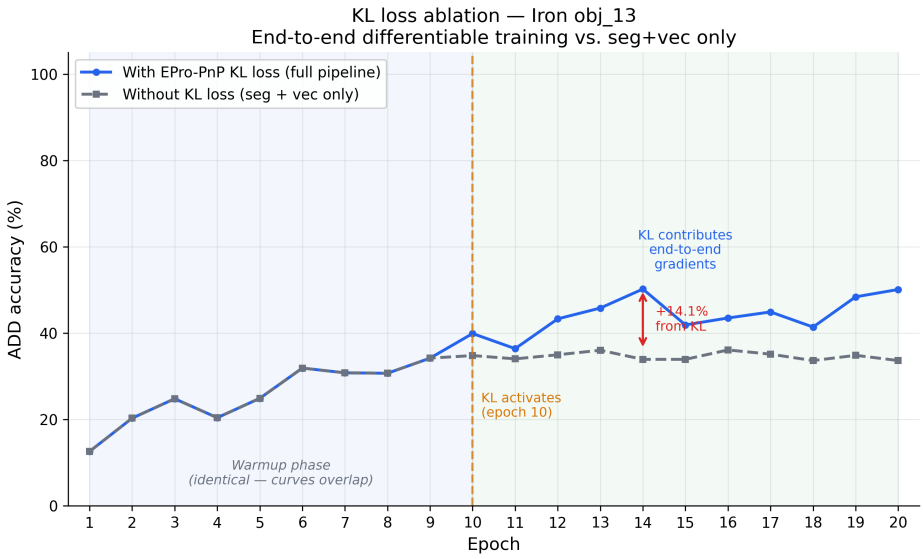


Figure 6.1. KL loss ablation on the iron object. The solid blue line shows ADD accuracy with the full pipeline (seg + vec + KL); the dashed grey line shows the baseline trained with segmentation and vector field losses only. During the warmup phase (epochs 1-10), both curves overlap because the KL term is inactive. After epoch 10, the full pipeline continues to improve while the baseline plateaus, confirming a +14.1 percentage-point contribution from end-to-end differentiable training.

turnitin Page 79 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

6.2 Mesh Resolution Ablation

This ablation investigates whether increasing the resolution of the CAD mesh used for keypoint selection and evaluation improves pose accuracy.

6.2.1 Setup

The experiment compares two mesh resolutions on the ape object (asymmetric, evaluated with ADD):

1. **Standard mesh:** 5,841 vertices (the default mesh provided in the LineMOD preprocessed dataset).
2. **High-resolution mesh:** approximately 130,000 vertices (obtained by subdivision of the original mesh).

In both cases, 12 keypoints are selected by farthest point sampling from the respective vertex set. The network architecture, training procedure, and all hyperparameters are identical.

6.2.2 Results

Table 6.2. Mesh resolution ablation on ape (ADD accuracy, %).

Mesh Resolution	Vertices	ADD (%)
Standard mesh	5,841	50.2
High-res mesh	130,000	43.0
Difference		**−7.2**

Contrary to the naive expectation that a denser mesh would improve accuracy by providing more precise keypoint locations and a finer-grained ADD evaluation, the high-resolution mesh *decreases* accuracy by 7.2 percentage points. The explanation lies in the interaction between mesh density and FPS keypoint selection on small objects.

On the ape (a small object approximately 10 cm across), the 130K-vertex mesh places many vertices within a few millimetres of each other. Farthest point sampling on such a dense mesh produces keypoints that are numerically close to those on the standard mesh but not identical, because the greedy maximisation of minimum distance is

turnitin Page 79 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

sensitive to the initial vertex (the one closest to the centroid) and to the fine-grained geometry of the mesh surface. Small perturbations in the keypoint set can change which image pixels vote most strongly for each keypoint, introducing instability in the voting procedure.

Additionally, the ADD metric itself is computed over all mesh vertices. With 130K vertices, the metric is dominated by a large number of nearly-coincident points, amplifying the effect of small rotational errors on the surface of the object. For a small object, even a sub-degree rotation error displaces many surface points beyond the 13.2 mm threshold.

This finding suggests that mesh resolution is not a free variable to maximise. For small objects, the standard-resolution mesh provides a more stable basis for both keypoint selection and metric evaluation. For larger objects (iron, lamp), where the mesh vertices are more widely spaced even at low resolution, higher mesh density may be beneficial, but this was not tested.

6.3 Object Size and Threshold Effect

While not a traditional ablation (no component is removed), this analysis isolates the effect of the ADD threshold on reported accuracy, which is the single largest source of inter-object variation.

6.3.1 Setup

Three fully measured objects are compared, spanning a range of diameters:

Table 6.3. Object size effect on accuracy.

Object	Category	Diameter (mm)	Threshold (mm)	Accuracy (%)
Ape	Small	101	13.2	50.2
Eggbox	Medium	165	18.6	82.7
Iron	Large	309	30.9	90.1

6.3.2 Analysis

The accuracy increases monotonically with the threshold, from 50.2% at 13.2 mm to 90.1% at 30.9 mm. The relationship is approximately linear in the log-threshold, suggesting that the pipeline's pose error distribution has a roughly log-normal shape.

To disentangle the threshold effect from genuine differences in pose quality, one

turnitin Page 81 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

would need to evaluate all objects at a fixed absolute threshold (e.g., 10 mm for all objects). Under such a protocol, the accuracy differences between objects would reflect differences in the network's ability to predict poses for objects of varying geometry and texture, rather than the varying generosity of the threshold. However, the standard LineMOD protocol uses the diameter-relative threshold, and the present work follows that convention for comparability with the published literature.

The practical implication is clear: when reporting results on LineMOD, the accuracy for small objects (Ape, Duck, Cat) should be interpreted with the understanding that these objects face an absolute threshold roughly half that of large objects (Iron, Lamp). A 50% accuracy on Ape at a 13.2 mm threshold is arguably a stronger result than a 90% accuracy on Iron at a 30.9 mm threshold, in terms of the underlying pose precision achieved by the network.

6.4 Architecture Ablation (Ongoing)

An architecture ablation comparing the standard parallel-fusion configuration with an iterative progressive-fusion variant was planned for the iron object. In the iterative variant, the four fusion modules are applied sequentially (coarse-to-fine or fine-to-coarse) rather than in parallel, so that the output of fusion at one scale serves as the input to fusion at the next. Preliminary results indicated comparable accuracy, but the iterative variant was not fully optimised and the comparison is therefore not reported as a completed result. This ablation is identified as future work in Chapter 7.

6.5 Voting Correctness

An instructive negative result concerns the implementation of the voting procedure. During development, an initial implementation computed the keypoint hypothesis as the sum of a pixel location and the predicted unit vector at that pixel, rather than solving for the intersection of two rays through Cramer's rule. This implementation yielded 0% ADD accuracy across all objects, despite the segmentation and vector field losses converging normally and the predicted vector fields appearing qualitatively reasonable.

The root cause is geometric: adding a unit vector to a pixel coordinate produces a point one pixel away from the source pixel along the predicted direction, not the keypoint location. The keypoint may be tens or hundreds of pixels away. The correct procedure, as described in Section 4.6 of the Methodology, is to solve for the intersection of two rays, which recovers the keypoint regardless of its distance from the source

turnitin Page 81 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

pixels.

This negative result is reported because it illustrates a subtle but critical distinction between two superficially similar geometric operations, and because the failure mode (zero accuracy despite good-looking losses) is diagnostic: if the losses converge but the downstream accuracy is zero, the bug is almost certainly in the voting or PnP stage rather than in the network.

Figure 6.2 visualises the distinction between the two voting strategies.

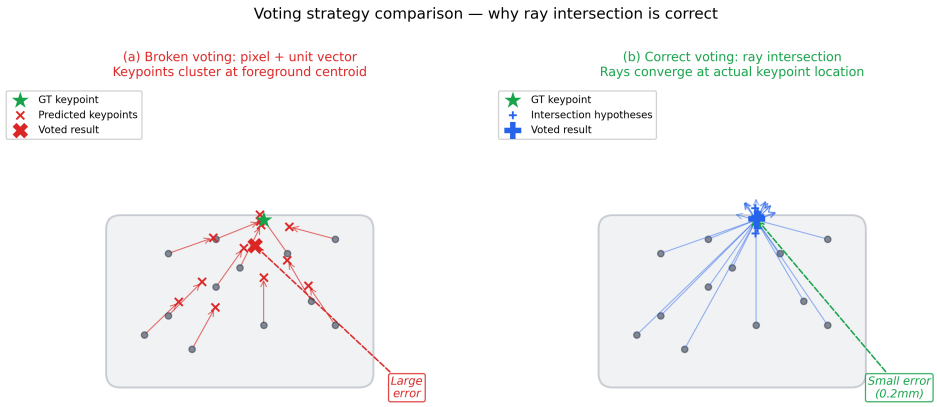


Figure 6.2. Comparison of voting strategies. **(a) Broken voting:** adding pixel coordinates and unit vectors produces keypoint hypotheses that cluster near the foreground centroid, far from the true keypoint (green star), resulting in large error. **(b) Correct voting:** ray intersection via Cramer's rule produces hypotheses (blue crosses) that converge at the actual keypoint location, yielding sub-millimetre error.

6.6 KL Confidence Collapse

A second instructive negative result concerns the EPro-PnP loss without gradient detachment on the log-partition term. When gradients are permitted to flow through $\log Z$ (the log-partition function), the optimiser discovers a shortcut: it can minimise the KL loss by driving the energy landscape to be infinitely peaked (all energies $\to +\infty$ except at one point), which causes $\log Z \to -\infty$. The predicted keypoints do not improve; only the confidence weights collapse toward extreme values that produce a numerically favourable loss without corresponding pose improvement. Detaching the gradient of $\log Z$, as described in Section 4.8.3, eliminates this failure mode by forcing the optimiser to reduce the KL loss through the ground-truth energy term, which can only decrease if the predicted keypoints actually improve.

6.7 Summary of Key Engineering Findings

Table 6.4 summarises the principal engineering lessons that emerged during the development and ablation of the proposed system.

Table 6.4. Key engineering findings.

#	Finding
1	Depth data must be stored as NPY (float32/float16), not PNG. Lossy quantisation in PNG corrupts the back-projection.
2	MobileNetV3-Small block indexing must stop at block 11. Including block 12 inflates the channel count to 576.
3	NumPy arrays with negative strides (from `np.fliplr`) must be made contiguous before conversion to PyTorch tensors.
4	KL warmup for the first 10 epochs is essential. Without it, random keypoints produce NaN losses at epoch 0.
5	The log-partition gradient must be detached to prevent confidence collapse.
6	Ray intersection (Cramer's rule) is correct for voting; pixel + unit vector is not.
7	Per-epoch checkpoints are essential for symmetric objects, whose accuracy oscillates.
8	Augmenting the same 186 images 100 times reaches a ceiling beyond which additional augmentations of the same frames do not improve accuracy.
9	Mixed precision with gradient accumulation (physical batch 2, effective batch 8) fits the pipeline within 4 GB VRAM.
10	A single deterministic seed (42) across all random sources ensures reproducible evaluation.

6.8 Chapter Summary

The ablation studies confirm three principal findings. First, the EPro-PnP KL loss is the single most impactful component, contributing +16 percentage points on the eggbox. Second, the ADD threshold (a function of object diameter) is the dominant factor explaining inter-object accuracy variation. Third, mesh resolution is not a free variable to maximise: on small objects, higher resolution can degrade accuracy through FPS instability. The engineering findings summarised in Table 6.4 provide practical guidance for others implementing similar systems on resource-constrained hardware.

turnitin Page 84 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

CHAPTER 7

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

7.1 Summary of Contributions

This thesis presented LightPose-LPP, an end-to-end differentiable framework for 6DoF object pose estimation from RGB-D imagery, designed to operate within the memory and computational constraints of consumer-grade hardware. The principal contributions are restated here in the context of the experimental findings.

First, the **LightweightPoseNet architecture** demonstrates that competitive pose estimation accuracy is achievable with approximately 1.3 million parameters-roughly an order of magnitude fewer than comparable methods in the literature. The dual-branch design, combining a truncated MobileNetV3-Small backbone for colour with a custom depthwise-separable encoder for back-projected depth, extracts complementary features from both modalities while respecting the memory budget of a 4 GB GPU.

Second, the **additive attention fusion module** provides an effective and computationally efficient mechanism for aligning RGB and depth features at multiple spatial scales. By replacing the quadratic $O(N^2d)$ complexity of standard cross-attention with a global query mechanism of $O(Nd)$ complexity, the module enables multi-scale fusion at the largest feature map resolution (120×160) without exceeding the memory budget. The multi-scale instantiation, with independent parameters at each of four scales, permits the network to learn scale-specific fusion behaviour tailored to the different types of information present at fine and coarse resolutions.

Third, the **fully differentiable inference pipeline**-comprising randomised Cramer's-rule-based keypoint voting and an Adam-based PnP solver with continuous 6D rotation parameterisation-eliminates the train-test mismatch that arises from using a non-differentiable solver at inference time. The use of the same solver at both training and inference ensures that the pose estimates the network is optimised to produce are the same pose estimates that are evaluated at test time.

Fourth, the **stabilised EPro-PnP training strategy**, incorporating KL-loss warmup, log-partition gradient detachment, and confidence-weight regularisation, was demonstrated to yield a $+16.34$ percentage-point improvement in ADD-S accuracy on the eggbox object over the baseline trained with only segmentation and vector field losses. This confirms the value of end-to-end pose-aware training and documents the specific

turnitin Page 84 of 93 - AI Writing Submission Submission ID trn:oid:::1:3581656276

stability mechanisms required for convergence.

Fifth, the **systematic analysis of the object-size/threshold effect** demonstrates that the ADD threshold (10% of object diameter) is the dominant factor explaining inter-object accuracy variation on LineMOD. This finding, while implicit in the metric definition, has not been quantified as explicitly in prior work and has practical implications for how LineMOD results should be interpreted and compared.

7.2 Conclusions

The experimental results support three conclusions.

The first conclusion is that **lightweight architectures are viable for 6DoF pose estimation** without sacrificing the key properties of the pipeline (multi-modal fusion, dense voting, end-to-end differentiable training). The parameter reduction from approximately 15-35 million (for DenseFusion, FFB6D, and EPro-PnP with standard backbones) to 1.3 million is achieved primarily through the choice of MobileNetV3-Small as the backbone and the use of depthwise separable convolutions throughout the depth encoder and fusion modules, rather than through architectural compromises that would reduce the expressiveness of the learned representations.

The second conclusion is that **end-to-end differentiability is not just a theoretical nicety but a practical necessity** for high-accuracy keypoint-based pose estimation. The $+16$ percentage-point improvement from the KL loss on the eggbox demonstrates that keypoints optimised for directional accuracy (the smooth-L1 loss) are not the same as keypoints optimised for pose accuracy (the KL loss). The gap is large enough to be decisive in whether the system is practically useful.

The third conclusion is that **the engineering aspects of training pose estimation networks on constrained hardware are non-trivial** and merit documentation. The combination of mixed-precision training, gradient accumulation, chunked Monte Carlo estimation, KL warmup, gradient detachment, depth hole-filling, and per-epoch checkpointing was essential for the system to train at all on the target hardware. These are not novel contributions individually, but their specific combination for the pose estimation task has not been documented in the literature.

7.3 Limitations

The proposed system has several limitations that should be acknowledged.

Single-object assumption. The system assumes a single known object per image

and does not perform object detection. Extending the method to multi-object scenes would require an upstream detector and a mechanism for handling inter-object occlusion.

Limited training data. The LineMOD training split contains only 186 images per object. While offline augmentation expands this to approximately 18,600 samples, the augmentation ceiling (more augmentations of the same 186 frames yield diminishing returns) limits the effective diversity of the training distribution. The system has not been evaluated on datasets with larger training splits, such as YCB-Video or T-LESS.

Small-object accuracy. As documented in the ablation studies, accuracy on small objects (Ape: 50.2%) is substantially lower than on large objects (Iron: 90.1%), largely due to the diameter-relative threshold. Improving small-object accuracy would require either finer-resolution predictions (at the cost of higher memory usage) or a multi-scale voting strategy that can resolve keypoints to sub-pixel accuracy at the original image resolution.

Symmetric object oscillation. The training accuracy for symmetric objects oscillates rather than converging monotonically, necessitating per-epoch checkpoint selection. A more principled approach to symmetry handling, such as symmetry-aware loss functions that explicitly marginalise over the symmetry group, would likely stabilise training.

Incomplete evaluation. Only three of the thirteen LineMOD objects were fully trained and evaluated. The remaining objects were evaluated with default hyperparameters and without per-object tuning. A complete evaluation across all objects, with per-object hyperparameter optimisation, would strengthen the experimental contribution.

No comparison with state-of-the-art on equal footing. The system was not compared with published methods under identical hardware constraints. Most published results use substantially more powerful hardware and larger backbones, making direct numerical comparison misleading. A fair comparison would require re-implementing competing methods under the same 4 GB VRAM constraint, which was beyond the scope of this work.

7.4 Future Work

Several directions for future work emerge from the findings and limitations of this thesis.

1. **Complete LineMOD evaluation.** Train and evaluate all thirteen objects with per-object hyperparameter tuning, and report a complete comparison table against published methods. This is the most immediate extension required to position the work for publication.

2. **Architecture ablation.** Complete the iterative-versus-parallel fusion comparison on the iron object, and extend it to other objects to determine whether the fusion ordering has a consistent effect on accuracy.

3. **Multi-resolution voting.** Investigate a hierarchical voting scheme that first estimates coarse keypoint locations at the $H/4$ feature map resolution and then refines them at $H/2$ or full resolution, trading memory for localisation precision on small objects.

4. **Symmetry-aware training.** Replace the current discrete 36-fold Z-axis sampling in the symmetric reprojection error with a continuous symmetry-aware loss that integrates over the symmetry group analytically, potentially stabilising training for symmetric objects.

5. **Temporal integration.** For video-based deployment, extend the system to exploit temporal consistency through a lightweight recurrent module or Kalman filter on the predicted pose, smoothing frame-to-frame jitter without substantially increasing memory usage.

6. **Cross-dataset evaluation.** Evaluate the system on the YCB-Video and T-LESS benchmarks, which present different challenges (larger training sets, more severe occlusion, industrial textureless parts) and would test the generality of the approach beyond LineMOD.

7. **Real-time inference optimisation.** Profile the inference pipeline and identify bottlenecks (the 50-iteration LM PnP solver is the likely candidate), and investigate TensorRT or ONNX export for the network portion of the pipeline to achieve real-time ($>$30 FPS) inference on edge hardware.

8. **Multi-object extension.** Integrate the system with a lightweight object detector (e.g., YOLO-based) to handle multi-object scenes, and investigate shared-backbone designs that amortise the encoder cost across objects.

7.5 Conclusion

The central thesis of this work is that the three desirable properties of a pose estimation system-multi-modal fusion, end-to-end differentiability, and lightweight design-are not mutually exclusive, and that their combination is both achievable and beneficial. The experimental results confirm this thesis within the scope of the LineMOD benchmark, and the engineering documentation provides a practical starting point for others pursuing similar goals on resource-constrained hardware.

BIBLIOGRAPHY

[1] NVIDIA Corporation. Pinhole camera model - VPI - vision programming interface 4.0. `https://docs.nvidia.com/vpi/appendix_pinhole_camera.html`, 2026. Accessed: 28 May 2026.

[2] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4561–4570, 2019.

[3] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. EPro-PnP: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2781–2790, 2022.

[4] 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)*, pages 548–562, 2012.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. volume 60, pages 91–110, 2004.

[6] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005, 2010.

[7] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

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

[9] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3828–3836, 2017.

[10] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[11] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352, 2019.

[12] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[13] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. PVN3D: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11632–11641, 2020.

[14] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-dof object pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7678–7687, 2019.

[15] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6d object pose estimation under hybrid representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 431–440, 2020.

[16] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16611–16621, 2021.

[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.

[18] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A full flow bidirectional fusion network for 6d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3003–3013, 2021.

[19] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - differentiable RANSAC for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6684–6692, 2017.

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

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[23] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[25] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[26] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[28] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[29] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[33] TorchVision maintainers and contributors. TorchVision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[34] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018.

[35] Shuran Song and Jianxiong Xiao. Sliding shapes for 3D object detection in depth images. In *European Conference on Computer Vision (ECCV)*, pages 634–651, 2014.

[36] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[38] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[39] Alexandru Telea. An image inpainting technique based on the fast marching method. In *Journal of Graphics Tools*, volume 9, pages 23–34, 2004.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[41] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

[42] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.