MCEN5228 Project 2 - Depth-Dependent Image Convolution

Anh Thy Thi Nguyen
University of Colorado Boulder
Email: anng8974@colorado.edu

Thanushraam Suresh Kumar
University of Colorado Boulder
Email: thsu9525@colorado.edu

***Abstract*—This project aims to enhance monocular depth estimation accuracy by combining phase-coded aperture imaging with a neural network model. The approach leverages depth-aware blur synthesis from RGB images, ground truth depth maps, and custom point spread functions (PSFs).**

Key components include depth-dependent blur synthesis using convolution operations on RGB images with discrete PSFs, followed by training on NYUv2 and UMDCodedVO-LivingRoom datasets. Model evaluation is conducted on out-of-domain datasets (ICL-NUIM and UMDCodedVO-DiningRoom) using quantitative metrics (RMSE, Abs-Rel) and qualitative assessments with 'viridis' color scheme.

This approach improves depth accuracy, particularly at occlusion boundaries, and highlights the potential of coded aperture imaging in monocular depth estimation.

I. Introduction

Depth-dependent convolution is crucial in image processing to simulate real-world effects such as focus blur based on distance from the camera. In this project, we developed and implemented functions for depth quantization, PSF convolution, and depth-aware image capture. This report details the approaches used, including challenges and observations during the implementation.

A. Project Overview

This project is divided into two phases, each with distinct objectives and methods. In the first phase, we focus on generating coded (or blurred) images from standard RGB images using the ground truth metric depth and a phase mask's Point Spread Function (PSF). This process essentially encodes depth information into the image. The second phase has two sections: the first involves training a monocular depth estimation model using All-in-Focus (AiF) RGB images to predict depth on a specified dataset. In the second section, the model is trained using the coded images generated in Phase 1, enabling a comparison of model performance when trained on conventional RGB images versus depth-encoded, coded images.

B. Dataset

In Part 1 of the project, we work with two datasets: NYU-data and LivingRoom. Each dataset contains 1,000 RGB images with corresponding depth information. The NYU Depth Dataset (NYUv2) provides depth in a PNG format, capturing real-world indoor scenes with Microsoft Kinect in grayscale depth units. The LivingRoom dataset, created using Blender, uses the EXR format to represent depth in high dynamic range, which allows for precise depth values and is useful for synthetic, high-accuracy depth estimation tasks in controlled environments.

II. Phase 1: Image Processing and Depth Layer Quantization

A. Depth Map Quantization

The initial phase aimed to quantize a depth map into distinct depth layers.

- **Input Data**: The input was a depth map of shape (H, W) and depth layers defined by discrete values.
- **Quantization Approach**: Depth values were quantized into discrete bins, mapping each pixel to a depth layer index for further processing.
- **Implementation**: The function returned binary masks stacked using `torch.stack()` to create a tensor of shape (D, H, W).

B. Single PSF Convolution

- **Input Image and PSF Selection**: The input image tensor (N, C, H, W) and a PSF tensor (D, C, H, W) were used.
- **2D Convolution**: Convolution was performed using PyTorch's `conv2d()` function, with padding to control output size.
- **Results**: The convolution produced an output highlighting depth-specific effects. Figure 1-10 shows the convolution results.

III. Phase 2: Linear Depth-Dependent Convolution

A. Linear Convolution Over All Depth Layers

- **Depth Masks**: Binary masks identified which pixels belonged to specific depth layers.
- **Implementation**: The function performed convolution over each depth layer, summing the results to create the final output.
- **Output**: Figure 1-10 shows the linear depth-dependent convolution outputs for test images.

IV. Results

The following images show the results of the depth-dependent convolution implementation for specific test images from the dataset. The outputs demonstrate the depth-aware blurring effects achieved through the linear convolution model, with blur intensity varying according to depth.

Fig. 1: Depth-dependent convolution output for Image 1

Fig. 4: Depth-dependent convolution output for Image 30

Fig. 2: Depth-dependent convolution output for Image 10

Fig. 5: Depth-dependent convolution output for Image 40

Fig. 3: Depth-dependent convolution output for Image 20

Fig. 6: Depth-dependent convolution output for Image 1

Fig. 7: Depth-dependent convolution output for Image 10

Fig. 8: Depth-dependent convolution output for Image 20

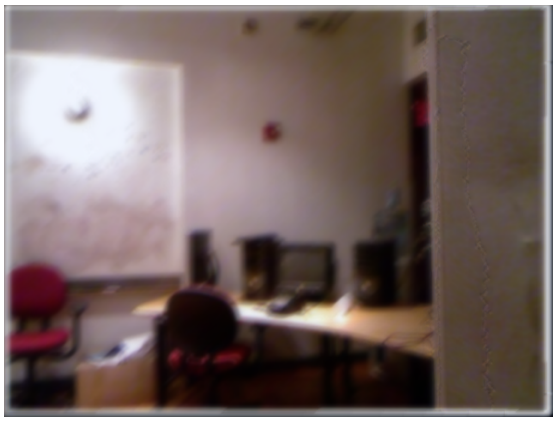

Fig. 9: Depth-dependent convolution output for Image 30

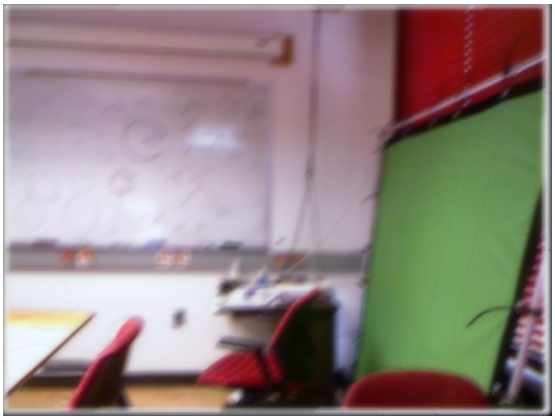

Fig. 10: Depth-dependent convolution output for Image 40

V. Challenges and Observations

A. Handling Outliers in Depth Values

Outliers in depth values posed challenges for accurate PSF selection, leading to unintended artifacts. To mitigate this, depth values were clamped within valid ranges.

B. Computational Complexity

The convolution process was computationally intensive, particularly for high-resolution images. Optimization opportunities include parallel processing and PSF selection improvements.

VI. Part 2: Depth Estimation with Coded RGB and Pinhole RGB Images

A. Experiment Overview

In Part 2, the goal was to train a monocular depth estimation model with:

1) **All-in-focus (AiF) RGB images and Ground Truth Depth maps:** This experiment used pinhole model RGB images.
2) **Coded RGB images and Ground Truth Depth maps:** Using the coded aperture images generated from the Part 1 methodology.

Both experiments were conducted using the NYUv2 and UMDCodedVO-LivingRoom datasets for training, with evaluations performed on the out-of-domain ICL-NUIM and UMDCodedVO-DiningRoom datasets. CUDA-enabled hardware was leveraged for optimal processing speed and efficiency.

B. Data Preprocessing and Handling

The ImageDepthDataset class was used to manage image-depth pairs, supporting memory caching and on-the-fly processing for performance optimization. Key preprocessing steps include:

- **Image and Depth Pair Loading:** Coded RGB images and corresponding depth maps were scaled and normalized.
- **Data Transformation:** A central cropping operation ensured uniform image size, enhancing model accuracy and consistency across different datasets.

C. Model Architecture

The U-Net architecture, a widely used deep learning model for image segmentation, was adapted for depth estimation. The architecture is structured as follows:

- **Convolutional Blocks (ConvBlock):** Each ConvBlock has two convolutional layers followed by batch normalization and ReLU activation, ensuring effective feature extraction while maintaining spatial dimensions for depth prediction. The ConvBlocks were placed throughout both encoder and decoder paths.
- **Encoder Path:** The encoder consists of five ConvBlocks, where channel dimensions increase progressively (from 3 to 1024) to capture higher-level spatial features. Each block is followed by max-pooling for downsampling.
- **Decoder Path:** The decoder uses transposed convolutions for upsampling, with skip connections between encoder and decoder layers to preserve spatial details. Each concatenated feature map from the encoder and decoder paths passes through a ConvBlock for refined reconstruction.
- **Output Layer:** A final 1x1 convolutional layer reduces channels to the depth map output.

D. Training and Loss Function

The model was trained using a weighted mean squared error (MSE) loss function:

$$\text{Weighted MSE} = \sum \left(2^{0.3 \cdot \text{depth}} \cdot (\text{prediction} - \text{ground truth})^2\right) \tag{1}$$

This depth-aware loss scales error based on depth, emphasizing accurate prediction for near-field objects.

E. Training Procedure

The model training was configured as follows:

- **Number of Epochs:** 45
- **Batch Size:** 1
- **Learning Rate:** 0.0001
- **Optimizer:** Adam optimizer, allowing for adaptive learning rates and reduced convergence time.

A checkpoint-based approach was used, saving the model with the lowest validation loss for robust performance.

VII. Results and Evaluation

Evaluation was conducted using quantitative metrics such as RMSE, Abs-Rel, and FPS. Qualitative results were visualized with the 'viridis' color scheme to demonstrate depth accuracy across different testing datasets.

A. Quantitative Results

The model's performance was evaluated across two sections, each containing both the Dining and Corridor datasets. The following metrics were obtained:

Part 2 - Section 1

- **Dining Dataset:**
  - **Absolute Relative Error (Abs-Rel):** The model recorded an Abs-Rel of 45.3%.
  - **Root Mean Square Error (RMSE):** The RMSE was measured at 1.164, representing an average error of approximately 1.164 meters.
  - **Frames per Second (FPS):** The model achieved 348 FPS
- **Corridor Dataset:**
  - **Absolute Relative Error (Abs-Rel):** For the Corridor dataset, the model recorded an Abs-Rel of 114%.
  - **Root Mean Square Error (RMSE):** The RMSE was measured at 1.428, indicating an average error of approximately 1.428 meters.
  - **Frames per Second (FPS):** The model processed 307 FPS.

PART 2 - SECTION 2

- **Dining Dataset:**
  - **Absolute Relative Error (Abs-Rel):** An Abs-Rel of 23.1% was observed, better than compared to section 1
  - **Root Mean Square Error (RMSE):** The RMSE was recorded to be 0.765.
  - **Frames per Second (FPS):** The model processed 380 FPS.
- **Corridor Dataset:**
  - **Absolute Relative Error (Abs-Rel):** The model recorded an Abs-Rel of 91.8%.
  - **Root Mean Square Error (RMSE):** The RMSE was 1.20
  - **Frames per Second (FPS):** The model processed 393 FPS.

TABLE I: Quantitative Results Comparison for Dining Room and Corridor Datasets

Model	Metric	Dining Room	Corridor
AiF (Ours)	Abs-Rel	0.4532	1.1486
	RMSE	1.1642	1.4284
	FPS	348.5	307.6
CodedDepth (Ours)	Abs-Rel	0.2317	0.9185
	RMSE	0.7653	1.2028
	FPS	380.5	393.7

B. Qualitative Results

Figures 13 and 14 show depth estimation outputs on the different model.

VIII. CONCLUSION

We successfully implemented a depth-dependent convolution system for image processing, simulating realistic depth-based effects. Challenges such as computational demands and outlier handling were addressed. Future work will focus on enhancing blending techniques and improving efficiency.

Fig. 11: Our Depth Prediction after 45 Epoch

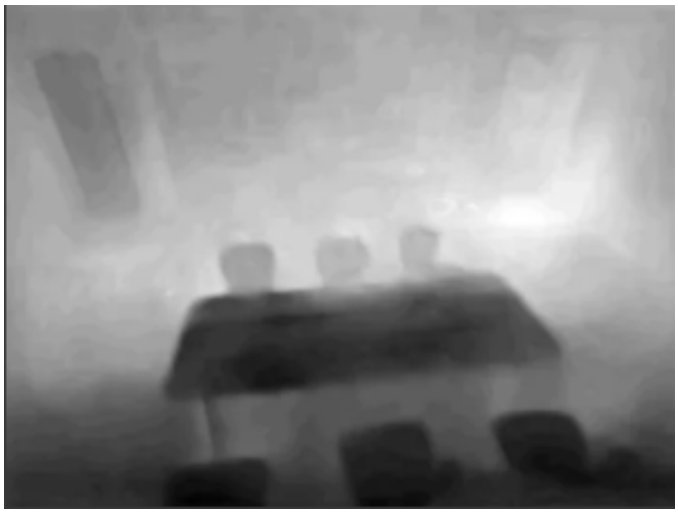

Fig. 12: CodedVO Paper's Depth Prediction after 45 Epoch

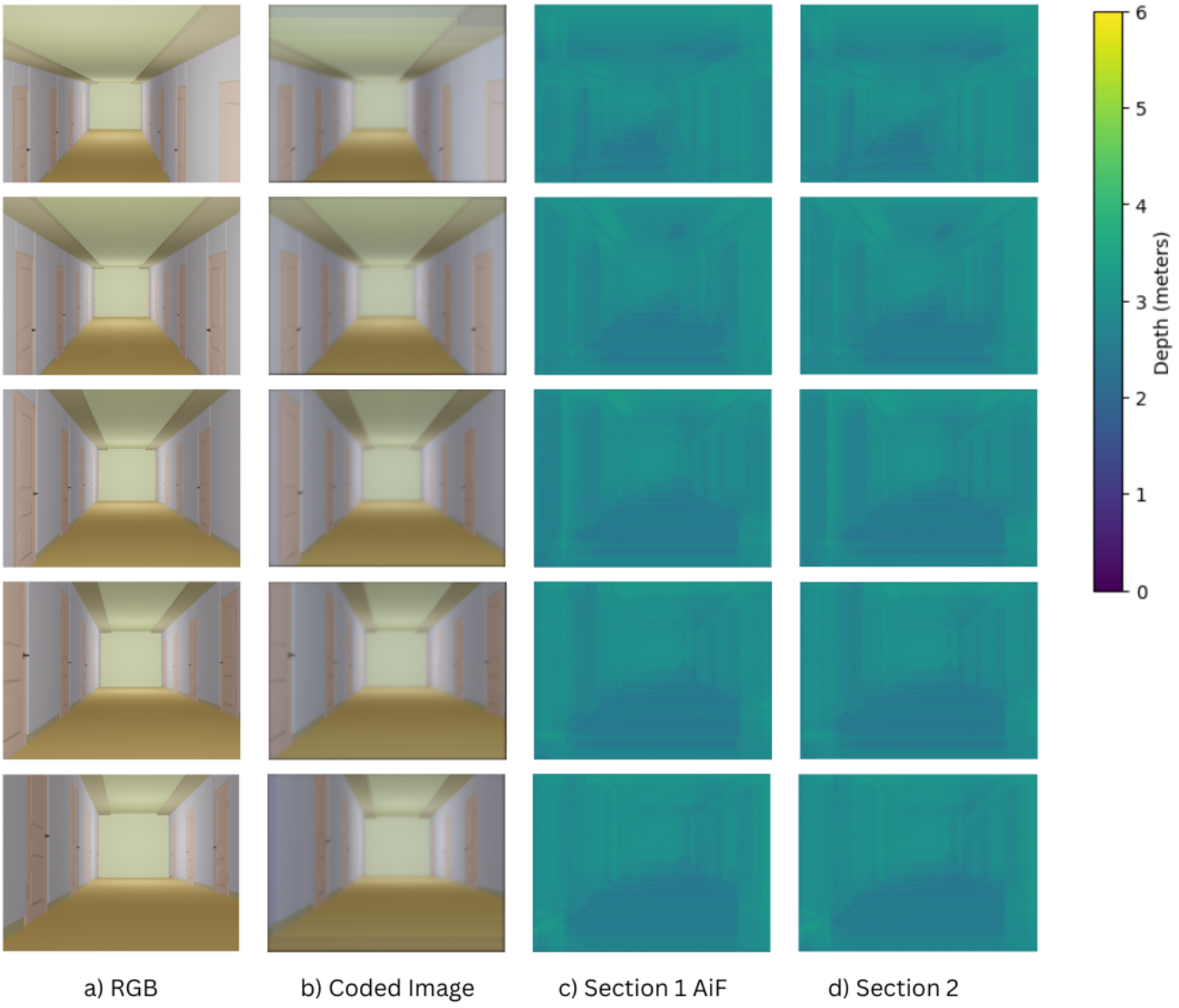

Fig. 13: Corridor Dataset

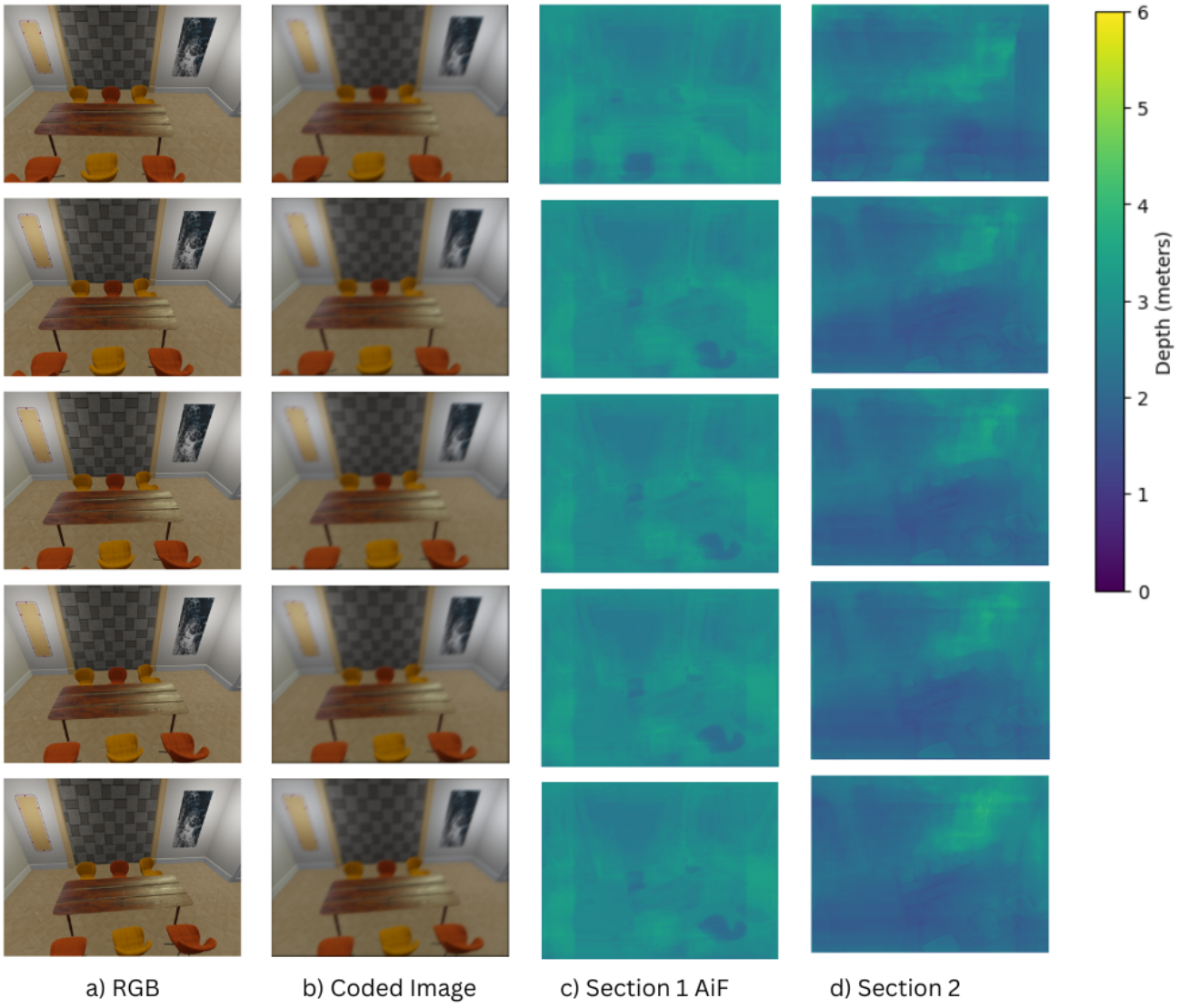

Fig. 14: Dining Dataset