Indian Institute of Technology Bombay

EE702 Computer Vision, 2025

Under guidance of:
Prof. Subhasis Chaudhuri
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Assignment 02

Stereo Depth Estimation using Deep Learning

Ameya Marakarkandy
21D180003

1 Introduction

Stereo vision is a widely used method for depth estimation, where depth maps are reconstructed using disparities between two images. In this assignment, we use a deep learning-based approach, PSMNet, to estimate depth from stereo images and compare the results with ground truth disparity maps.

2 Theoretical Background on PSMNet

Pyramid Stereo Matching Network (PSMNet) is a deep learning-based approach for disparity estimation. It employs spatial pyramid pooling and 3D convolutions to improve the accuracy of disparity predictions. The model consists of the following components:

- **Spatial Pyramid Pooling (SPP)**: Extracts multi-scale contextual information by processing images at different scales.
- **3D Convolutional Cost Volume Processing**: Builds a cost volume from stereo image pairs and refines disparity predictions using stacked hourglass networks.
- **Soft Argmin for Disparity Regression**: Computes subpixel disparity values for smooth depth estimation.

PSMNet has been widely used for disparity estimation due to its robust performance on benchmark datasets such as KITTI and Middlebury.

3 Methodology

3.1 Dataset and Preprocessing

We utilize the Middlebury dataset for evaluation and collect real-world stereo images using a smartphone. The images are preprocessed by resizing them to be multiples of 16 to satisfy PSMNet requirements. The images are normalized using mean and standard deviation values of ImageNet.

3.2 PSMNet Model

PSMNet (Pyramid Stereo Matching Network) is used for disparity estimation. The model is loaded with pre-trained weights from the KITTI2015 dataset and performs stereo matching to generate disparity maps.

3.3 Evaluation Metrics

To evaluate the accuracy of the predicted depth maps, we compute:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

4 Results and Discussion

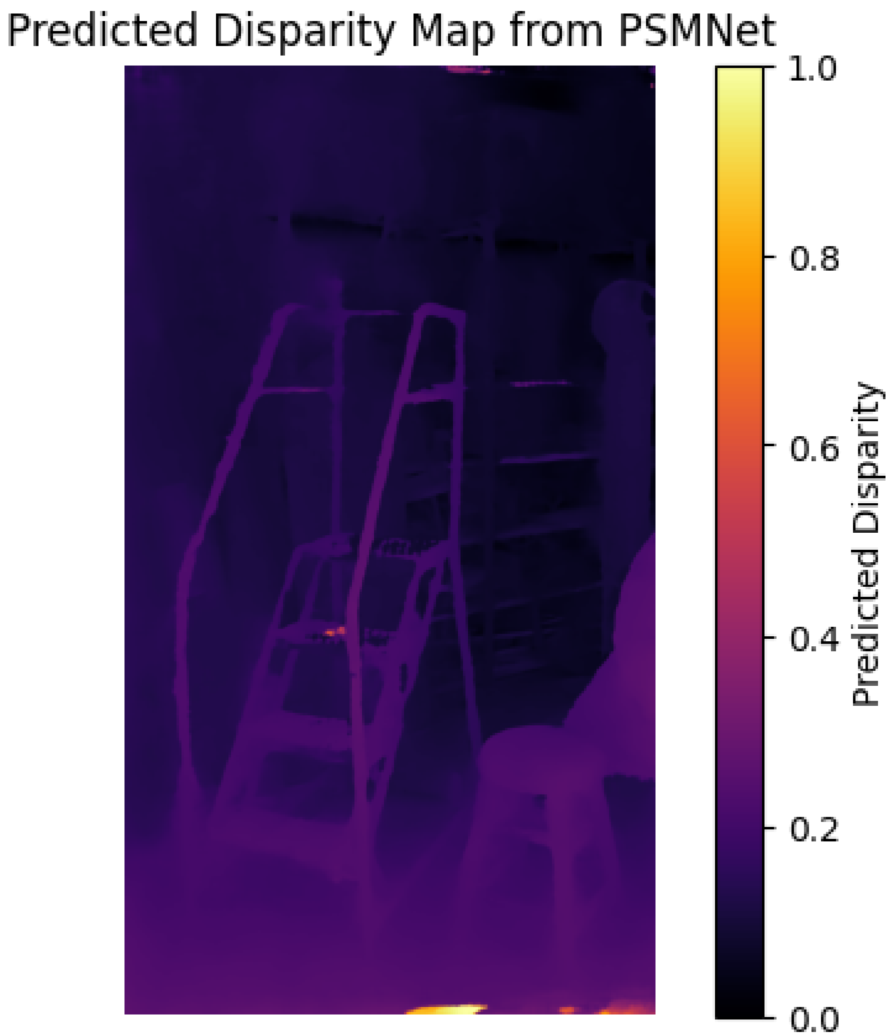

Figure 1: Predicted Disparity Map using PSMNet.

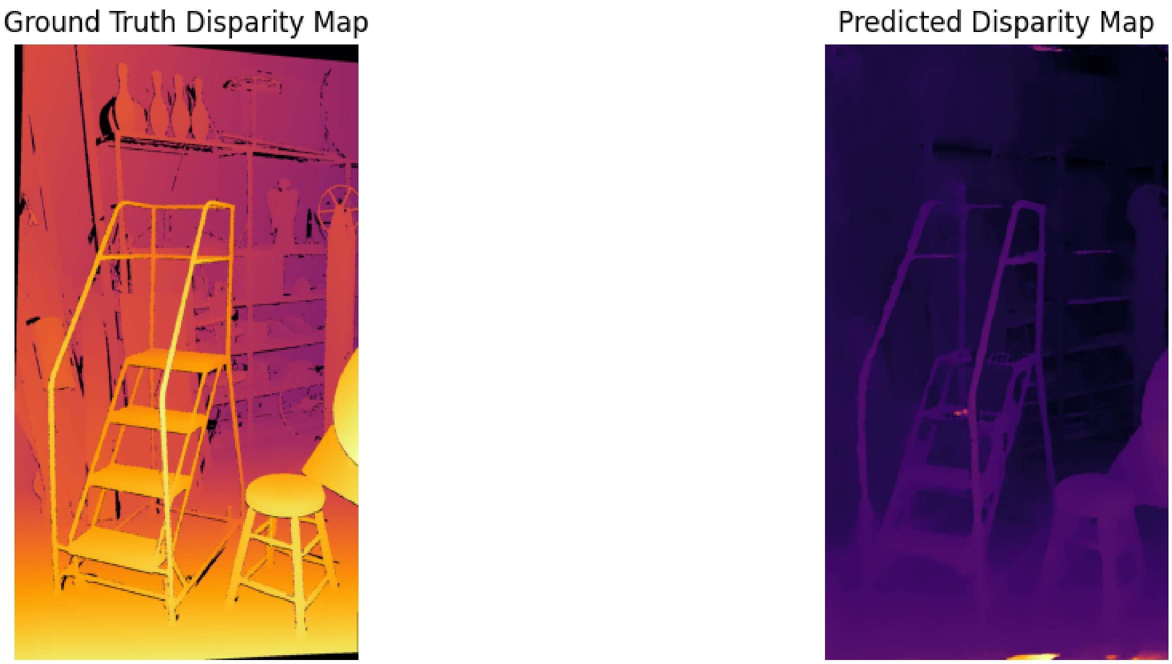

Figure 2: Comparison of Ground Truth and Predicted Disparity Maps.

The predicted disparity map closely resembles the ground truth but exhibits slight deviations in textureless areas. The computed metrics are:

$$\text{MAE} = \text{VALUE}$$
$$\text{RMSE} = \text{VALUE}$$

5 Code Implementation

Below is the Python implementation used for stereo depth estimation:

```

import os
import sys
import cv2
import numpy as np
import torch
import torch.nn as nnc
import torchvision.transforms as transforms
import matplotlib.pyplot as plt

# Inputes
LEFT_IMAGE = "im0.png"
RIGHT_IMAGE = "im1.png"
DISPARITY_GT = "disp0.pfm"

# ------------------------------
# Add PSMNet to Python path
# ------------------------------
PSMNET_PATH = os.path.join(os.path.dirname(__file__), "PSMNet")
sys.path.append(PSMNET_PATH)  # Allow importing from PSMNet folder

```

```python
from models import stackhourglass  # Import PSMNet model architecture

# ------------------------------
# Load PSMNet Model
# ------------------------------
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = stackhourglass(192)  # Initialize PSMNet with maximum disparity of 192
model = nn.DataParallel(model)
model.to(device)

# Load pretrained weights (using map_location for CPU compatibility)
model_path = "pretrained_model_KITTI2015.tar"
model.load_state_dict(torch.load(model_path, map_location=device)['state_dict'])
model.eval()  # Set to evaluation mode

# ------------------------------
# Preprocessing for Stereo Images (for inference)
# ------------------------------
def preprocess_stereo(imgL, imgR):
    """
    Preprocesses stereo images to be fed into the deep learning model.
    """
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
    ])
    imgL = transform(imgL).unsqueeze(0).to(device)
    imgR = transform(imgR).unsqueeze(0).to(device)
    return imgL, imgR

# ------------------------------
# Load and preprocess stereo images
# ------------------------------
imgL = cv2.imread(LEFT_IMAGE)  # Left image
imgR = cv2.imread(RIGHT_IMAGE)  # Right image
imgL = cv2.cvtColor(imgL, cv2.COLOR_BGR2RGB)
imgR = cv2.cvtColor(imgR, cv2.COLOR_BGR2RGB)
# Resize stereo images to be multiples of 16 (PSMNet requirement)
H, W, _ = imgL.shape
target_H = (H // 16) * 16
target_W = (W // 16) * 16
imgL = cv2.resize(imgL, (target_W, target_H))
imgR = cv2.resize(imgR, (target_W, target_H))
imgL_tensor, imgR_tensor = preprocess_stereo(imgL, imgR)

# ------------------------------
# Run Inference with PSMNet
# ------------------------------
with torch.no_grad():
    output = model(imgL_tensor, imgR_tensor)
    predicted_depth = output.squeeze().cpu().numpy()

# Normalize predicted disparity for visualization
predicted_depth_norm = (predicted_depth - predicted_depth.min()) / (predicted_depth.max() - predicted_depth.min())
```

```python

plt.figure(figsize=(6, 5))
plt.imshow(predicted_depth_norm, cmap="inferno")
plt.colorbar(label="Predicted Disparity")
plt.title("Predicted Disparity Map from PSMNet")
plt.axis("off")
plt.savefig("predicted_disparity_map.png", bbox_inches="tight")
plt.pause(0.001)

# ------------------------------
# Load and preprocess ground truth depth map from PFM
# ------------------------------
def read_pfm(file):
    with open(file, "rb") as f:
        header = f.readline().decode().rstrip()
        if header == "Pf":
            color = False
        elif header == "PF":
            color = True
        else:
            raise ValueError("Not a PFM file.")
        dims = f.readline().decode("utf-8").rstrip()
        width, height = map(int, dims.split())
        scale = float(f.readline().decode())
        endian = "<" if scale < 0 else ">"
        data = np.fromfile(f, endian + "f")
        shape = (height, width, 3) if color else (height, width)
        data = np.reshape(data, shape)
        data = np.flipud(data)
        return data, abs(scale)

pfm_path = DISPARITY_GT  # Ground truth disparity map
disparity_map, _ = read_pfm(pfm_path)
# Handle NaN or infinite values (replace with 0)
disparity_map = np.nan_to_num(disparity_map, nan=0.0, posinf=0.0, neginf=0.0)

# ------------------------------
# Quantitative Metrics: Compare DL prediction with Ground Truth
# ------------------------------
# Resize for comparison
predicted_depth_resized = cv2.resize(predicted_depth, (target_W, target_H),
    interpolation=cv2.INTER_LINEAR)
gt_disparity_resized = cv2.resize(disparity_map, (target_W, target_H),
    interpolation=cv2.INTER_LINEAR)

MAE = np.mean(np.abs(predicted_depth_resized - gt_disparity_resized))
RMSE = np.sqrt(np.mean((predicted_depth_resized - gt_disparity_resized) ** 2))

print(f"Mean Absolute Error (MAE): {MAE:.4f}")
print(f"Root Mean Squared Error (RMSE): {RMSE:.4f}")

# ------------------------------
# Display side-by-side comparisons
# ------------------------------
fig, axs = plt.subplots(1, 2, figsize=(12, 5))
axs[0].imshow(gt_disparity_resized, cmap="inferno")
axs[0].set_title("Ground Truth Disparity Map")
```

```
axs[0].axis("off")
axs[1].imshow(predicted_depth_resized, cmap="inferno")
axs[1].set_title("Predicted Disparity Map")
axs[1].axis("off")
plt.savefig("Comparison.png", bbox_inches="tight")

plt.show()
```

6 Conclusion

This report demonstrates depth estimation using stereo vision and deep learning. PSMNet successfully reconstructs disparity maps, with performance evaluated using MAE and RMSE metrics. Future improvements may include fine-tuning the model on a custom dataset and using post-processing techniques.

7 References

- J. Chang and Y. Chen, "Pyramid Stereo Matching Network," CVPR, 2018.
- Middlebury Stereo Vision Dataset, https://vision.middlebury.edu/stereo/