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 extends traditional stereo matching pipelines by incorporating global context information through spatial pyramid pooling. It then leverages a stacked hourglass 3D CNN architecture to regularize the cost volume in a top-down and bottom-up manner. This allows the model to better handle ambiguous and textureless regions that traditional approaches struggle with. The final disparity map is obtained through a soft argmin operation over the refined cost volume.

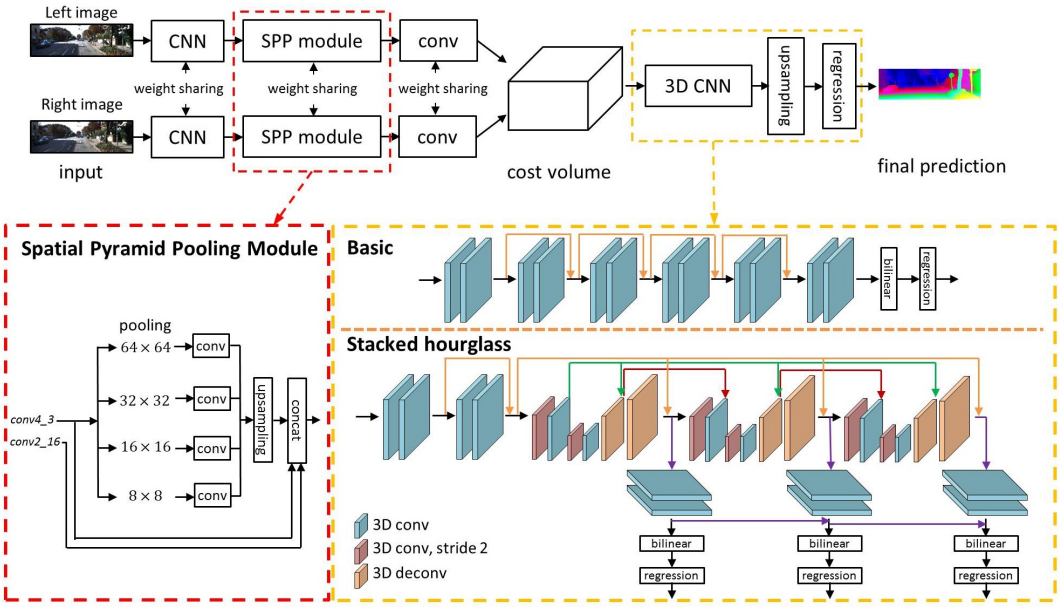

Figure 1: Architecture of PSMNet, consisting of feature extraction, cost volume formation, and disparity regression.

3 Methodology

3.1 Dataset and Preprocessing

We utilize the Middlebury 2021 Stereo dataset for evaluation which employe a mobile device (Apple iPod touch 6G) mounted on a UR5 robot arm to capture stereo images odf scenes. 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.

(a) Left Image

(b) Right Image

Figure 2: Original Stereo Image Pair

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)
- Bad-3 Error: The percentage of pixels where the predicted disparity error exceeds 3 pixels.

4 Results and Discussion

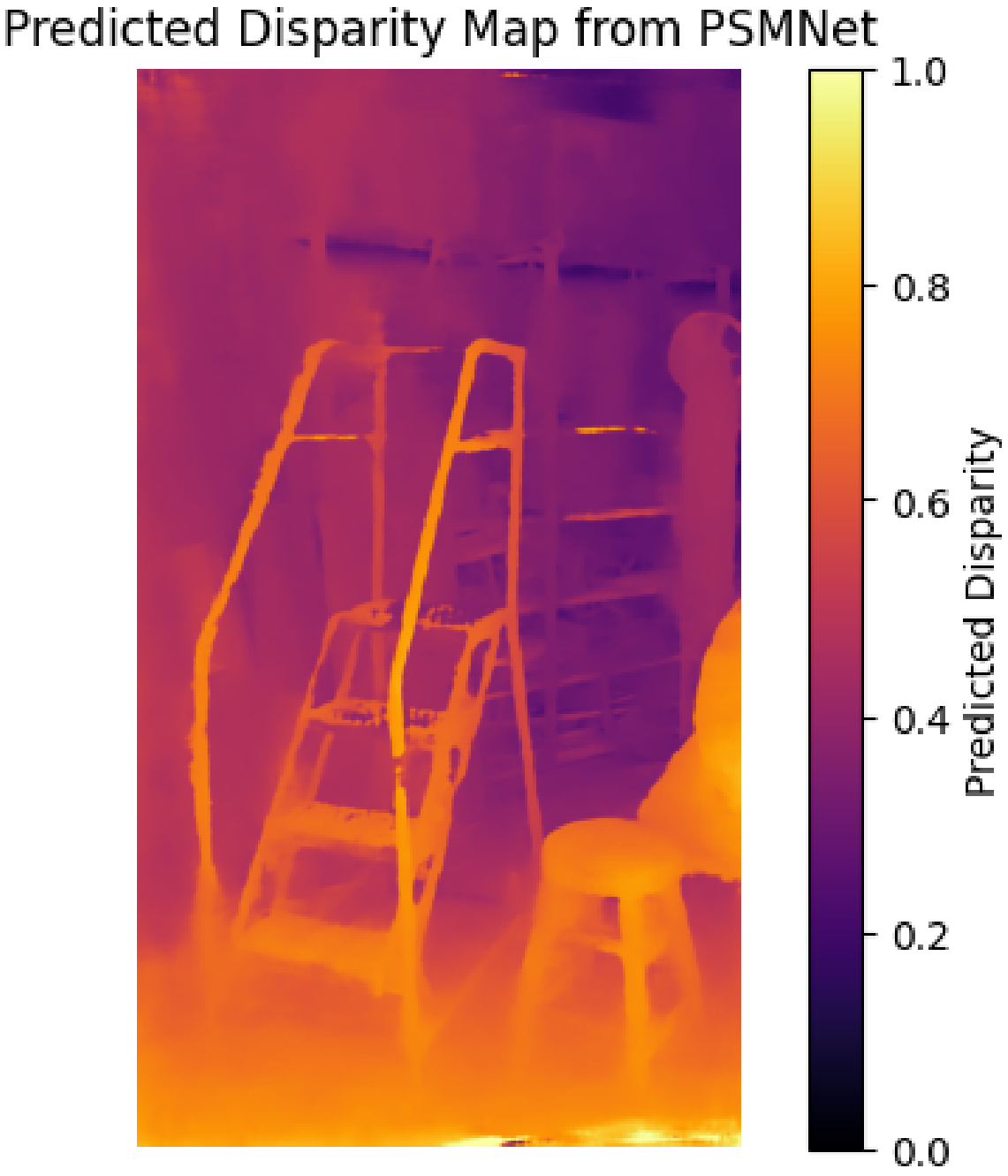

Figure 3: Predicted Disparity Map using PSMNet.

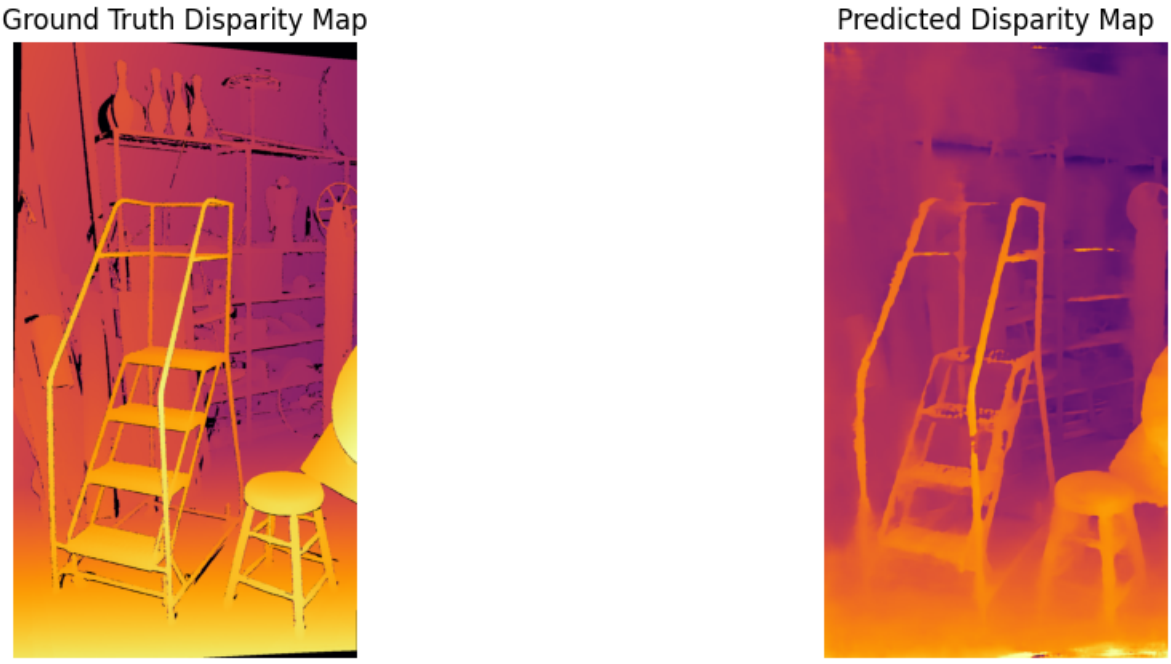

Figure 4: Comparison of Ground Truth and Predicted Disparity Maps.

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

$$\begin{aligned} \text{MAE} &= 3.048 \text{ pixels} \\ \text{RMSE} &= 5.5937 \text{ pixels} \\ \text{Bad-3 Error} &= 26.93\% \end{aligned}$$

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 nn
import torchvision.transforms as transforms
import matplotlib.pyplot as plt

# Inputs
LEFT_IMAGE = "testcases/image1/im0.png"
RIGHT_IMAGE = "testcases/image1/im1.png"
DISPARITY_GT = "testcases/image1/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
```

```
21
22 from models import stackhourglass  # Import PSMNet model architecture
23
24 # ------------------------------
25 # Load PSMNet Model
26 # ------------------------------
27 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
28 model = stackhourglass(80)  # Initialize PSMNet with maximum disparity of 80
29 model = nn.DataParallel(model)
30 model.to(device)
31
32 # Load pretrained weights (using map_location for CPU compatibility)
33 model_path = "pretrained_model_KITTI2015.tar"
34 model.load_state_dict(torch.load(model_path, map_location=device)['state_dict'
   ])
35 model.eval()  # Set to evaluation mode
36
37 print("Model Loaded")
38
39 # ------------------------------
40 # Preprocessing for Stereo Images (for inference)
41 # ------------------------------
42 def preprocess_stereo(imgL, imgR):
43     """
44     Preprocesses stereo images to be fed into the deep learning model.
45     """
46     transform = transforms.Compose([
47         transforms.ToTensor(),
48         transforms.Normalize(mean=[0.485, 0.456, 0.406],
49                              std=[0.229, 0.224, 0.225])
50     ])
51     imgL = transform(imgL).unsqueeze(0).to(device)
52     imgR = transform(imgR).unsqueeze(0).to(device)
53     return imgL, imgR
54
55 # ------------------------------
56 # Load and preprocess stereo images
57 # ------------------------------
58 imgL = cv2.imread(LEFT_IMAGE)  # Left image
59 imgR = cv2.imread(RIGHT_IMAGE)  # Right image
60 imgL = cv2.cvtColor(imgL, cv2.COLOR_BGR2RGB)
61 imgR = cv2.cvtColor(imgR, cv2.COLOR_BGR2RGB)
62 # Resize stereo images to be multiples of 16 (PSMNet requirement)
63 H, W, _ = imgL.shape
64 target_H = (H // 16) * 16
65 target_W = (W // 16) * 16
66 imgL = cv2.resize(imgL, (target_W, target_H))
67 imgR = cv2.resize(imgR, (target_W, target_H))
68 imgL_tensor, imgR_tensor = preprocess_stereo(imgL, imgR)
69
70 print("Testcase Loaded")
71
72 # ------------------------------
73 # Run Inference with PSMNet
74 # ------------------------------
75 with torch.no_grad():
76     output = model(imgL_tensor, imgR_tensor)
```

```
    predicted_depth = output.squeeze().cpu().numpy() # * 1.17  # Apply correction factor

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

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))

valid_mask = (gt_disparity_resized > 0) & (predicted_depth_resized > 0)  # Ignore invalid values
```

```
MAE = np.mean(np.abs(predicted_depth_resized[valid_mask] - gt_disparity_resized
    [valid_mask]))
RMSE = np.sqrt(np.mean((predicted_depth_resized[valid_mask] -
    gt_disparity_resized[valid_mask]) ** 2))


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

bad_3 = np.mean(np.abs(predicted_depth_resized[valid_mask] -
    gt_disparity_resized[valid_mask]) > 3) * 100
print(f"Bad-3 Error: {bad_3:.2f}% of pixels have disparity error > 3 pixels")

# ------------------------------
# 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()
```

The code and testcases can be found on my GitHub repository Stereo-Depth-Vision

6 Conclusion

The evaluation of PSMNet's performance shows that while the model achieves a reasonable level of accuracy, there is room for improvement. The Mean Absolute Error (MAE) of 3.0483 pixels and the Root Mean Squared Error (RMSE) of 5.5937 pixels indicate that while most predictions are relatively close to the ground truth, there are significant outliers causing higher RMSE values. Additionally, the Bad-3 error of 26.93% highlights that over a quarter of the pixels have a disparity estimation error greater than 3 pixels, which may lead to inaccuracies in depth reconstruction, particularly in textureless or occluded regions. Future improvements could include fine-tuning the network on a custom dataset that better represents real-world scenarios, incorporating additional post-processing techniques to refine disparity maps, and experimenting with hybrid approaches that integrate classical stereo matching algorithms with deep learning methods.

7 References

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