CSE616 – Neural Networks And Their Applications

Project 1 (1)

Pyramid Stereo Matching Network

Name	Mohamed Ehab Fathy Abd El-Wahaab
ID	2002597

Disparity Estimation Using Pyramid Stereo Matching Network

1. Overview

Depth estimation from stereo images is essential to computer vision applications. Given a pair of rectified stereo images, the goal of depth estimation is to compute the disparity d. Disparity refers to the horizontal displacement between a pair of corresponding pixels on the left and right images for each pixel in the reference image. The traditional pipeline for stereo matching involves the finding of corresponding points based on matching cost and post-processing.Although CNN yields significant gains compared to conventional approaches in terms of both accuracy and speed, it is still difficult to find accurate corresponding points in inherently ill-posed regions such as occlusion areas, repeated patterns, textureless regions, and reflective surfaces. Solely applying the intensity-consistency constraint between different viewpoints is generally insufficient for accurate correspondence estimation in such ill-posed regions, and is useless in textureless regions. Therefore, regional support from global context information must be incorporated into stereo matching.

In this project, a novel pyramid stereo matching network (PSMNet) is used to exploit global context information in stereo matching. Spatial pyramid pooling and dilated convolution are used to enlarge the receptive fields. In this way, PSMNet extends pixel-level features to region-level features with different scales of receptive fields; the resultant combined global and local feature clues are used to form the cost volume for reliable disparity estimation. Moreover, we design a stacked hourglass 3D CNN in conjunction with intermediate supervision to regularize the cost volume. The stacked hourglass 3D CNN repeatedly processes the cost volume in a top-down/bottomup manner to further improve the utilization of global context information.

2. Detailed Architecture

2.1. Overall Architecture

As shown in figure 1. PSMNet consists of CNN layers for unary features extraction, then an SPP module for effective incorporation of global context and a stacked hourglass module for cost volume regularization, and finally output regression layer for disparity prediction.

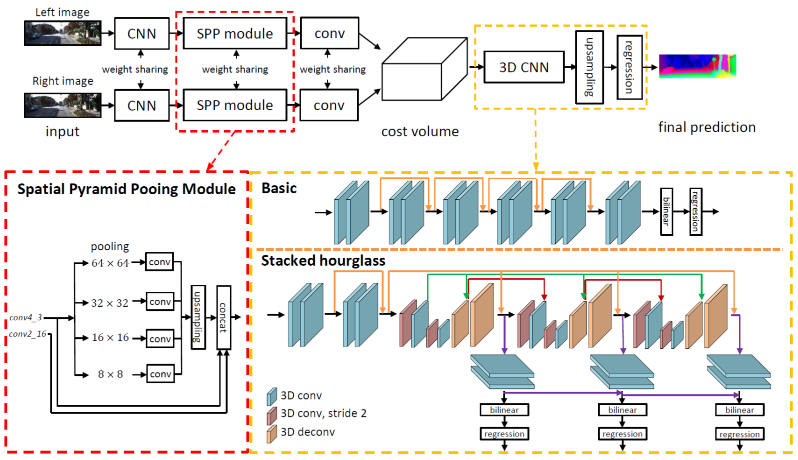
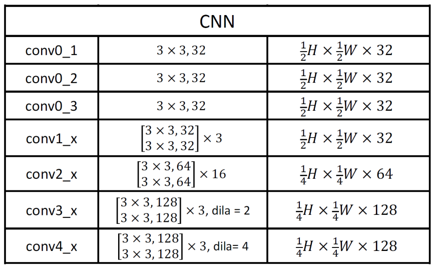


Figura 1: Overall Architecture

2.2. CNN Layer

Three small convolution filters (3×3) are cascaded to construct a deeper network with the same receptive field. The $conv1\_x$, $conv2\_x$, $conv3\_x$, and $conv4\_x$ are the basic residual for learning the unary feature extraction. for $conv3\_x$ and $conv4\_x$, dilated convolution is applied to further enlarge the receptive field.

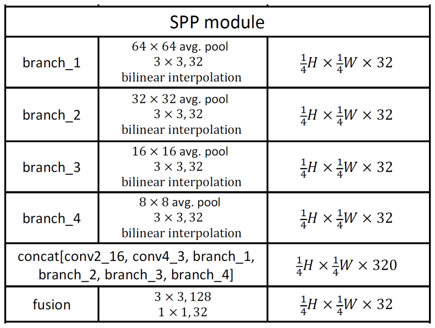


CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$

Figura 2: CNN layer

2.3. Spatial Pyramid Pooling Layer

The SPP module uses adaptive average pooling to compress features into four scales and is followed by a (1×1) convolution to reduce feature dimension, after which the low-dimensional feature maps are upsampled to the same size of the original feature map via bilinear interpolation. The different levels of feature maps are concatenated as the final SPP feature maps.

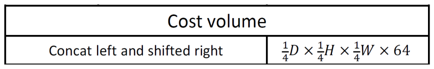


SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$

Figura 3: SPP layer (Four fixed-size average pooling blocks for SPP: 64×64, 32×32, 16×16, and 8×8 are used).

2.4. Cost Volume Layer

Rather than using a distance metric, the left and right features are concatenated to learn matching cost estimation using deep network. we SPP features are adopted to form a cost volume by concatenating left feature maps with their corresponding right feature maps across each disparity level, resulting in a 4D volume ($height \times width \times disparity \times feature\,size$).

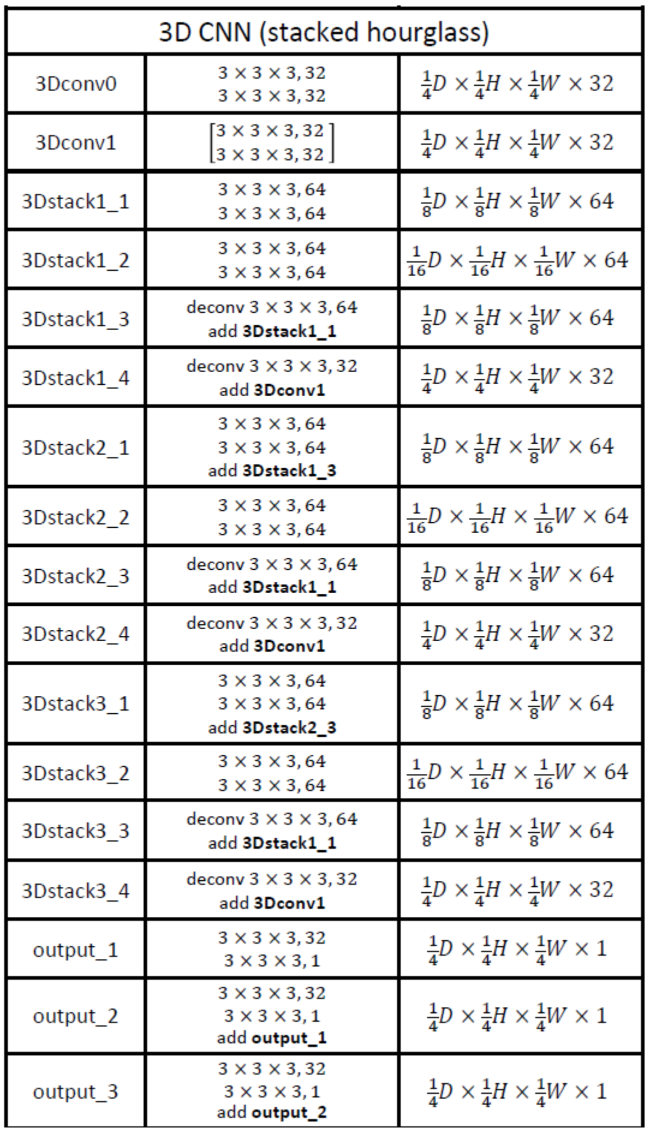


Cost volume	
Concat left and shifted right	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$

Figura 4: Cost Volume Layer

2.5. 3D CNN Layer

The SPP module facilitates stereo matching by involving different levels of features. 3D CNN layer is used aggregate the feature information along the disparity dimension as well as spatial dimensions, for cost volume regularization: there are two options the basic and stacked hourglass architectures. In the basic architecture, as shown in Figure 1, the network is simply built using twelve residual blocks. . Then cost volume is upsampled the back to size $H \times W$ via bilinear interpolation. To learn more context information, stacked hourglass (encoder-decoder) architecture is used,The stacked hourglass architecture has three main hourglass networks, each of which generates a disparity map. That is, the stacked hourglass architecture has three outputs and losses (Loss 1, Loss 2, and Loss 3). The loss function is described in Section 3.6. During the training phase, the total loss is calculated as the weighted summation of the three losses. During the testing phase, the final disparity map is the last of three outputs



3D CNN (stacked hourglass)		
3Dconv0	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dconv1	$\begin{bmatrix} 3 \times 3 \times 3, 32 \\ 3 \times 3 \times 3, 32 \end{bmatrix}$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack1_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack1_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack2_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack1_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack2_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack3_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack2_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack3_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
output_1	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_2	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_3	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_2**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$

Figura 5: Stacked Hourglass Layer

2.6. Disparity Regression

The predicted disparity $\hat{d}$ is calculated as the sum of each disparity d weighted by its probability $\sigma(-c_d)$, where this probability is computed by applying sigmoid function along the disparity's dimension, and d is a vector with length D containing all possible disparity values.

$$\hat{d} = \sum_{d=0}^{D_{max}} d \times \sigma(-c_d)$$

3 output [output_1, outpu_t2, output_3]		
upsampling	Bilinear interpolation	$D \times H \times W$
	Disparity Regression	$H \times W$

Figura 6: Cost Volume Layer

3. Loss

Since the output is resulted from regression (continuous), $L1$ loss function is used to train the proposed PSMNet. The smooth version of the L1 loss is applied because of its robustness and low sensitivity to outliers, as compared to L2 loss. which is defined as:

$$L(d, \hat{d}) = \frac{1}{N} \sum_{i=0}^{N} smooth_{L_1}(d - \hat{d})$$

Where,

$$smooth_{L_1}(x) = \begin{cases} 0{,}5x^2, & \text{if } |x| \leq 1 \\ |x| - 0{,}5, & \text{otherwise} \end{cases}$$

4. Optimization

Adam optimization algorithm is used with hyperparameters $\beta_1 = 0{,}9$ and $\beta_2 = 0{,}999$.

5. Preprocessing

Color normalization was performed on the entire datase. During training, images were randomly cropped to size $H = 256$ and $W = 512$ for augmentation.

6. Experiments

Since the model has 5224768 parameters in total and contains many $3D$ convoultion operations, the original paper used **four nNvidia Titan-Xp** GPUs to train the model. At First, the model was trained on **Scene Flow** dataset with a constant learning rate of $0{,}001$ for 10 epochs and a batch size of 12, which took about 13 hours. After that the pretrained model is trained on **KITTI 2015** dataset for disparity detection. The training process used 300 epochs and a patch size of 12. With a learning rate of 0.001 for the first 200 epochs and 0.0001 for the remaining 100 epochs. which took about 5 hours.

In my case both time and resources are limited, accordingly, the **Scene Flow**'s pretrained model is used as an initial model. After that the model trained on **KITTI 2015** dataset with a constant learning rate of $0{,}001$ for 15 epochs and a batch size of 2 (due to limited size of the GPU's RAM). Also, the maximum disparity D is set to 80 instead of 192 (as mentioned in the paper), to limit the dimensions of data propagating through the model.

KITTI 2015 dataset contains 200 training stereo image pairs with sparse ground-truth disparities obtained using LiDAR and another 200 testing image pairs without ground-truth disparities. Image size is $H = 376$ and $W = 1240$. Training images are further divided into a training set ($80\,\%$) and a validation set ($20\,\%$).

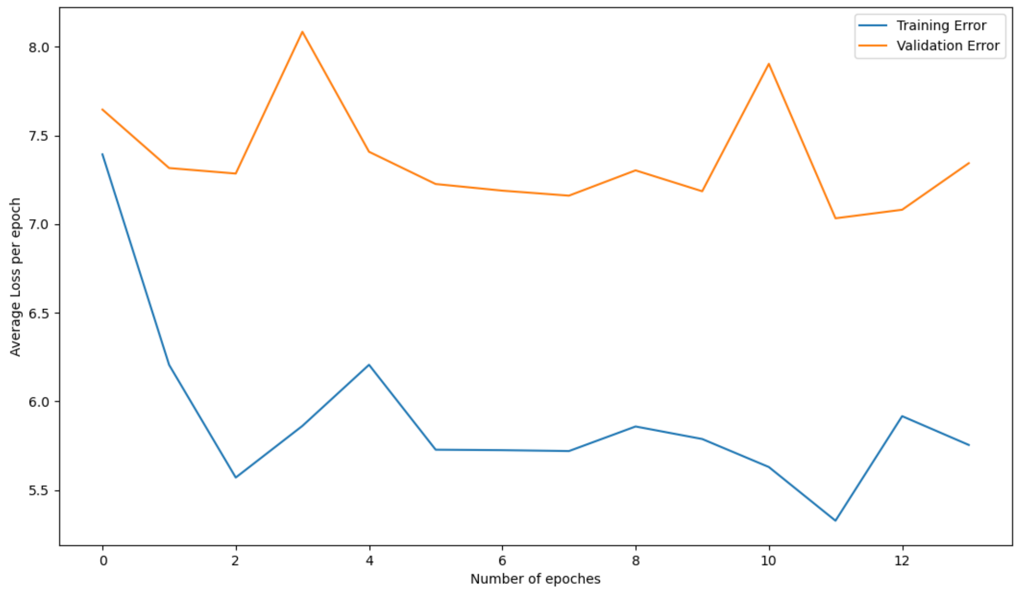


Figura 7: Training and validation errors

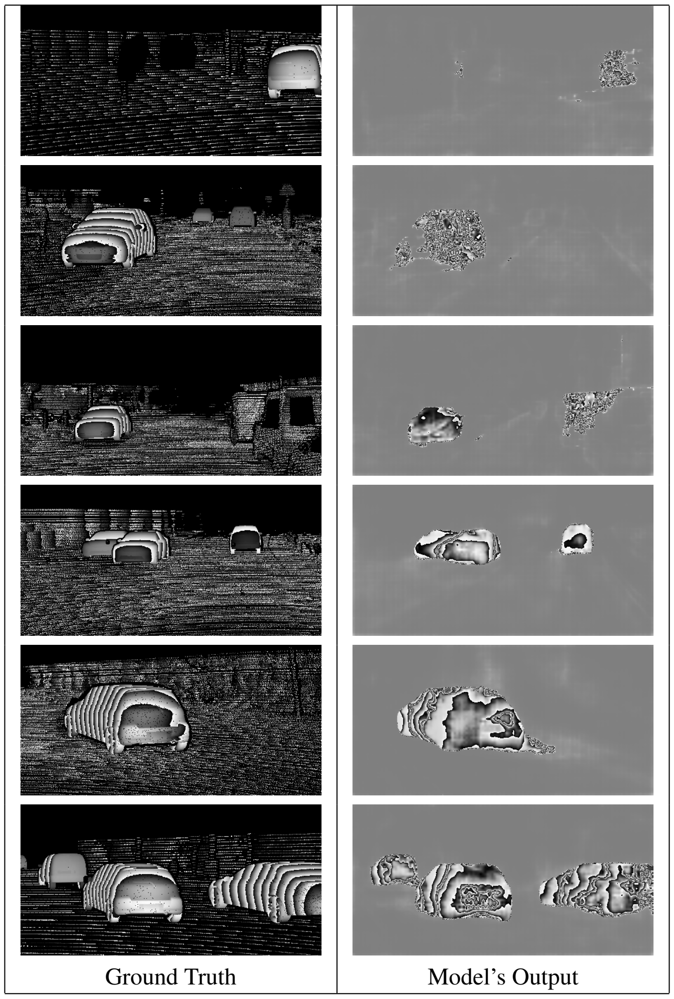


Figura 8: Output's improvement overtime