# A Lightweight Real-time Stereo Depth Estimation Network with Dynamic Upsampling Modules

Yong Deng[1][a], Jimin Xiao[2][b] and Steven Zhiying Zhou[1,3]

[1]*Department of Electrical and Computer Engineering, National University of Singapore, 117583, Singapore*

[2]*Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, 215123, P.R.China*

[3]*National University of Singapore Suzhou Research Institute, Suzhou, Jiangsu, 215123, P.R.China*

Keywords: Stereo Matching, Depth Estimation, Deep Learning, Dynamic Upsampling.

Abstract: Deep learning based stereo matching networks achieve great success in the depth estimation from stereo image pairs. However, current state-of-the-art methods usually are computationally intensive, which prevents them from being applied in real-time scenarios or on mobile platforms with limited computational resources. In order to tackle this shortcoming, we propose a lightweight real-time stereo matching network for disparity estimation. Our network adopts the efficient hierarchical Coarse-To-Fine (CTF) matching scheme, which starts matching from the low-resolution feature maps, and then upsamples and refines the previous disparity stage by stage until the full resolution. We can take the result of any stage as output to trade off accuracy and runtime. We propose an efficient hourglass-shaped feature extractor based on the latest MobileNet V3 to extract multi-resolution feature maps from stereo image pairs. We also propose to replace the traditional upsampling method in the CTF matching scheme with the learning-based dynamic upsampling modules to avoid blurring effects caused by conventional upsampling methods. Our model can process $1242 \times 375$ resolution images with 35-68 FPS on a GeForce GTX 1660 GPU, and outperforms all competitive baselines with comparable runtime on the KITTI 2012/2015 datasets.

## 1 INTRODUCTION

Depth estimation is a fundamental problem in computer vision, with numerous applications including 3D reconstruction (Izadi et al., 2011; Alexiadis et al., 2012), robotics (Schmid et al., 2013; Mancini et al., 2016; Ye et al., 2017; Wang et al., 2017), augmented reality (Alhaija et al., 2018; Zenati and Zerhouni, 2007), etc. Stereo matching is a passive depth estimation method based on stereo triangulation between two rectified images taken from different viewpoints with a slight displacement. By stereo matching, we can obtain the disparity between corresponding pixels in the stereo images pair, which can be further transformed into depth information according to the focal length and the stereo camera's baseline.

Unlike active depth sensors (e.g., time-of-flight cameras, structured light cameras, and LiDAR), stereo matching only relies on dual cameras without the need for a particular illumination component, making it significantly more affordable and energy-efficient. Therefore, stereo depth estimation is especially suitable for mobile platforms with strict power restrictions.

Stereo matching has been studied for decades (Lucas et al., 1981; Hamzah and Ibrahim, 2016), where the algorithms can be classified into local or global approaches in general. Recently, deep convolutional neural networks (CNN) have been adopted in this feild (Mayer et al., 2016; Kendall et al., 2017) and achieve significant progress. Deep neural networks can learn to incorporate the context information and thus better handle the ill-posed regions such as occlusion areas, repeated patterns, and textureless regions.

Despite the remarkable advances, deep neural networks tend to consume large amounts of computational power, leading to significant processing time. Most approaches on the KITTI stereo 2012/2015 leaderboards (Geiger et al., 2012; Menze and Geiger, 2015) cannot achieve real-time processing even though with a high-end GPU. For example, CSPN (Cheng et al., 2019), the current state-of-the-

[a] https://orcid.org/0000-0003-0987-2182
[b] https://orcid.org/0000-0002-9416-2486

Deng, Y., Xiao, J. and Zhou, S.
A Lightweight Real-time Stereo Depth Estimation Network with Dynamic Upsampling Modules.
DOI: 10.5220/0010197607010710
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 701-710
ISBN: 978-989-758-488-6

art stereo matching algorithm, obtains a frame rate of 1FPS on a Titian X GPU, which is too slow for real-time applications like augmented reality.

In this paper, we propose a lightweight real-time stereo matching network for depth estimation. Our network adopts the efficient hierarchical Coarse-To-Fine (CTF) matching scheme (Quam, 1987; Yin et al., 2019), which starts matching from the low-resolution feature maps, and then upsamples and refines the previous results stage by stage until the full resolution. The nature of such hierarchical processing allows us trade-off accuracy and runtime on demand, i.e., we can take the result of any stage as output and cancel the following processing. This is called *anytime computational approach* in (Wang et al., 2019b).

The hierarchical CTF matching scheme is efficient, which results from two reasons. For one thing, it performs correspondence search hierarchically — it first searches for a rough disparity value in the low-resolution stage, and then refine it by searching for a residual disparity within a small neighborhood of previous value in the higher resolution stage. This strategy avoids the time-consuming full range searching. For another, it upsamples the low-resolution result for the initialization in the higher resolution stage, i.e., it propagates the result of a pixel to its neighborhoods. Compared to performing a hierarchical search in full resolution, this strategy further reduces the computational overhead. However, this strategy leads to a drawback — it introduces errors to the disparity boundary in the upsampling process. This is because the high-frequency information is lost in the low-resolution disparity and cannot be recovered by naive upsampling.

To overcome this drawback, we propose to replace the naive upsampling method with the dynamic upsampling modules. The proposed module first generates dynamic upsampling kernels for each pixel in the high-resolution disparity. The dynamic upsampling kernels are inferred from the high-resolution feature map. They are both sample and spatial variant, unlike conventional upsampling kernels. In this way, the high-frequency information can be encoded in the dynamic upsampling kernels and recovered in the high-resolution disparity by the dynamic upsampling process effectively.

For the multi-resolution feature maps extraction, we propose an efficient hourglass-shaped feature extractor MobileNetV3-Up based on the latest MobileNetV3. Compared to original MobileNetV3, our feature extractor aggregates the multi-scale features, allowing the network to exploit multi-scale context information, which is essential for the stereo matching process.

The proposed network performs stereo matching and dynamic upsampling alternately, where the results of any stage can be taken as output (Figure 1). It can process $1242 \times 375$ resolution image with a frame rate range from 35 to 68 FPS on a mid-end GeForce GTX 1660 GPU, depending on which output is finally adopted. We refer our network as LiteStereo since it is designed to be lightweight. We evaluate LiteStereo on multiple stereo benchmark datasets. The results show that it outperforms all competitive baselines with comparable runtime.

## 2 RELATED WORKS

**Stereo Matching.** Stereo matching, or depth from stereo, is a long-standing computer vision task that has been studied for decades (Barnard and Fischler, 1982). Detailed surveys can be found in (Scharstein and Szeliski, 2002; Hamzah and Ibrahim, 2016). A stereo matching pipeline typically consists of four steps: (1) matching costs volume computation, (2) cost volume aggregation, (3) disparity estimation, and (4) optional disparity refinement (Scharstein and Szeliski, 2002; Hamzah and Ibrahim, 2016). Recently deep convolutional neural networks have been adopted for stereo matching and achieve great success, where most successful network designs also follow the classical pipeline (Kendall et al., 2017; Chang and Chen, 2018; Khamis et al., 2018; Yin et al., 2019). Hierarchical Coarse-To-Fine (CTF) matching is an essential strategy in stereo matching (Quam, 1987), since it reduces both computational complexity and matching ambiguity. HD3 (Yin et al., 2019) proposes a stereo network following this strategy and achieve state-of-the-art performance. MADNet (Tonioni et al., 2019) proposes a real-time self-adaptive network which can perform online adaptation in real-time. Our work also adopts the hierarchical CTF matching strategy to achieve real-time processing.

**Efficient Backbone Networks.** Like the networks for many other tasks, such as image classification (He et al., 2016), object detection (Lin et al., 2017) and pose estimation (Sun et al., 2019), stereo matching networks also need a backbone network for feature extraction. Efficient backbone networks have been an active research area in recent years. MobileNet (Howard et al., 2017) improves computation efficiency substantially by introducing depth-wise separable convolution. The following work MobileNet V2 (Sandler et al., 2018) employs a resource-efficient block with inverted residuals and linear bottlenecks. MobileNet V3 (Howard et al., 2019) uses a combination of these layers as building blocks

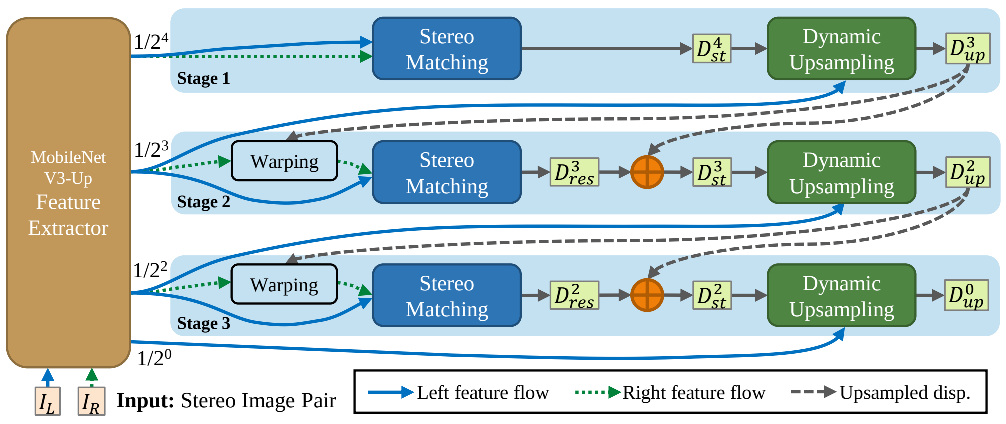


Figure 1: Network architecture of LiteStereo, which consists of a pyramid feature extractor and three stages for stereo matching and dynamic upsampling. $D^l$ denotes the disparity map with the scale of $1/2^l$. The dynamic upsampling ratio is 2 for Stage 1 & 2, and 4 for Stage 3. See text for details.

and exploits network architecture search algorithms for network design. Apart from MobileNet family, there are other efficient backbone networks like SqueezeNet (Iandola et al., 2016), ShuffleNet (Zhang et al., 2018), ShiftNet (Wu et al., 2018), etc.

**Depth Image Upsampling.** As pointed out above, we need a more elaborate upsampling method to recover the high-frequency information in the upsampled disparity so as to avoid the edge blurring effect. There are many works on depth image upsampling (Eichhardt et al., 2017). Joint upsampling approaches (Li et al., 2016; Hui et al., 2016) use feature maps as guidance by merely concatenating the feature maps of the color image and the depth image. PAC (Su et al., 2019) predicts spatially varying kernels from the guidance and applies them to the feature maps of depth image for upsampling. Our dynamic upsampling module is more concise and closely integrated with the hierarchical CTF framework.

## 3 METHODOLOGY

The architecture overview of the proposed LiteStereo is shown in Figure 1. The network takes a stereo image pair $I_L, I_R$ as input, and output six disparity maps $D^4_{st}, D^3_{up}, D^3_{st}, D^2_{up}, D^2_{st}, D^0_{up}$ with different accuracy successively, where the superscript of $D^l$ denotes that the resolution is $1/2^l$ of the full one, and $st, up$ denote that the disparity is produced by **st**ereo matching module and dynamic **up**sampling module respectively.

For each input image, the MobileNetV3-Up feature extractor computes a feature pyramid that consists of feature maps of different scales (1/16, 1/8, 1/4, 1). For a better trade-off between accuracy and runtime, all computation is performed on demand. For example, when we start with the stereo matching module in Stage 1, only the features with the scale of $1/2^4$ are computed. This stereo matching module produces a coarse disparity map $D^4_{st}$ as the first output of the network. If time is permitted, we continue with the dynamic upsampling module in Stage 1. At this time, the feature computation in MobileNetV3-Up resumes from where it has stopped and outputs the left image feature with a scale of $1/2^3$. The dynamic upsampling module increases the resolution of $D^4_{st}$ and produces an upsampled disparity map $D^3_{up}$ with higher resolution and accuracy.

Stage 2 follows a similar process as Stage 1, except that it uses the disparity $D^3_{up}$ from the previous stage as initialization, which is achieved by the warping operation. The output of the stereo matching module in Stage 2 is a residual disparity $D^3_{res}$, which is added to the initial disparity $D^3_{up}$ to obtain the whole disparity $D^3_{st}$. Stage 3 follows the same process, in which the stereo disparity map $D^2_{st}$ is upsampled to full resolution $D^0_{up}$ via the dynamic upsampling module with an upsampling ratio of 4.

In the rest of this section, we will introduce the details of the feature extractor, stereo matching module, and dynamic upsampling module.

### 3.1 Feature Extractor

In order to keep the network lightweight and efficient, we adopt the latest MobileNetV3 (Howard et al., 2019) as backbone for feature extraction. However, the original MobileNetV3 is not suitable for the stereo matching task. Since stereo matching is a pixel-to-pixel task, high spatial resolution feature maps are required for matching cost evaluation. However, the high-resolution features in MobileNetV3 are in shallow layers, which means their receptive fields are small and lack semantic information. Therefore, inspired by the U-Net (Ronneberger et al., 2015), we add an expansion part to MobileNetV3 to aggregate the low-scale feature with the high-scale one, so as to exploit the context information from a larger receptive field and obtain more semantic meaning. We use a single $3 \times 3$ 2D convolution layer for feature aggregation. Thus, the increased computation overhead is slight. The detailed network architecture can be found in Table 1, where Operator 1-6 are the same as in MobileNetV3-Small (Howard et al., 2019).

## 3.2 Stereo Matching Module

The architecture of the stereo matching module is illustrated in Figure 2. The stereo matching module takes as input the left and (warped) right feature maps in order to compute a disparity map. Note that the right feature maps for Stage 2 & 3 are warped according to the disparity of previous stage:

$$F^l_{R,wp}(x,y) = F^l_R(x + D^l_{init}(x,y),y), \qquad (1)$$

where $F^l_{R,wp}$ denotes the wraped feature map, $F^l_R$ denotes the right feature map, $D^l_{init}$ denotes the disparity map for initialization, $x,y$ denote the horizontal and vertical coordinates on the 2D image plane, the superscript $l$ denotes the scale $1/2^l$. The right feature map for Stage 1 does not need to be warped since no previous disparity is available. This is equivalent to warping with an all-zero disparity map, i.e., $F^l_{R,wp} = F^l_R$.

The stereo matching consists of three steps:

**1) Cost Volume Computation.** Given the left $F^l_L$ and warped right feature maps $F^l_{R,wp}$, the module first computes a preliminary cost volume $C^l_{pre}$:

$$C^l_{pre}(c,d,x,y) = F^l_L(c,x,y) - F^l_{R,wp}(c,x+d,y), \quad (2)$$

where $c$ denotes the index of feature channels, $d$ denotes the disparity, $x,y$ denote the horizontal and vertical coordinates on the 2D image plane.

The resulting cost volume is a 4D volume with size $C \times D \times H \times W$, where $C$ denotes the number of feature channels of the feature map, $D$ denotes the number of disparities under consideration, $H \times W$ is the size of feature maps. The $C^l_{pre}(:,d,x,y)$ entry is a distance vector that describes the matching cost between the two pixels $F^l_L(x,y)$ and $F^l_{R,wp}(x+d,y)$.

The search range (the disparities under consideration) ranges from 0 to 11 for Stage 1, and from -2 to 2 for Stage 2 & 3. Note that the search range in a low-scale feature map is equivalent to $2^l$ times of it in the full resolution feature map. For example, the search range $\pm 2$ for Stage 2 / 3 is equivalent to $\pm 16 / \pm 8$ pixel in the full resolution.

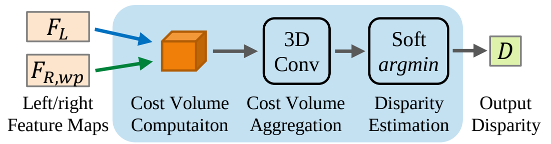


Figure 2: Stereo matching module that performs stereo matching between the left feature map and the (warped) right feature map. See text for details.

**2) Cost Volume Aggregation.** The preliminary cost volume usually is noisy due to the matching ambiguity, occlusion, or blurring in the input images. To reduce the noise, a cost volume aggregation step is often applied (Hamzah and Ibrahim, 2016). We implement the cost volume aggregation with 3D convolutional layers (Chang and Chen, 2018). We expect the 3D CNN learns to locally aggregate the cost by exploiting the context information, and produces a 3D cost volume with the size of $D \times H \times W$. The details of 3D CNN can be found in Table 1.

**3) Disparity Estimation.** Given the estimated 3D cost volume $C^l$, a naive way to estimate the disparity map would be the winner-take-all (WTA) strategy, where the disparity with the lowest cost would be chosen as the output:

$$D(x,y) = \arg\min_d C^l(d,x,y). \qquad (3)$$

However, the WTA strategy cannot provide disparity with sub-pixel accuracy. Moreover, it blocks most of the backward propagation path during network training due to the non-differentiable *argmin* operation. Therefore, we adopt the soft *argmin* for disparity estimation as suggested by (Kendall et al., 2017):

$$D^l_{res}(x,y) = \sum_d d \cdot \frac{exp(-C^l(d,x,y))}{\sum_{d'} exp(-C^l(d',x,y))}. \qquad (4)$$

The estimated disparity residual $D^l_{res}$ is added to the initial disparity $D^l_{init}$ to obtain whole disparity $D^l_{st}$. Again, since there is no initial disparity for Stage 1, we have $D^4_{res} = D^4_{st}$ at Stage 1.

## 3.3 Dynamic Upsampling Module

The proposed dynamic upsampling module is inspired by (Jia et al., 2016; Wang et al., 2019a). The dynamic upsampling process is demonstrated in Figure 3. Each pixel in upsampled disparity is calculated as the weighted sum of the supported window in low-resolution disparity centered at the reference pixel, where the weights are defined by the predicted dynamic kernel. In order to achieve minimal computational overhead, we use a $2 \times 2$ kernel size for dynamic upsampling, which is similar to bilinear interpolation, except that the kernel weights are generated by 2D convolutional layers. The key insight of our dynamic upsampling module is that we predict dynamic upsampling kernels from the high-resolution feature map. The predicted kernels are both sample and spatial variant, preserving the high-frequency information. With the predicted kernels, the finer details of the disparity map can be recovered in the dynamic upsampling process. More specifically, the predicted dynamic kernel matrix is a 4-channel feature map with the same resolution of upsampled disparity. The kernel weights for each pixel are normalized with softmax. The module detail can be found in Table 1. If computational overhead is permitted, the kernel size can be easily changed to a large size. For example, we can use a $3 \times 3$ kernel size, and the predicted dynamic kernel matrix should be a 9-channel feature map. The upsamping scale factor is 2 for Stage 1 & 2, and 4 for Stage 3.

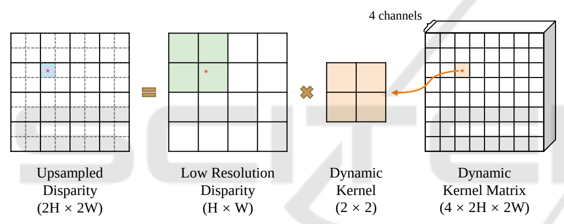

4 channels
Upsampled Disparity (2H × 2W)
Low Resolution Disparity (H × W)
Dynamic Kernel (2 × 2)
Dynamic Kernel Matrix (4 × 2H × 2W)

Figure 3: Dynamic upsampling process with a scale factor of 2. Each pixel in upsampled disparity is calculated as the weighted sum of the supported window in low resolution disparity centered at the reference pixel, where the weights are defined by the predicted dynamic kernel matrix.

## 3.4 Loss Function

The network outputs the results of Operator {28, 34, 30, 36, 32, 37} successively, which correspond to $\{D^4_{st}, D^3_{up}, D^3_{st}, D^2_{up}, D^2_{st}, D^0_{up}\}$. We upsample all outputs to full resolution with bilinear interpolation, and compute the loss for each output disparity map:

$$\mathcal{L}(d, \hat{d}) = \frac{1}{N}\sum_{i=1}^{N} smooth_{L1}(d_i - \hat{d}_i), \tag{5}$$

where $d$ denotes the ground truth disparity, and $\hat{d}$ denotes the predicted disparity, $N$ denotes the number of labeled pixels, $smooth_{L1}$ denotes the smooth L1 loss function (Girshick, 2015). The losses for different outputs are weighted differently, with weights of 0.25, 0.5, 1 for Stage 1, 2, 3 respectively.

Table 1: Network architecture of LiteStereo. $s_{in}$,$s_{out}$ denote the scale of input and output. $c_{in}$,$c_{out}$ denote the number of channels of input and output. $(\cdot,\cdot)$ denotes concatenation of two inputs. $\cdot[:k]$ denotes taking the first $k$ channels as input. '2x' and '4x' before the upsampling method denotes the upsampling scale. 'conv3d x4' denotes the layer replicating four times with independent weights. The bold number indicates the incoming skip link from nonsequential layers.

| # | Input | $s_{in}$ | $c_{in}$ | Operator | $s_{out}$ | $c_{out}$ |
|---|---|---|---|---|---|---|
| **MobileNetV3-Up Fearute Extractor** | | | | | | |
| 1 | Image | 1 | 3 | conv2d, 3x3, stride 2 | 1/2 | 16 |
| 2 | 1 | 1/2 | 16 | bneck, 3x3, stride 2 | $1/2^2$ | 16 |
| 3 | 2 | $1/2^2$ | 16 | bneck, 3x3, stride 2 | $1/2^3$ | 24 |
| 4 | 3 | $1/2^3$ | 24 | bneck, 3x3, stride 1 | $1/2^3$ | 24 |
| 5 | 4 | $1/2^3$ | 24 | bneck, 3x3, stride 2 | $1/2^4$ | 40 |
| 6 | 5 | $1/2^4$ | 40 | bneck, 3x3, stride 1 | $1/2^4$ | 40 |
| 7 | 6 | $1/2^4$ | 40 | 2x bilinear upsample | $1/2^3$ | 40 |
| 8 | (**4**,7) | $1/2^3$ | 64 | conv2d, 3x3, stride 1 | $1/2^3$ | 24 |
| 9 | 8 | $1/2^3$ | 24 | 2x bilinear upsample | $1/2^2$ | 24 |
| 10 | (**2**,9) | $1/2^2$ | 40 | conv2d, 3x3, stride 1 | $1/2^2$ | 16 |
| 11 | 10[:4] | $1/2^2$ | 4 | 2x bilinear upsample | 1/2 | 4 |
| 12 | (**1[:4]**,11) | 1/2 | 8 | conv2d, 3x3, stride 1 | 1/2 | 4 |
| 13 | 12 | 1/2 | 4 | 2x bilinear upsample | 1 | 4 |
| 14 | **Image** | 1 | 3 | conv2d, 3x3, stride 1 | 1 | 4 |
| 15 | (**13**,14) | 1 | 8 | conv2d, 3x3, stride 1 | 1 | 4 |
| **Cost Volume Computation** | | | | | | |
| 16 | **6** | $1/2^4$ | 40 | build cost vol. | $1/2^4$ | 40 |
| 17 | **34 & 8** | $1/2^3$ | 24 | warp, build cost vol. | $1/2^3$ | 24 |
| 18 | **36 & 10** | $1/2^2$ | 16 | warp, build cost vol. | $1/2^2$ | 16 |
| **Cost Volume Aggregation** | | | | | | |
| 19 | **16** | $1/2^4$ | 40 | conv3d, 3x3x3 | $1/2^4$ | 16 |
| 20 | 19 | $1/2^4$ | 16 | conv3d x4, 3x3x3 | $1/2^4$ | 16 |
| 21 | 20 | $1/2^4$ | 16 | conv3d, 3x3x3 | $1/2^4$ | 1 |
| 22 | **17** | $1/2^3$ | 24 | conv3d, 3x3x3 | $1/2^3$ | 4 |
| 23 | 22 | $1/2^3$ | 4 | conv3d x4, 3x3x3 | $1/2^3$ | 4 |
| 24 | 23 | $1/2^3$ | 4 | conv3d, 3x3x3 | $1/2^3$ | 1 |
| 25 | **18** | $1/2^2$ | 16 | conv3d, 3x3x3 | $1/2^2$ | 4 |
| 26 | 25 | $1/2^2$ | 4 | conv3d x4, 3x3x3 | $1/2^2$ | 4 |
| 27 | 26 | $1/2^2$ | 4 | conv3d, 3x3x3 | $1/2^2$ | 1 |
| **Disparity Estimation** | | | | | | |
| 28 | **21** | $1/2^4$ | 12 | soft argmin | $1/2^4$ | 1 |
| 29 | **24** | $1/2^3$ | 5 | soft argmin | $1/2^3$ | 1 |
| 30 | 29 & **34** | $1/2^3$ | 5 | sum | $1/2^3$ | 1 |
| 31 | **27** | $1/2^2$ | 5 | soft argmin | $1/2^2$ | 1 |
| 32 | 31 & **36** | $1/2^2$ | 5 | sum | $1/2^2$ | 1 |
| **Dynamic Upsampling** | | | | | | |
| 33 | **8[:12]** | $1/2^3$ | 12 | conv2d, 3x3, stride 1 | $1/2^3$ | 4 |
| 34 | 33 & **28** | $1/2^4$ | 1 | 2x dynamic upsamp. | $1/2^3$ | 1 |
| 35 | 10[:8] | $1/2^2$ | 8 | conv2d, 3x3, stride 1 | $1/2^2$ | 4 |
| 36 | 35 & **30** | $1/2^3$ | 1 | 2x dynamic upsamp. | $1/2^2$ | 1 |
| 37 | **15 & 32** | $1/2^2$ | 1 | 4x dynamic upsamp. | 1 | 1 |

## 4 EXPERIMENTS

In this section, we evaluate our method on different datasets and compare it with existing stereo algorithms on accuracy and runtime and show that we can achieve high-quality results and high frame rate. In addition, we conduct ablation studies to demonstrate the effectiveness of our network designs.

### 4.1 Experiment Details

#### 4.1.1 Datasets

We trained and evaluated our method on three stereo datasets:
**1) Scene Flow (Mayer et al., 2016):** a large synthetic dataset containing 35454 training and 4370 testing stereo image pairs, where the size of the image is $960 \times 540$ pixels, and the provided ground truth disparity maps are dense.
**2) KITTI 2012 (Geiger et al., 2012):** a real-world dataset containing 194 training and 195 testing stereo image pairs, where the size of image is $1242 \times 375$ pixels, and the provided ground truth disparity maps are sparse.
**3) KITTI 2015 (Menze and Geiger, 2015):** a real-world dataset containing 200 training and 200 testing stereo image pairs, where the size of the image is $1242 \times 375$ pixels, and the provided ground truth disparity maps are sparse.

#### 4.1.2 Training Details

We implement the proposed network LiteStereo with PyTorch, where the detailed network architecture is shown in Table 1. Our model is trained end-to-end using Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with a batch size of 6. Color normalization is applied to the entire dataset for data preprocessing. As for training set data augmentation, we randomly crop the image to size H = 256 and W = 512.

Since the two KITTI datasets are too small for training, we first train our model on the Scene Flow dataset and then fine-tune it on the two KITTI datasets respectively before evaluating on them. Before training, the weights of the front part of the MobileNetV3-Up feature extractor (Operator 1-6 in Table 1) are initialized from the ImageNet pretrained MobileNetV3-Small model (Howard et al., 2019) and then frozen.

On the Scene Flow dataset, the model is trained for 10 epochs in total with a constant learning rate of $5 \times 10^{-4}$ . The frozen weights are unfrozen after one training epoch. For the KITTI datasets, we fine-tune the model pretrained on the SceneFlow dataset for 300 epochs with an initial learning rate of $5 \times 10^{-4}$. The learning rate is reduced to $5 \times 10^{-5}$ after the 200th epoch. The results on KITTI datasets are averaged over five randomized 80/20 train/validation splits, which follows the evaluation protocol in (Wang et al., 2019b).

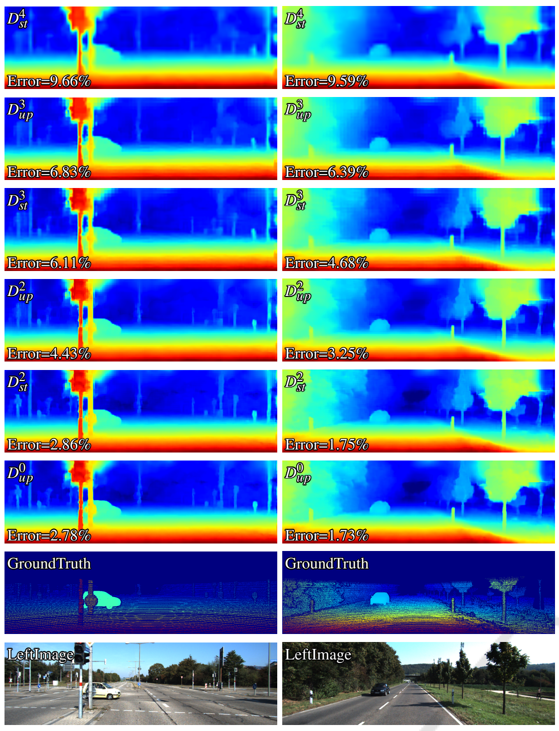
#### 4.1.3 Baseline Comparison

We compare our method with other four real-time stereo matching methods: StereoNet (Khamis et al., 2018), AnyNet (Wang et al., 2019b), MADNet (Tonioni et al., 2019), and DispNet (Mayer et al., 2016), where the comparison focuses on both disparity accuracy and inference time. We compare different methods only on KITTI 2012 & 2015 datasets, since some methods did not report their results on SceneFlow, or the evaluation protocols are different from each other. For a fair comparison, we perform inference with each network on the same computer with a GeForce GTX Titan X GPU to estimate the average runtime, where the input is a stereo image pair with the resolution of $1242 \times 375$. Note that the GeForce GTX Titan X with Maxwell$^{TM}$ architecture we used is significantly inferior to the NVIDIA TITAN X with Pascal$^{TM}$ architecture, although they have very similar names. As for the disparity accuracy, we adopt the performance results reported in the original papers.

### 4.2 Experiment Results

Here, we first show the qualitative and quantitative results of our LiteStereo on different datasets and then compare our method with other baselines.

The qualitative results on KITTI 2015 can be found in Figure 4. The percentage of outliers is indicated in the figure. We only count outliers if the disparity or flow exceeds 3 pixels *and* 5% of its true value, which is consistent with KITTI 2015 paper (Menze and Geiger, 2015). Since the prediction is refined step by step, as more inference time is given, more accurate results we get. Different trade-offs between accuracy and runtime can be achieved on demand using one model. The quantitative results on KITTI 2012 & 2015 and SceneFlow can be found in Table 2. The outlier rate is used for KITTI, and End-Point-Error (EPE) is used for SceneFlow. We can see that the dynamic upsampling module can efficiently improve the accuracy with a small computational overhead. The improvement of the dynamic upsampling module of the last stage is still significant on the SceneFlow dataset but not on KITTI datasets. A reasonable explanation is that the ground truth of KITTI datasets lacks valid pixels on the disparity discontinuity due to its sparsity. Thus the dy-



(a) (b)

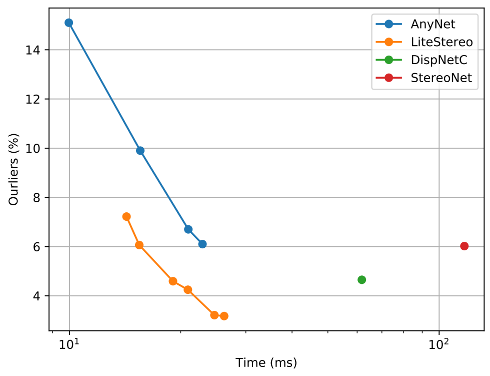
Figure 4: Qualitative results on KITTI2015. The notations of six outputs correspond to those in Figure 1. The prediction is refined step by step. Different trade-offs between accuracy and runtime can be achieved on demand in one model. Error denotes the percentage of outliers. Zoom in to see the details.

Table 2: Runtime and outlier(%) of LiteStereo on KITTI-2012 / KITTI-2015 datasets and EPE on SceneFlow. Lower values are better. Runtime is measured on KITTI dataset. “Incr.” denotes the increased time since last output, “Acc.” denotes the accumulated time from beginning.

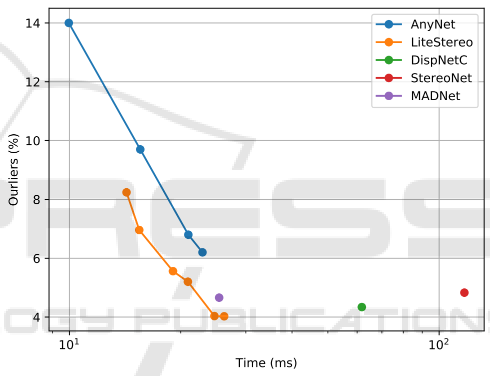
| | Time (ms) | | Outliers (%) | | EPE (px) |
|---|---|---|---|---|---|
| Output | Incr. | Acc. | 2012 | 2015 | SceneFlow |
| 1. $D^4_{st}$ | 14.27 | 14.27 | 7.22 | 8.24 | 3.49 |
| 2. $D^3_{up}$ | 1.18 | 15.45 | 6.06 | 6.96 | 2.80 |
| 3. $D^3_{st}$ | 3.61 | 19.06 | 4.59 | 5.56 | 2.51 |
| 4. $D^2_{up}$ | 1.86 | 20.91 | 4.25 | 5.20 | 2.18 |
| 5. $D^2_{st}$ | 3.75 | 24.67 | 3.21 | 4.03 | 1.95 |
| 6. $D^0_{up}$ | 1.55 | 26.21 | 3.18 | 4.03 | 1.74 |

namic upsampling kernel CNN fails to learn reasonable weights for the upsampling kernel prediction to produce an accurate disparity boundary.

The comparison with other baseline is demonstrated in Figure 5. The outlier rate is used as the metric. Our method achieves a better accuracy-runtime trade-off than all competitive real-time baselines. We can achieve lower error rates within less runtime. LiteStereo does not rely on any customized operator or CUDA C/C++ programming, making it easy to be deployed on other platforms such as mobile phones.



(a) Comparisons on KITTI 2012 dataset.



(b) Comparisons on KITTI 2015 dataset.

Figure 5: Comparisons of different baselines on KITTI datasets. The outlier rate is used as the metric. The time axis is logarithmic axes.

## 4.3 Ablation Studies

We conduct ablation studies to examine the impact of different components of the LiteStereo network. We evaluate different variants of our model on the SceneFlow dataset.

### 4.3.1 Feature Extractor

As described in Section 3.1, we add an expansion part to MobileNetV3 to aggregate the multi-scale features. In the first ablation study, we remove the expansion part and directly use MobileNetV3 (Operator 1-6 in Table 1) as the feature extractor. We compare the performance of MobileNetV3 and MobileNetV3-Up. To avoid being disturbed by dynamic upsampling module, we use a bilinear upsampler in this ablation study.

Table 3: EPE of LiteStereo with different settings evaluated on SceneFlow. The number in the parentheses denotes the reduction of EPE w.r.t. last output. "FeatExt" denotes Feature Extractor, "feat_gui" denotes feature guided joint upsampling, "dyn_up" denotes dynamic upsampling.

| FeatExt | MobileNetV3 | MobileNetV3-Up | | |
|---|---|---|---|---|
| Upsampler | bilinear | bilinear | feat_gui | dyn_up |
| 1. $D_{st}^4$ | **3.46** | 3.56 | 3.51 | 3.49 |
| 2. $D_{up}^3$ | - | - | 3.38 | **2.80** |
| 3. $D_{st}^3$ | 2.84 (-0.62) | 2.86 (**-0.70**) | 2.84 | **2.51** |
| 4. $D_{up}^2$ | - | - | 2.80 | **2.18** |
| 5. $D_{st}^2$ | 2.52 (-0.32) | 2.43 (**-0.43**) | 2.37 | **1.95** |
| 6. $D_{up}^0$ | - | - | 2.41 | **1.74** |

The results is reported in Table 3. As shown in the table, MobileNetV3 results in higher error than MobileNetV3-Up at high-resolution output. This is because the high-resolution features of MobileNetV3 are in shallow layers and unable to aggregate enough context information. The resulting feature vectors for the pixels do not contain enough information to be distinguished from each other, which leads to ambiguity in the stereo matching process. The feature maps for the stereo matching module in Stage 1 are produced in the same layer in both MobileNetV3 and MobileNetV3-Up, which are generated by the Operator 6 in Table 1. Thus, there is no deterioration in the first output $D_{st}^4$ even if the MobileNetV3 is used.

#### 4.3.2 Dynamic Upsampling Module

In order to demonstrate the effectiveness of the dynamic upsampling module, we compare it with traditional bilinear upsampler and a feature guided joint upsampling method. The guided joint upsampling module first upsample the disparity and concatenate it with a feature. Then, a 2D convolutional layer is applied to it for disparity refinement. We design the guided joint upsampling module with a similar computational overhead as the dynamic upsampling module. MobileNetV3-Up is used as feature extractor.

The results are reported in Table 3. As shown in the table, the feature guided joint upsampling only achieves a slightly smaller error (2.37) than traditional bilinear (2.43) at the output $D_{st}^2$, while our dynamic upsampling achieve significantly lower errors than feature guided upsampling at all outputs. We conclude that under such strict computational limitations, dynamic upsampling is better than feature guided upsampling.

## 5 CONCLUSIONS

In this paper, we have proposed a lightweight efficient stereo matching network for disparity estimation in real-time applications. Our network adopts the efficient hierarchical Coarse-To-Fine (CTF) matching scheme. We can take the result of any stage as output to achieve different trade-offs between accuracy and runtime on demand in one model. We propose an efficient hourglass-shaped feature extractor based on the latest MobileNetV3, which is able to aggregate more context information from different scales. We also propose to replace the traditional upsampling method in the CTF matching scheme with the learning-based dynamic upsampling modules, which improves the accuracy significantly with little extra overhead. In the future, we are going to implement our network on the mobile phone for further downstream applications.

## ACKNOWLEDGEMENTS

Steven Zhiying Zhou was supported by National Key Research and Development Program of China under 2018YFB1004904; Science and Technology Program of Suzhou City under SYG201920. Jimin Xiao was supported by National Natural Science Foundation of China under 61972323; Key Program Special Fund in XJTLU under KSF-T-02, KSF-P-02. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

## REFERENCES

Alexiadis, D. S., Zarpalas, D., and Daras, P. (2012). Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358.

Alhaija, H. A., Mustikovela, S. K., Mescheder, L., Geiger, A., and Rother, C. (2018). Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9):961–972.

Barnard, S. T. and Fischler, M. A. (1982). Computational stereo. *ACM Computing Surveys (CSUR)*, 14(4):553–572.

Chang, J.-R. and Chen, Y.-S. (2018). Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418.

Cheng, X., Wang, P., and Yang, R. (2019). Learning depth with convolutional spatial propagation network. *IEEE* *transactions on pattern analysis and machine intelligence*.

Eichhardt, I., Chetverikov, D., and Janko, Z. (2017). Image-guided tof depth upsampling: a survey. *Machine Vision and Applications*, 28(3-4):267–282.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Hamzah, R. A. and Ibrahim, H. (2016). Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hui, T.-W., Loy, C. C., and Tang, X. (2016). Depth map super-resolution by deep multi-scale guidance. In *European conference on computer vision*, pages 353–369. Springer.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568.

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. (2016). Dynamic filter networks. In *Advances in neural information processing systems*, pages 667–675.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., and Bry, A. (2017). End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75.

Khamis, S., Fanello, S., Rhemann, C., Kowdle, A., Valentin, J., and Izadi, S. (2018). Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Y., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2016). Deep joint image filtering. In *European Conference on Computer Vision*, pages 154–169. Springer.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision.

Mancini, M., Costante, G., Valigi, P., and Ciarfuglia, T. A. (2016). Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4296–4303. IEEE.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048.

Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070.

Quam, L. H. (1987). Hierarchical warp stereo. In *Readings in computer vision*, pages 80–86. Elsevier.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Schmid, K., Tomic, T., Ruess, F., Hirschmüller, H., and Suppa, M. (2013). Stereo vision based indoor/outdoor navigation for flying robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3955–3962. IEEE.

Su, H., Jampani, V., Sun, D., Gallo, O., Learned-Miller, E., and Kautz, J. (2019). Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11166–11175.

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE conference on* *computer vision and pattern recognition*, pages 5693–5703.

Tonioni, A., Tosi, F., Poggi, M., Mattoccia, S., and Stefano, L. D. (2019). Real-time self-adaptive deep stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–204.

Wang, C., Meng, L., She, S., Mitchell, I. M., Li, T., Tung, F., Wan, W., Meng, M. Q.-H., and de Silva, C. W. (2017). Autonomous mobile robot navigation in uneven and unstructured indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 109–116. IEEE.

Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C. C., and Lin, D. (2019a). Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3007–3016.

Wang, Y., Lai, Z., Huang, G., Wang, B. H., Van Der Maaten, L., Campbell, M., and Weinberger, K. Q. (2019b). Anytime stereo image depth estimation on mobile devices. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5893–5900. IEEE.

Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., and Keutzer, K. (2018). Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135.

Ye, M., Johns, E., Handa, A., Zhang, L., Pratt, P., and Yang, G.-Z. (2017). Self-supervised siamese learning on stereo image pairs for depth estimation in robotic surgery. *arXiv preprint arXiv:1705.08260*.

Yin, Z., Darrell, T., and Yu, F. (2019). Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6044–6053.

Zenati, N. and Zerhouni, N. (2007). Dense stereo matching with application to augmented reality. In *2007 IEEE International Conference on Signal Processing and Communications*, pages 1503–1506. IEEE.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856.