Reinforcing Multi-Scale Analysis For Depth Estimation

Vasilis Gkolemis, Anastasios Delopoulos

Aristotle University of Thessaloniki

Abstract. Convolutional neural networks exhibit exceptional performance in predicting depth from stereo images. However, this performance comes with two essential drawbacks (a) they consume extraordinary computational power (clusters of GPUs) even for a single prediction and (b) their memory and computational demand are predefined from the training phase, hence they cannot be adjusted to the available resources on-demand. For confronting these problems, we propose a scalable CNN architecture (MSNet), adjustable to the specific requirements of each application; it can reduce its computational demands by sacrificing some precision or target for high accuracy if more resources are available. The bias towards accuracy or efficiency can be determined at test time, without any need for retraining. For achieving such scalability, we adopted the basic ideas of scale-space theory and incorporated them into the MSNet architecture. MSNet exhibits challenging performance comparing to the state-of-the-art methods in the SceneFlow dataset, even though it uses considerably less learnable parameters.

Keywords: Depth Estimation, Stereo Vision, Deep Learning, Multi-Scale Processing

1 Introduction

Stereo vision forms a particular case of the general 3D reconstruction problem. Stereo cameras share the same orientation while their center is displaced horizontally by a distance B. The following relation expresses stereo vision's 3D geometry:

$$z = f\frac{B}{d} \tag{1}$$

where z is the distance from the camera level (depth), f is the focal length and d is the disparity. Defining the stereo pair as $X = (X^L, X^R)$, the stereo problem demotes in finding all correspondences $X^L(x, y) \leftrightarrow X^R(x - d, y) \forall (x, y)$ which, is a patch-matching procedure. Thus, disparity estimation is based on the hypothesis that the local context of correspondent points is similar. If we define as $P^{\{L|R\}}_{n\times n}[x, y]$ the $n \times n$ square patch centered at $X^{\{L|R\}}[x, y]$, $g(\cdot)$ a metric of patch similarity, and $Y[x, y]$ the ground truth disparities, then the hypothesis is:

$$g(P^L_{n\times n}[x,y], P^R_{n\times n}[x-d^*,y]) > g(P^L_{n\times n}[x,y], P^R_{n\times n}[x-d,y])$$
$$\forall d \in [0,D] : d \neq d^*, \text{where } d* = Y[x,y] \tag{2}$$

For hypothesis 2 to hold, we must choose thoughtfully two critical parameters that depend on the visual properties of the reference point: (a) the size of the surrounding area that will be incorporated and (b) the scale of the comparison. Multiscale analysis provides an elegant framework for handling both issues. Furthermore, it offers the mechanism for designing a scalable CNN model.

We define as $X^{\{L|R\}(k,q)}$ the image obtained from $X^{\{L|R\}} \equiv X^{\{L|R\}(k_0=1,q_0=1)}$ through the typical downscaling process:

$$X^{\{L|R\}(k_0=1,q_0=1)} \xrightarrow{downscaling} X^{\{L|R\}(k,q_0=1)} \xrightarrow{downsampling} X^{\{L|R\}(k,q)}$$
$$k \geq 1, q \geq 1 \tag{3}$$

The paramters $k \geq 1$ and $q \geq 1$ are the downscaling and downsampling rate. The downscaling part is performed with an appropriate low-pass filter (e.g. Gaussian kernel) and the downsampling through an interpolation method. Based on this formulation, we define a patch on a downscaled image as:

$$P^{\{L|R\}(k,q)}_{n\times n}[x,y] = X^{\{L|R\}(k,q)}[x-n : x+n, y-n : y+n] \tag{4}$$

For each reference point $[x,y]$ there is a different combination of scale k and patch size $n \times n$ that leads to a succesfull matching procedure. For example, regions without texture require large patch size in order to incorporate features from neighbour objects. On the other hand, for small foreground objects, a small patch is suitable, since, in this case, background objects add noise to the comparison. This variability requires the execution of the patch matching process for various combinations of scales and patch-sizes. For keeping the complexity low, we restrict the search space in a single dimension, binding k, n to one parameter $t \geq 1$:

$$n \times n = t(n_o \times n_0) \tag{5}$$
$$k = t \tag{6}$$

Due to the downscaling process (low-pass filetering), it is feasible to downsample the image without loss of information. Therefore, the two following patches contain similar information, even though they have different sizes:

$$P^{\{L|R\}(k,q=1)}_{n\times n}[x,y] \approx P^{\{L|R\}(k,q)}_{(n\times n)/q}[\lceil x/q \rceil, \lceil y/q \rceil], \forall q \leq k \tag{7}$$

Due to equation 7 we obtain similar patch matching score, if instead of increasing the patch size, we downsample the stereo pair:

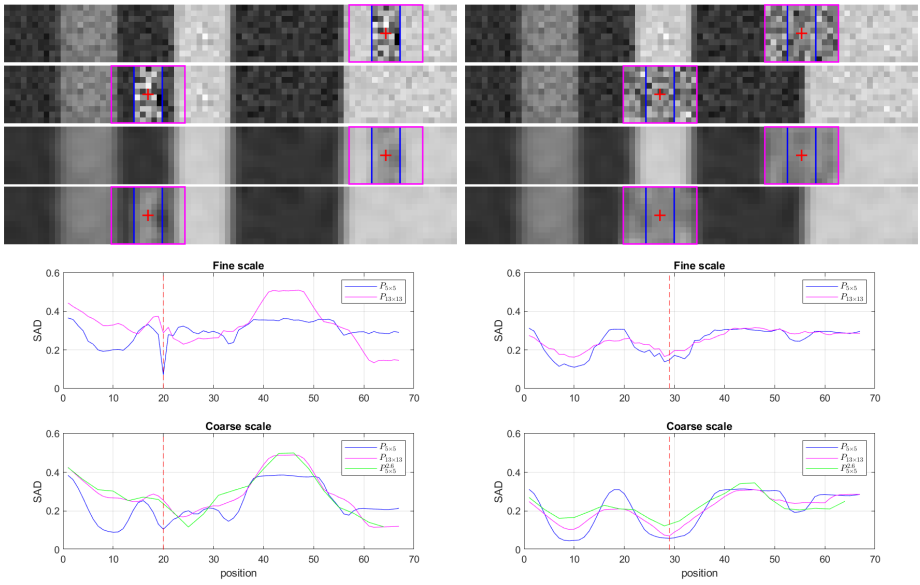

Fig. 1. Simple artificial example. The left and the right column correspond to two different scenarios. In both columns, the first two images are the left and the right stereo view of the same 3D pattern, using orthographic projection. The next two images are the same views downscaled by a Gaussian filter. The red and blue bounding box corresponding to the two different patch sizes and the red cross to the reference point. In the graphs, we observe the curves of the matching process. The red dotted vertical line corresponds to the ground truth disparity.

$$P_{(n_0\times n_0)}^{\{L|R\}(k=t,q=t)}[\lceil x/t\rceil,\lceil y/t\rceil] \approx P_{t(n_0\times n_0)}^{\{L|R\}(k=t,q=1)}[x,y] \tag{8}$$

In figure 1, we design a simple artificial example to test the aforementioned claims and to underline the importance of multiscale processing. We draw two different test cases: in the left-column scenario the fine-scale ($k=1$) with a small patch size ($n_0 \times n_0 = 5\times 5$) leads to superior accuracy, whereas in the right-column scenario the coarse-scale ($k=t=2.6$) with larger patch size ($t(n_0\times n_0) = 2.6(5\times 5) = 13\times 13$) fits better. In both cases, the patch matching scores are similar using $P_{13\times 13}^{\{L|R\}(k=2.6,q=1)}[x,y]$ (red curve) and $P_{5\times 5}^{\{L|R\},(k=2.6,q=2.6)}[\lceil x/2.6\rceil,\lceil y/2.6\rceil]$ (green curve) confirming the claim of 8.

2 Related Work

Stereo reconstruction, as a core problem of computer vision, has been heavily studied over the last decades [1],[2]. Scharstein and Szeliski [3] provided a generic

taxonomy of the stereo vision methods based on their approach to the four fundamental tasks: (a) matching cost computation (b) cost aggregation (c) disparity computation/optimisation and (d) disparity refinement.

Up until recently, most methods relied on hand-engineered features for solving the matching cost and disparity computation problems. There have been proposed many different dissimilarity metrics, such as LoG, CENSUS [4] and BRIEF [5]. The Graph Cut [6], [7] and belief propagation [8] methods managed to incorporate broader information in the final prediction. Hirschmuller minimised a global energy function with the Semi-Global-Matching (SGM) method [9]. Geiger et al. [10] attempted a Bayesian approach, using a prior distribution of the disparity image based on some robustly matched points.

Zagoruyko and Komodakis introduced the use of CNNs for comparing image patches [11]. Zbontar and LeCun [12] used a similar CNN for computing the matching cost. They trained a binary CNN classifier to predict whether two patches correspond to the same 3D point. Their complete method, which involved some non-learnable parts (SGM), achieved state-of-the-art performance in the two well-known stereo datasets Middlebury [13] and KITTI [14]. Luo et al. [15] achieved comparable accuracy with a much faster implementation by treating the matching cost as a multi-label classification problem. For the disparity refinement task, Shaked and Wolf [16] proposed a disparity network which refined the initial disparity predictions, using confidence scores. Gidaris and Komodakis [17] proposed a CNN with three separate parts (Detect, Replace, Refine) for computing a refined disparity image. Seki and Pollefeys implemented a neural network version of the SGM algorithm, which learned from the data the penalty-scores of the discontinuities in the disparity map.

All the approaches mentioned above used a CNN method for solving a particular subtask of the disparity estimation. On the contrary, Mayer et al. [18] implemented an end-to-end CNN architecture (DispNet) for disparity estimation. The architecture included a contracting and an expanding part before making the final prediction. Apart from the proposed model, they also introduced a new synthetic dataset (SceneFlow) with approximately 35000 stereo images. Kendall et al. [19] adjusted their architecture to the stereo vision geometry, by forming explicitly the Cost Matrix. They achieved that by introducing two novelties; (a) a 3D-convolutional network for processing the cost matrix and (b) a differentiable soft-argmin operator for the disparity estimation.

Having achieved to formulate the stereo vision problem with an end-to-end CNN architecture, many works focused on exploiting semantic and context information. For this purpose, it was widely used the encoder-decoder architecture with residual connections, which contains many downscaling and upscaling layers.

Chang et al. [20] designed a Spatial Pyramid Pooling (SPP) architecture for incorporating global information by aggregating context from different scales in the formation of the Cost Matrix. The Cost Matrix was then processed by a stacked-hourglass architecture (encoder-decoder structure). The Cascade Residual Network [21] has two subsequent stack-hourglass CNNs. The first one pro-

duces an initial disparity prediction that is used for warping the reference image and thus estimate an initial (unsupervised) error. Afterwards, the second CNN exploits this estimate for computing a refined prediction. EdgeStereo [22] focuses on the fine details that appear mainly around edges. Noticing that this area is error-prone due to occlusions, the network produces an edge-map which is subsequently used for measuring an edge-aware smoothness loss. DeepPruner [23] focuses on speeding-up the inference time by excluding some disparities and formulating a reduced Cost Matrix. This approach decreases the search space drastically leading to an efficient prediction.

3 The MSNet model

In this section we analyse the building blocks (modules) that comprise the proposed MultiScaleNetwork (MSNet). Each module, named as m_i, is a procedure that gets a tensor as input and produces another one as output. The learnable parts of each module are named as f_i and the non-learnable ones as g_i. All modules are differentiable so that the backpropagation algorithm to be applicable end-to-end. Figure 2 provides an overview of the model.

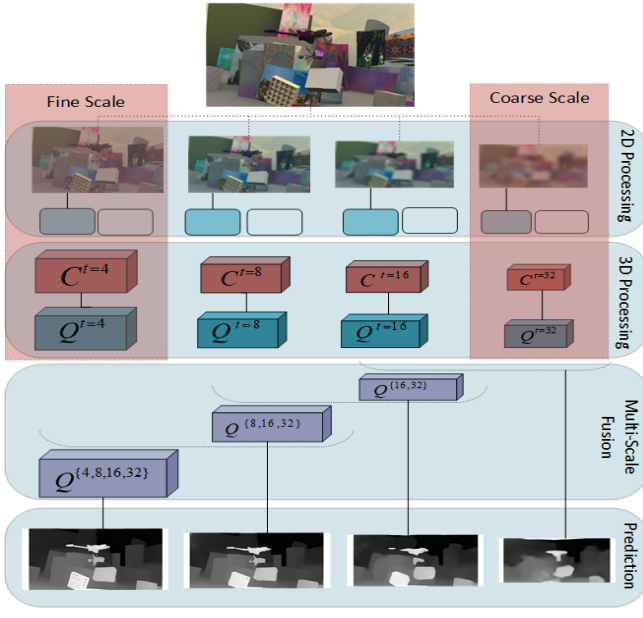

Fig. 2. Overview of MSNet

Downscaling Module - m_1^t The module $m_1^t : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{\lceil H/t \rceil \times \lceil W/t \rceil}$ is applied separately to each stereo image for producing a downscaled stereo pair:

$$(X^{L,t}, X^{R,t}) = (m_1^t(X^L), m_1^t(X^R)) \tag{9}$$

The parameter t is the downscaling and downsampling factor, as defined in equations 5, 6. The module m_1^t is a two-step procedure; firstly each stereo image $X^{\{L|R\}}$ is convolved with the discrete Gaussian kernel of equation 10 in order to remove the high-frequency components, and afterwards the downscaling is performed with bilinear downsampling interpolation.

$$k(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \cdot e^{-(x^2+y^2)/2\sigma^2}, \quad \sigma = t/3 \tag{10}$$

Feature Extraction - m_2 The module $m_2 : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times K}$ extracts local features from the raw stereo images, through a CNN (f_2). The feature extraction process is repeated separately at each scale:

$$m_2 : (X_{desc}^{L,t}, X_{desc}^{R,t}) = (f_2(X^{L,t}), f_2(X^{R,t})) \tag{11}$$

Comparison Volume - m_3 The module m_3

$$m_3 : (\mathbb{R}^{H\times W\times K}, \mathbb{R}^{H\times W\times K}, \mathbb{R}^{H\times W\times 3}) \to \mathbb{R}^{D\times H\times W\times (K+3)}$$

forms the Comparison Volume (C^t), by zipping the comparison information into a 3D tensor. Normally, the Comparison Volume is formed by simply concatenating the descriptors in each disparity position. This approach introduces significant redundancy; from the $D \times H \times W \times 2K$ values of the Comparison Volume only the $H \times W \times 2K$ are unique. For reducing such redundancy, we design a new comparison metric between two scalar features:

$$l : \mathbb{R}^2 \to \mathbb{R} : l(a_1, a_2) = \frac{|a_1| + |a_2|}{2} \cdot e^{-|a_1 - a_2|} \tag{12}$$

The first term $\frac{|a_1|+|a_2|}{2}$ measures the existence of a feature in each patch separately (0 signifies non-existence) and the second term $e^{-|a_1-a_2|} \in (0, 1]$ measures the coexistence of the feature in both patches. Finally, we concatenate the raw left image X^L, in order to propagate the raw image in the subsequent layers. The formulation of the comparison volume is described in equation 13:

$$C^t[d, x, y, i] = \begin{cases} l(X_{desc}^{L,t}[x, y, i], X_{desc}^{R,t}[x - d, y, i]) & , i \leq K \\ X^{L,t}[x, y, i - K] & , i > K \end{cases} \tag{13}$$

Comparison Volume Processing - m_4 The module $m_4 : \mathbb{R}^{D\times H\times W\times K} \to \mathbb{R}^{D\times H\times W\times K}$ is a CNN (f_4) with 3D convolutional layers, that incorporates local information along all 3 dimensions (the 2 spatial ones x, y and the disparity d) for refining the Comparison Volume. It ouptuts a same-dimension volume Q^t:

$$m_4 : Q^t = f_4(C^t) \tag{14}$$

Multiscale Fusion - m_5 The module m_5 is responsible for exploiting the information from different scales and combine it in a single tensor. The procedure takes place recursively in pairs of two Comparison Volumes, from coarse to fine scales. A trilinear upsampling layer $g_3^t : \mathbb{R}^{D\times H\times W\times K} \rightarrow \mathbb{R}^{tD\times tH\times tW\times K}$ is applied to the low-dimension Comparison Volume before a CNN (f_5) with 3D-Convolutional layers merge the information of the two scales into one tensor, as describe in equation 15. Repeating this procedure recursively as shown in algorithm 1 leads to a single Comparison Volume, that incorporates information from all separate scales. The implementation of the scale fusion procedure as a recursive merging is the key idea that enables our network to be readjustable:

$$Q^{\{t_i,\cdots,t_n\}} = f_5(Q^{t_i} \oplus g_5^{t_i/t_{i-1}}(Q^{\{t_{i-1},\cdots,t_n\}})) \tag{15}$$

Algorithm 1 Multi-scale fusion - Module m_5

procedure MULTI SCALE FUSION($Q^{t_0}, Q^{t_1}, \cdots, Q^{t_n}$) $\rightarrow Q^{\{t_0,t_1,\cdots,t_n\}}$
  $Q \leftarrow Q^{t_n}$ ▷ Initialize
  for `i=n-1;-1;0` **do**
    $Q \leftarrow g_3^{t_{i+1}/t_i}(Q)$ ▷ 3D Upsampling
    $Q \leftarrow Q^{t_i} \oplus Q$ ▷ Concatenation
    $Q \leftarrow f_5(Q)$ ▷ Merge information
  end for
  return $Q^{\{t_0,t_1,\cdots,t_n\}} \leftarrow Q$ ▷ Result
end procedure

Module For Depth Prediction - m_6 The module $m_6^t : \mathbb{R}^{D/t\times H/t\times W/t\times K} \rightarrow \mathbb{R}^{D\times H\times W}$ implements the final disparity prediction. It consists of three sequential parts: (a) a CNN with 3D-convolutional layers $f_6 : \mathbb{R}^{D\times H\times W\times K} \rightarrow \mathbb{R}^{D\times H\times W}$ that assigns a probability in each possible disparity $S^{\{t_0,\cdots,t_n\}} = f_6(Q^{\{t_0,t_1,\cdots,t_n\}})$, (b) a trilinear upsampling layer for upsampling S in the initial dimensions $D \times H \times W$ and finally (c) a softmax operator applied along the disparity dimension, for obtaining the final prediction $\hat{Y}^{\{t_0,\cdots,t_n\}}$.

Putting all pieces together Before the execution of a single prediction, a set of processing scales $T = \{t_0, ..., t_n\}$ must be defined. The modules m_1, m_2, m_3, m_4 operate separately on the stereo pair for producing a single scale Comparison Volume Q^t. Subsequently, the module m_5 combines the information from all single-scale volumes to a single tensor $Q^{t_0,\cdots,t_n}$ and the module m_6 makes the disparity prediction. The whole prediction procedure is summarized in algorithm 2.

Algorithm 2 MultiScaleNetwork (MSNet)

$T \leftarrow \{t_0, t_1, ..., t_n\}$ ▷ Define Processing Scales
procedure MSNET(X^L, X^R, T) $\rightarrow \hat{Y}$
  for `t in T` **do**
    $(x^{L,t}, x^{R,t}) \leftarrow (m_1^t(X^L), m_1^t(X^R))$ ▷ Downscaling
    $(x^{L,t}_{desc}, x^{R,t}_{desc}) \leftarrow (m_2(X^L), m_2(X^L))$ ▷ Features
    $C^t \leftarrow m_3(X^{L,t}_{desc}, X^{R,t}_{desc}, X^{L,t})$ ▷ Comparison Volume
    $Q^t \leftarrow m_4(C^t)$
  end for
  $Q^{\{t_0,t_1,\cdots,t_n\}} \leftarrow m_5(Q^{t_0}, Q^{t_1}, \cdots, Q^{t_n})$ ▷ MultiScaleFusion
  $\hat{Y}^{\{t_0,\cdots,t_n\}} = m_6(Q^{\{t_0,t_1,\cdots,t_n\}})$ ▷ Prediction
  return $\hat{Y}^{\{t_0,\cdots,t_n\}}$
end procedure

Cnn architectures

The learnable parts of MSNet (f_2, f_4, f_5, f_6) are four CNNs, that follow similar architecture. A residual connection, which comprises of two blocks of Batch Normalization, ReLU and Convolution in sequential order with a residual connection added to the output, is used as a fundamental building block in all architectures. In f_2, which operates on 2D inputs, the convolution is 2D (i.e. kernel size 3x3), whereas for f_4, f_5, f_6 the convolution is 3D, applied along the disparity dimension as well (kernel size 3x3x3).

Module	Structure	Input	Output	Parameters
f_2	BN+ReLU+Conv2D	HxWx3	HxWx32	74592
	4xRes2D	HxWx32	HxWx32	
f_4	BN+ReLU+Conv3D	DxHxWx64	DxHxWx32	196128
	3xRes3D	DxHxWx32	DxHxWx32	
f_5	BN+ReLU+Conv3D	DxHxWx64	DxHxWx64	331776
	BN+ReLU+Conv3D	DxHxWx64	DxHxWx32	
	3xRes3D	DxHxWx32	DxHxWx32	
f_6	BN+ReLU+Conv3D	DxHxWx64	DxHxWx64	139104
	2xRes3D	DxHxWx32	DxHxWx32	
	BN+ReLU+Conv3D	DxHxWx32	DxHxW	
MSNet		HxWx3	HxW	741600

Table 1. Description of CNN architectures

4 Experimental Evaluation

The experimental evaluation is organized in four parts; In the first part (4.1), MSNet is compared to the State-of-the-art methods, in terms of accuracy and efficiency. In the second part (4.2, 4.3), we set up an internal benchmark for questioning the major architectural choices of MSNet; we evaluate some alternative models that share similar designing principles with the MSNet (i.e. same

backbone architecture) with crucial modifications in the decisive details we want to measure. In the third part (4.4), we observe how MSNet behaves under different sets of processing scales, quantifying the trade-off between accuracy and efficiency. In the final part (section 4.5), we illustrate how MSNet has learned to incorporate information from all processing scales in an incremental fashion.

All experiments are based on the synthetic SceneFlow dataset [24], which consists of dense ground truth disparity images (H=540, W=960). The default training/test set split (35454 training vs 4370 testing images) has been followed by all benchmarks. For the evaluation of the models, we use (a) the End-Point-Error (EPE) metric - mean absolute prediction error - and (b) the percentage (PCG_k) metric - the percentage of points with absolute error over k pixels. All models have been trained from scratch in a GeForce GTX 1080 GPU, with random initialization following the fan-in approach ($\theta \sim N(0, \sigma = \sqrt{2/n})$) [25]. The training images are randomly cropped in (H=256, W=512) patches and normalized with ImageNet preprocessing statistics (mean, std). The Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with a learning rate of 0.001 was used for 15 epochs, with batch size set at 2. At the training phase of MSNet, the processing scales were preset to $T = \{2^2, 2^3, 2^4, 2^5\}$ and the final loss L was a weighted sum of the EPE loss computed after incorporating each processing scale:

$$Loss := \frac{1}{2}L^{\{t_0,t_1,t_2,t_3\}} + \frac{1}{6}L^{\{t_1,t_2,t_3\}} + \frac{1}{6}L^{\{t_2,t_3\}} + \frac{1}{6}L^{\{t_3\}} \tag{16}$$

$$\text{where: } L^{\{t_i,\ldots,t_j\}} = |\hat{Y}^{\{t_i,\ldots,t_j\}} - Y| \tag{17}$$

4.1 Comparison With The State-Of-The-Art

The comparisons are based on three criteria; (a) accuracy (EPE, PCG_3) (b) efficiency (runtime) and (c) the number of free parameters. The efficiency is measured as the runtime for a single prediction. The number of free parameters is manifested since it is related to the memory resources and the amount of training data demanded by each model. In table 2, we observe that MSNet achieves competitive performance (i.e. 1.017px EPE compared to the 0.74px of the SOTA [23]) even though it uses much less free parameters and it is more efficient than (almost) all its competitive networks. Specifically, only [26] outperforms MSNet both in terms of accuracy and efficiency; models [27], [23], [28] are more accurate but less efficient and they use more free parameters. The only model (apart from [26]) that is more efficient than MSNet is [18], but is less accurate.

4.2 The Effect Of Reinforcing Multi-Scale Processing

In this section, we question whether enforcing the multi-scale processing explicitly, as we do in MSNet, is advantageous over using multi-scale processing internally as part of the CNN architecture. For this reason, we design four

Method	Parameters(M)	Runtime(s)	EPE(px)	PCG_3(%)
Our benchmark - IMS architectures				
MSNet	0.741	0.32	1.017	3.98
OneRes (S)	0.463	0.11	1.508	6.11
MRes2d (S)	0.659	0.10	1.671	6.98
MRes3d (S)	0.514	0.15	1.504	5.86
MRes2d3d (S)	0.677	0.3	1.897	8.078
OneRes (B)	1.608	0.22	1.37	5.53
MRes2d (B)	1.458	0.17	1.32	5.43
MRes3d (B)	1.682	0.21	1.322	5.11
MRes2d3d (B)	1.772	0.32	1.613	6.67
Our benchmark - Free weights				
Free2d	0.969	0.24	1.126	4.33
Free3d	2.75	0.33	0.882	**3.48**
Free2d3d	2.974	0.33	1.107	4.36
SOTA				
PSMNet[20]	5.2	0.45	1.09	-
CRL[21]	-	0.47	1.32	-
DispNetC[18]	-	**0.06**	1.68	-
GC-Net[19]	3.5	0.95	2.51	9.34
Edge-Stereo[22]	-	-	1.12	4.99
CSPN[27]	250	0.5	0.78	-
GA-Net[28]	2.3	1.5	0.84	-
AMNet[23]	4.37	-	**0.74**	-
DeepPruner[26]	-	0.6	0.97	-

Table 2. Model comparison on the SceneFlow dataset. The EPE and PCG of MSNet are measured for the default set of scales $T = \{2^2, 2^3, 2^4, 2^5\}$.

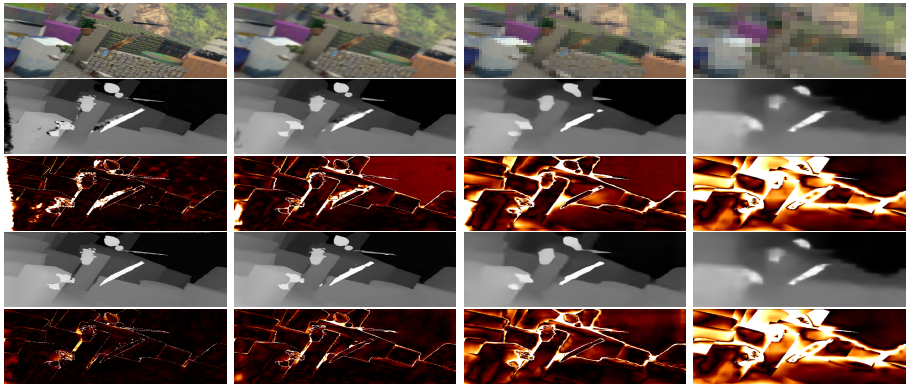

Fig. 3. Illustration of the prediction process at different processing scales. The first row (left to right) contains the left image at each scale $X^{L,t} : t \in \{2^2, 2^3, 2^4, 2^5\}$. The second and third-row contain the **single scale** predictions $\hat{Y}^t$ and the corresponding absolute error images E^t, following the scale ordering of the first row. The last two rows contain the **multiscale** predictions; processing scales are added as we move from right to left (i.e. $\hat{Y}^{\{t_0=2^5\}}$ corresponds to the far-right image, while $\hat{Y}^{\{t_0=2^2,\ldots,t_0=2^5\}}$ to the far-left one) and the corresponding error images. We can observe the limitations of single-scale predictions; coarse scales fail in describing the details between objects while fine scales suffer from instabilities. On the other hand, the multiscale prediction combines the benefits of both worlds; it starts with a rough estimation of the disparity image, incorporating higher-resolution details as it integrates fine scales.

new models that follow the MSNet designing principles, but without reinforcing multi-scale processing explicitly; instead, they follow the hourglass architecture (i.e. encoder-decoder) which is the proposed method, by the literature, for applying multi-resolution processing internally. We design the following four architectures; OneRes operates only on the initial resolution, MRes2d uses the hourglass model only in the 2D-processing part (f_2), MRes3d only in the 3D-processing part (f_4, f_5, f_6) and finally MRes2d3d in both the 2D and 3D processing part (f_2, f_4, f_5, f_6). For each of the 4 CNNs, we create two versions; the small (S) version with the same order of free parameters as the MSNet (approximately $600K$) and the big (B) version which has as many parameters as the GPU's memory allows (approximately $1.6M$). For clarity, we call all these new models with the common name Implicit Multi-Scale (IMS) architectures.

In figure 4 (Left), we observe that MSNet outperforms both versions of the IMS architectures, which is a strong indicator that reinforcing multi-scale processing explicitly is beneficial for the problem of depth estimation. As expected, all big (B) networks outperform small (S) ones.

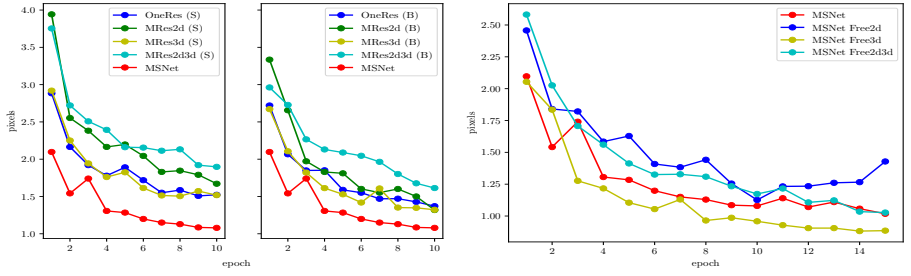

Fig. 4. (Left) MSNet vs IMS architectures. (Right) MSNet vs Free*-Weight Networks.

4.3 The Cost of Sharing Weights

MSNet obtains its fundamental advantage of being adjustable by repeating its learnable-parts in between the different processing scales. An important question that arises is how much is the cost in terms of accuracy for gaining such scalability; what is the corresponding EPE if we use the MSNet architecture without weight sharing (i.e. train different learnable parts at each scale, instead of repeating the same building blocks). For answering this question, we create a second internal benchmark, by implementing three new CNN architectures; The MSNet-Free2d has free weights only in the 2D-processing part (does not repeat f_2 blocks), MSNet-Free3D has free weights only in the 3D-processing part (does not repeat f_4, f_5, f_6 building blocks) and, finally, MSNet-Free2D3D has free weights in both parts. Since these new architectures are no longer adjustable, we predefine the processing scales to be $T = \{2^2, 2^3, 2^4, 2^5\}$ for being in-line with MSNet.

The evaluation on the SceneFlow dataset is presented in figure 4. We notice that training different CNNs for each scale of the 3D-processing part (f_4, f_5, f_6) leads to superior performance. This experimental observation indicates that the MultiScaleFusion algorithm, which incorporates the different scales in pairs of two, achieves a suboptimal solution. This observation can be explained due to two reasons; (a) the number of free weights in MSNet Free-* models is increased by a factor of $4x$ compared to MSNet and (b) MultiScaleFusion's designing principle of incorporating only one new scale at a time, deprives the beneficial global view of all scales simultaneously. Even though MSNet-Free3D outperforms our proposed model, such approach cancels the fundamental scalability advantage of MSNet.

4.4 MSNet Evaluation Under Different Scale Combinations

The fundamental advantage of MSNet compared to other architectures is the ability to get tuned between accuracy and efficiency at inference. In figure 5, we measure how three fundamental properties of MSNet (a) accuracy (b) stability and (c) efficiency vary under different scale combinations. For the accuracy, measured through the EPE, we can observe two interesting properties; (a) combined scales have different impact on the accuracy compared to when applied individually (i.e. T $= \{t_0 = 4, t_1 = 32\}$ has the best performance among all two-scale combinations, though $t_0 = 4$, $t_1 = 32$ have the worst accuracy individually) and (b) adding an additional scale in a set always improves accuracy. Stability is quantified through the standard deviation σ of the EPE. We observe that coarse scales tend to be more stable and that adding processing scales increases stability. Finally, we notice that the runtime increases exponentially as we add scales and that it is mainly driven by the downscaling factor t of the highest resolution rather than the amount of scales involved (e.g the set $T = \{t_0 = 4\}$ is less efficient than $T = \{t_0 = 8, t_1 = 16, t_2 = 32\}$).

In general terms, MSNet has a robust behavior under different scale combinations. Adding a new processing scale increases the accuracy. Excluding the computational demanding fine scales, improves the efficiency exponentially, without an accuracy crumble. As a characteristic example, the combination $T = \{8, 32\}$ has EPE < 2 px and execution time $< 0.05s$.

4.5 Multi-Scale fusion analysis

In this section, we provide a qualitative illustration of how MSNet gradually adopts the information from coarse to fine scales.

In figure 6, we focus on the *MultiScaleFusion* algorithm. We choose two regions with different properties; the yellow-box region contains a thin object with rich texture, appropriate for a small, fine-scale patch. Conversely, the orange-box contains a background wall with a repetitive pattern, appropriate for a large, coarse-scale patch. We verify that the single-scale predictions agree with our intuition; only fine-scales ($t = 2^2, t = 2^3$) succeed in the yellow-box area, whereas only coarse-scales succeed in the orange area ($t = 2^4, t = 2^5$).

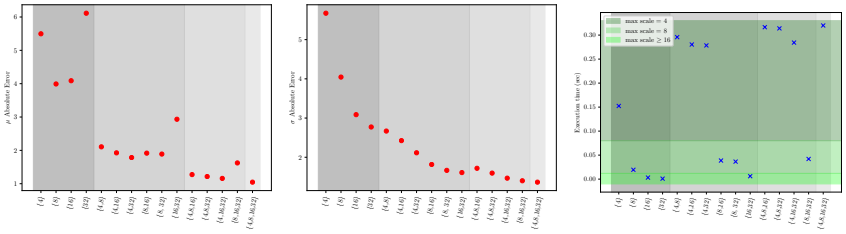

Fig. 5. Presentation of MSNet accuracy and efficiency for different scale combinations. On the x-axis, we have alligned the scale combinations in increasing order according to the number of processing scales (the shades of grey represent groups with an equal amount of scales). For each scale combination; the first plot represents the absolute μ end-point-error (EPE), the second plot, the absolute deviation σ of the EPE and the third plot the inference time.

The multi-scale prediction succeeds in both scenarios. In the yellow-box area, the model initiates with an erroneous prediction from the coarse-scale $t = 2^5$, but it gradually esteems the (most accurate) fine-scales. In the orange-box, where the initial prediction $t = 2^5$ is accurate, the model remains unaffected by the erroneous coarse-scales.

In figure 3, all the single and multi-scale predictions of a single example (from the test set) are presented accompanied by their corresponding error images. We observe that the prediction procedure follows a step-by-step refinement of the initial low-resolution prediction; initially, the network predicts a rough coarse-scale estimation of the depth. As it explores the fine-scales, it gradually adopts high-resolution details for producing the final disparity map.

5 Conclusion and Future Work

In this paper, we proposed a scalable CNN model for depth estimation, that can be tuned for accuracy or efficiency, according to the priorities of the user. For achieving such agility, we designed and trained all the learnable parts of the model in scale-independent tasks (feature extraction, combining information in pairs of two processing scales etc.), which enables the definition of the processing-scales during inference. We confirmed that in general terms, our model has learned to utilise only the accurate sub-regions from each processing scale and discard the erroneous ones. We also observed that reinforcing multi-scale processing explicitly leads to superior performance compared to the models that apply multi-scale implicitly. Finally, our method exhibits competitive results compared to the State-of-the-art methods in the SceneFlow dataset, even though it involves significantly less learnable parameters.

As we proposed, the most significant advantage of our method is the privilege of choosing the processing scales at inference time. Nevertheless, we provide no tool for answering a critical question; which specific scales to use according

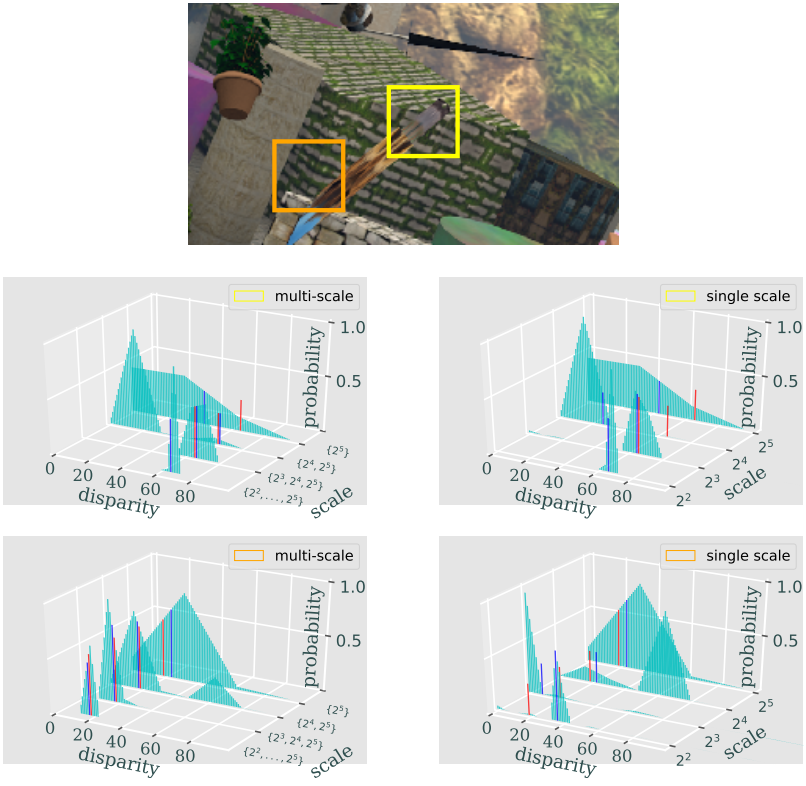

Fig. 6. Illustration of the prediction process at two qualitatively different sub-regions of the image. The first row of plots is related to the yellow-box-region and the second to the orange-box-region. In all graphs, we observe the probabilities attached to each disparity under consideration. The cyan area represents the probability mass attached to each disparity; the blue line is the predicted disparity after the $softargmax$ operator is applied; the red line is the ground-truth disparity. The single-scale graphs (right column) exhibit the probability mass when the prediction is made at each scale separately. Conversely, in the multi-scale graphs (left column) the prediction is based on an incremental number of scales; from $\hat{Y}^{\{t_0=2^5\}}$ to $\hat{Y}^{\{t_0=2^2, t_1=2^3, t_2=2^4, t_3=2^5\}}$

to the properties of the stereo pair. Hence, the scale specification depends on the availability of computational resources and the prioritisation of accuracy or efficiency, based on the following general rule; incrementing the number of scales increases accuracy but deducts efficiency and vice versa. However, apart from the above general rule, there is a qualitative relationship between the appropriate processing scales and the properties of the stereo pair; future work should focus on exploring this relationship.

References

1. Barnard, S.T., Fischler, M.A.: Computational Stereo. ACM Computing Surveys (1982)
2. Brown, M.Z., Burschka, D., Hager, G.D.: Advances in computational stereo (2003)
3. Scharstein, D., Szeliski, R., Zabih, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In: Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001. (2001)
4. Zabih, R.: A non-parametric approach to visual correspondence. IEEE Transactions on Pattern Analysis and Machine (1996)
5. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF : Binary Robust Independent Elementary Features. Computer (2010)
6. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. Proceedings Eighth IEEE International Conference on Computer Vision, ICCV 2001 **2** (2001) 508–515
7. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence (2001)
8. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: Proceedings - International Conference on Pattern Recognition. (2006)
9. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence (2008)
10. Geiger, A., Roser, M., Urtasun, R.: Efficient large-scale stereo matching. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). (2011)
11. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 4353–4361
12. Zbontar, J., LeCun, Y.: Computing the stereo matching cost with a convolutional neural network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015)
13. Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., Westling, P.: High-resolution stereo datasets with subpixel-accurate ground truth. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). (2014)
14. Menze, M., Heipke, C., Geiger, A.: Joint 3d estimation of vehicles and scene flow. In: ISPRS Workshop on Image Sequence Analysis (ISA). (2015)
15. Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 5695–5703
16. Shaked, A., Wolf, L.: Improved stereo matching with constant highway networks and reflective confidence learning. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 4641–4650
17. Gidaris, S., Komodakis, N.: Detect, replace, refine: Deep structured prediction for pixel wise labeling. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. (2017)
18. Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 4040–4048

19. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-End Learning of Geometry and Context for Deep Stereo Regression. In: Proceedings of the IEEE International Conference on Computer Vision. (2017)
20. Chang, J.R., Chen, Y.S.: Pyramid Stereo Matching Network. cvpr (2018)
21. Pang, J., Sun, W., Ren, J.S., Yang, C., Yan, Q.: Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching. In: Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017. (2018)
22. Song, X., Zhao, X., Hu, H., Fang, L.: EdgeStereo : A Context Integrated Residual
23. Du, X., El-Khamy, M., Lee, J.: Amnet: Deep atrous multiscale stereo disparity estimation networks. arXiv preprint arXiv:1904.09099 (2019)
24. Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). (2016) arXiv:1512.02134.
25. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. (2015)
26. Duggal, S., Wang, S., Ma, W.C., Hu, R., Urtasun, R.: Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In: ICCV. (2019)
27. Cheng, X., Wang, P., Yang, R.: Learning depth with convolutional spatial propagation network. arXiv preprint arXiv:1810.02695 (2018)
28. Zhang, F., Prisacariu, V., Yang, R., Torr, P.H.: Ga-net: Guided aggregation net for end-to-end stereo matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 185–194