# Multi-scale Iterative Residuals for Fast and Scalable Stereo Matching

Kumail Raza
René Schuster
Didier Stricker
kumail.raza@dfki.de
rene.schuster@dfki.de
didier.stricker@dfki.de
German Research Center for Artificial Intelligence – DFKI
Kaiserslautern, Germany

## ABSTRACT

Despite the remarkable progress of deep learning in stereo matching, there exists a gap in accuracy between real-time models and slower state-of-the-art models which are suitable for practical applications. This paper presents an iterative multi-scale coarse-to-fine refinement (iCFR) framework to bridge this gap by allowing it to adopt any stereo matching network to make it fast, more efficient and scalable while keeping comparable accuracy. To reduce the computational cost of matching, we use multi-scale warped features to estimate disparity residuals and push the disparity search range in the cost volume to a minimum limit. Finally, we apply a refinement network to recover the loss of precision which is inherent in multi-scale approaches. We test our iCFR framework by adopting the matching networks from state-of-the art GANet and AANet. The result is 49× faster inference time compared to GANet-deep and 4× less memory consumption, with comparable error. Our best performing network, which we call FRSNet is scalable even up to an input resolution of 6K on a GTX 1080Ti, with inference time still below one second and comparable accuracy to AANet+. It out-performs all real-time stereo methods and achieves competitive accuracy on the KITTI benchmark.

## CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Machine learning algorithms*; *Scene understanding*; Vision for robotics; *Matching*.

## KEYWORDS

stereo, efficiency, coarse-to-fine, cost volume, matching

## 1 INTRODUCTION

Stereo matching is one of the top research areas in modern computer vision, with enormous applications in the industry particularly in autonomous driving and robotics. The idea is to reconstruct dense 3D geometry by estimating the disparity between image pixels in a rectified stereo image pair. To this end, dense matching pixel correspondences are used. Since, this is one of the major classical problems in computer vision, it has been studied substantially for more than half a century [30] and has been matured. A huge amount of literature is available proposing various architectures both classical and modern, which aim to solve the problem of stereo matching successfully. A large part of these are deep neural networks. Although most of these networks provide sub-pixel accuracy with a state-of-the-art pixel outlier rate, a small number of them concentrate on the execution time and model growth, especially in the case of end-to-end trainable networks. The architectures with a design focus on run-time have much worse end point errors (EPE) and pixel error rates (ER) and are not comparable to the state-of-the-art models at all [6, 14, 29].

The problem of stereo matching is solved traditionally with some key steps namely: 1) Extracting features, 2) Cost volume construction, 3) Cost aggregation for matching, 4) Regressing the final disparity. Two types of approaches exist in the literature incorporating some or all of these steps. These are direct disparity regression without building a cost volume and by using cost volume filtering. The first category disregards geometric variations by calculating disparity on dense matching pixel correspondences. These approaches although being relatively fast lead to large EPE and ER [10]. The latter namely *volumetric methods*, construct and refine a cost volume. The idea is to build a higher dimensional feature volume containing all the candidate disparity values up to a maximum disparity range. These approaches are slower in classical processing as they incorporate some customized versions of dynamic programming algorithms for finding the best matching features over the cost volume. Neural networks built on volumetric matching often use two sub-networks in an end-to-end architecture. The first sub-net extracts features from the stereo image pairs which are used to build a 3D or 4D cost volume. The second network then uses 3D convolutions to compute matching costs in the cost volume before finally regressing the final disparity values. The run-time for these models is also nowhere comparable to real-time as these have to employ heavy operations on the high dimensional cost volume.

There exists a gap in the literature between real-time models with higher EPE and ER and slower state-of-the-art models with lower EPE and ER. We aim to bridge this gap by proposing a novel multi-scale coarse-to-fine refinement framework to estimate disparity by leveraging the already produced features usually at different resolutions by the feature extraction backbone and using much shallower cost volumes. These lower resolution disparity maps are then iteratively refined using the notion of disparity residuals. The idea being that these lower resolution operations on smaller disparity search ranges will be computationally more efficient and much more scalable in terms of memory while keeping the EPE and ER comparable. As shown in Figure 6 and Table 2, our approach bridges the accuracy gap between the real-time and state-of-the-art

stereo matching models such as GANet [29] and AANet [23], while staying as close as possible to real-time performance.

We introduce the notion of a *prediction head*, which can be adopted from any stereo matching architecture and is used in our iterative coarse-to-fine refinement algorithm (iCFR). Separate heads are produced for each scale and their parameters are learned in an end-to-end fashion. We also propose a final refinement layer to recover the loss of precision that is inherent in down and upsampling operations. The resulting final network which we call FRSNet (Fast iterative Residual Stereo Network), using the iCFR, outperforms all available real-time architectures for stereo matching such as StereoNet [14], DispNet [17], Toast [27] and other deeper networks like GC-Net [12] while producing comparable EPE and ER to modern state-of-the-art networks such as AANet [23], PSMNet [2] and GANet [29], on the Sceneflow dataset [17] and the KITTI benchmark [8].

## 2 RELATED WORK

Traditional algorithms for stereo matching [7, 10, 18, 21, 24] yield discontinuous disparity maps with higher pixel outlier rates. This is because they usually use feature-based matching which does not take the geometric variations in images into account. Thus, the matching is ambiguous, due to occlusions, reflections, texture-less surfaces and repetitive patterns. Volumetric methods with cost volume construction and cost aggregation strategies were developed to achieve better matching [11, 16, 24]. Then with the advent of deep learning, these strategies transformed into learnable layers in terms of convolution and other operations. The first deep learning architecture for stereo matching was proposed in MC-CNN [28], which used a feature extraction network instead of handcrafted features. Then, DispNet [17] provided a real-time end-to-end disparity estimation approach by using an encoder-decoder to directly regress the final disparity map. Since this network does not use prior information, it is completely data-driven and requires a huge dataset to train on. To this end, the authors of DispNet also presented the synthetic Sceneflow dataset [17] to train such kind of end-to-end networks, which we also partially use to train our network. Then a two-stage convolutional neural network was presented by Pang *et al.* [20], which separates the workflow for feature extraction and disparity refinement. The first approach with a 4D cost volume was proposed in GC-Net [12]. It applies 3D convolutions to aggregate the matching costs. GC-Net presents a three stage end-to-end trainable network for stereo estimation following the key steps of stereo matching *i.e.* feature extraction, cost aggregation and disparity regression. It achieved state-of-the-art accuracy, however the inference times are much higher compared to the real-time models.

PSMNet [2] uses a feature pyramid network and stacked hourglass network along with twenty-five convolution layers to achieve state-of-the-art EPE and ER. However, higher number of convolutional layers deteriorate the run-time quite considerably and puts it far from the real-time category. GANet [29] proposes to incorporate geometric optimization algorithms like SGM [10] in deep learning pipelines as learnable layers. To this end, it presents the concept of guided aggregation layers in the end-to-end pipeline. These layers aim to leverage the local neighbourhood information as well as the global image context into guiding convolutions in the cost aggregation network. These layers push the sub-pixel accuracy even further and replace a large number of computationally complex 3D convolutions. LEAStereo [5] applies neural architecture search to choose the best possible parameters for a set of operations performed in the network for stereo matching. It achieves state-of-the-art accuracy on KITTI benchmark as well as close to real-time performance. This however, requires enormous amounts of computational resources to compute the best neural architecture for the job. To cater for this, the authors applied task-specific human knowledge to come up with a three step stereo matching architecture and refined the parameters with the neural architecture search, *e.g.* convolution filters sizes, strides, *etc.* Other faster techniques use coarse-to-fine matching with hierarchical upsampling [3, 6, 9, 22, 25, 26]. Among them, DeepPruner achieves great performance. However, they ignore recovering details lost by lower resolution matching. Despite considerable research for the last half-century, a stereo matching architecture with a focus on practical implications is still an open question in the computer vision community, where a state-of-the-art algorithm can be applied to real-time applications. Our approach takes a step in this direction.

## 3 FRSNET: A MULTI-SCALE IMPROVEMENT

In this section, we describe the details of our proposed multi-scale iterative coarse-to-fine refinement framework for stereo matching (iCFR) and a proposed best performing network called FRSNet which uses this iCFR. The overall architecture is shown in Figure 1.

### 3.1 Feature Extraction

As a feature extraction backbone, we employ a stacked hourglass network with skip connections between corresponding layers of the encoder and decoder [23, 29]. This densely connected network is shared by the input stereo image pair. We use the already calculated features at multiple scales from the decoder part (*i.e.* $\frac{1}{3}$, $\frac{1}{6}$, $\frac{1}{12}$, $\frac{1}{24}$, $\frac{1}{48}$), as input to the prediction heads at the corresponding scales. Figure 1 shows the feature extraction network with dense connections. These features are then either concatenated or correlated to form the cost volume which is then processed in the prediction head.

### 3.2 Prediction Head

We combine the three latter parts of the stereo matching pipeline namely cost volume construction, cost aggregation and disparity regression into a *prediction head*, for modularity. Modern volumetric end-to-end stereo matching architectures such as GC-Net [12] and GANet [29] employ a 4D cost volume. This cost volume is built by either concatenating or correlating the features from the stereo image pair, extracted from the feature extraction backbone, typically a stacked hourglass network [19] or a Feature Pyramid [15]. Depending on the adaptation from any stereo matching network, the cost volume can have different dimensions. Typically a correlation cost volume requires much less memory and a lower number of learnable parameters. The cost volume is then filtered by the adapted strategy of the corresponding stereo matching architecture. Finally, the disparity regression layer estimates the output disparity map. Note that the prediction head is designed to be adopted from any stereo matching architecture. As shown in Figure 3, the prediction

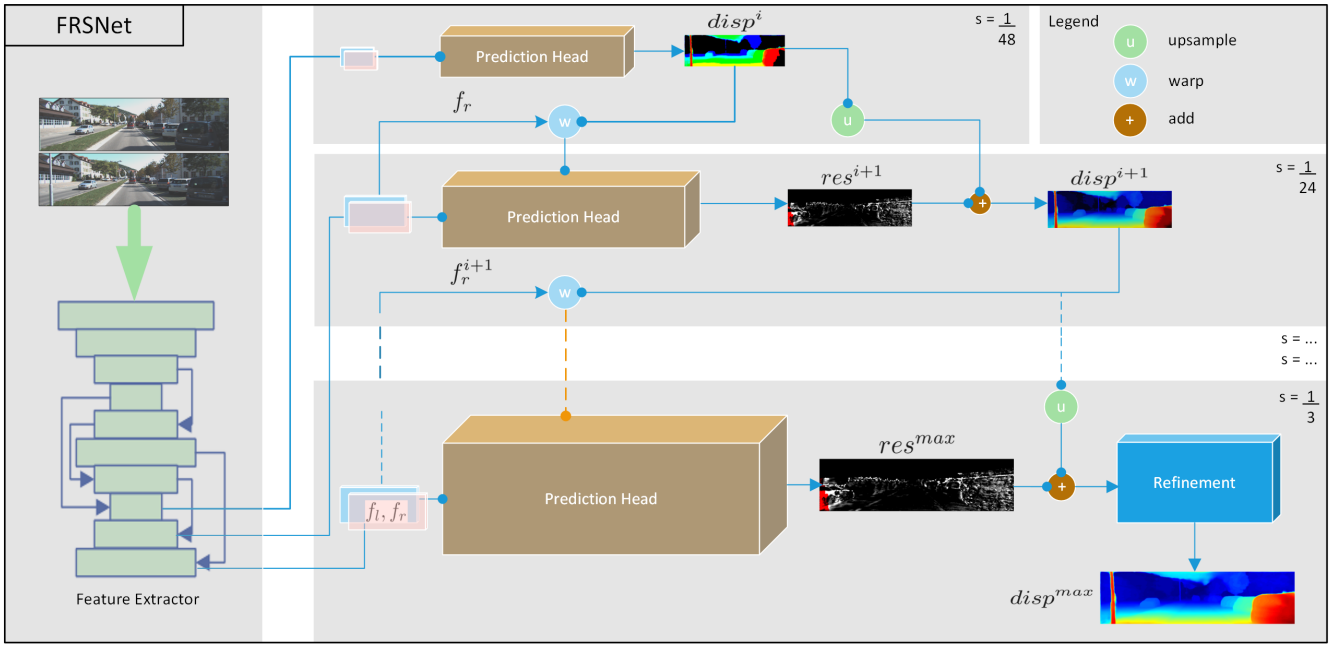



**Figure 1. Overview of the architecture of our proposed FRSNet. Given a rectified stereo image pair, a stacked hourglass network with residual connections extracts the features. As the decoder in the stacked hourglass produces features at different scales, we employ them directly in the proposed iCFR algorithm, instead of down-sampling them. In the first iteration, the smallest resolution features are used *i.e.* at $\frac{1}{48}$ of the original resolution to calculate a dense but very coarse disparity map $disp^i$. This full disparity map is then used to warp the target image features $f_r^{i+1}$ of the next scale *i.e.* $\frac{1}{24}$. Using these warped features, the prediction head produces a residual disparity map $res^{i+1}$ this time. This is then added to the upsampled $disp^{i'}$ to get $disp^{i+1}$. The process is continued for all the remaining scales until passing the final disparity $disp^{i+4}$ through the refinement block to get the final disparity map $disp^{max}$ at input resolution.**

head can also produce multiple disparity maps from different stages of cost aggregation for intermediate supervision during training. The ground truth is down-sampled accordingly to calculate the loss for each of these predictions. Sections 3.6 and 3.7 describe the working of the overall multi-scale architecture using prediction heads from two state-of-the-art networks GANet [29] and AANet [23].

*3.2.1 Symmetric Cost Volume.* The disparity map produced at the smallest scale *i.e.* $\frac{1}{48}$ is a very coarse estimate, usually consisting of just blobs of information. These blobs when up-sampled, produce large erroneous estimates of disparity initially. If these are not corrected in the next subsequent scale, the error keeps accumulating and amplifying much like a drift error. Negative disparity residuals are required to correct such coarse estimates. These residuals will help reduce the disparity values if they are initially too high. To this end, we change the way the cost volume is build to allow negative disparity hypotheses. Figure 2 illustrates the construction of such a cost volume. The shape of the cost volume will remain the same $[F, D_{cv}, H, W]$, however the way it is filled along the disparity dimension, will be different. The idea is to fill the disparity dimension of length $D_{cv}$, with disparity hypotheses symmetrically in both directions, *i.e.* positive and negative. This means that in position $\frac{D_{cv}}{2}$, the candidate with zero displacement will be placed. From here in both directions, the cost volume will be filled for the

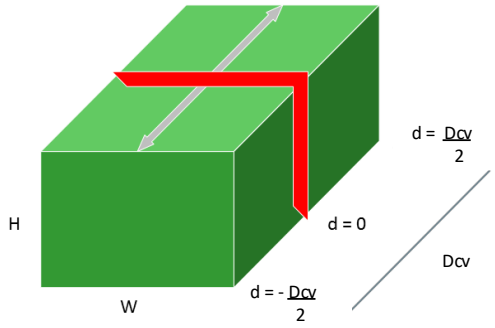



**Figure 2. Illustration of the proposed symmetric cost volume to allow for negative disparity hypotheses. It ensures that initially too high estimates can be corrected by the residual disparities on finer scales.**

range $(\frac{-D_{cv}}{2}, \frac{D_{cv}}{2})$ respectively, leaving out one disparity candidate at the negative search space if required due to rounding. It is worth noting here that after this change the effective search space for disparity is $\frac{D_{cv}}{2}$.

*3.2.2 Disparity Regression.* For calculating the final disparity map we use disparity regression as proposed by Kendall *et al.* [13]. This

yields a much more continuous and consistent disparity map than just using classification based operations. The disparity prediction $\hat{d}$ is given by:

$$\hat{d} = \sum_{d=\frac{-D_{cv}}{2}}^{\frac{D_{cv}}{2}} d \cdot \sigma(-C^A(d)) \tag{1}$$

Pseudo-probabilities for each disparity candidate $d$ are calculated from the filtered cost volume $C^A$ by applying the softmax operation $\sigma$. Note that the summation bounds in the equation above are consistent with the cost volume construction for negative disparity candidates.

## 3.3 iCFR: Iterative Coarse-to-Fine Refinement

As shown in Figure 1, the overall network architecture works in a coarse-to-fine refinement fashion using features at multiple scales from the feature extractor. The initial coarse disparity map $disp^i$ is calculated at $\frac{1}{48}$ of the original resolution, with the prediction head for the corresponding scale. This $disp^i$ is then up-sampled to the next finer scale *i.e.* $\frac{1}{24}$. Then the right image features at $\frac{1}{24}$th scale ($f_r^{i+1}$) are *warped* towards this up-sampled $disp^{i'}$. The idea is that these warped features $f_{r,warped}^{i+1}$ will only represent the remaining disparity to be calculated. These $f_{r,warped}^{i+1}$ are then passed to the prediction head for $\frac{1}{24}th$ scale. Since the target image features are now warped, the prediction head will only produce a residual of the full disparity map $res^{i+1}$. Finally the up-sampled coarse disparity map at $disp^{i'}$ is added to this residual map $res^{i+1}$ to get a refined disparity map at this scale, *i.e.* $disp^{i+1}$. This process is iterated for all scales *i.e.* $\{\frac{1}{12}, \frac{1}{6}, \frac{1}{3}\}$ to get the fine grained disparity map $disp^{i+4}$ at $\frac{1}{3}$ resolution. The $disp^{i+4}$ is then passed through the refinement network to get a final refined disparity map $disp^{max}$ at input resolution. A simplified pseudo-code of the iCFR is shown in Algorithm 1. Although the iterations at multiple resolutions introduce additional computational effort for calculating the final refined disparity, these computations follow a geometric series applying the sub-sampling factor (usually 2). This means the computational cost can only be twice w.r.t. the highest resolution for infinitely many sub-scales of iCFR. This is worth investing, considering the saved computations due to the smaller cost volumes at every scale.

**Algorithm 1** iCFR

```
[feat_l] = FeatureExtractor(left)
[feat_r] = FeatureExtractor(right)
for (f_l, f_r) in each scale do
    if smallest scale then
        disp = PredictionHead(f_l, f_r)
    else
        disp = upSample(disp)
        f_{r,warped} = Warp(f_r, disp)
        residual = PredictionHead(f_l, f_{r,warped})
        disp = disp + residual
    end if
end for
disp^max = Refinement(disp, left, right)
return disp^max
```

*3.3.1 Disparity Search Range: $D_{cv}$.* Due to the use of prediction heads at different scales and because each subsequent disparity map is up-sampled for the next scale, the disparity values are also scaled appropriately. This means the $D_{cv}$ for the cost volume can be considerably reduced leading to a much shallower cost volume. For example a $D_{cv}$ of 12 at scale $\frac{1}{24}$ is scaled to 96 at the scale $\frac{1}{3}$. The $D_{cv}$ can be chosen so that at the highest scale, this value can estimate a $D_{max}$ of typically 192 pixels, which is usually seen in the state-of-the-art models. For any prediction head, the number of scales $S$ and the $D_{cv}$ is to be chosen correctly. The $D_{max}$ is given by:

$$D_{max} = \sum_{i=0}^{S} \frac{D_{cv}}{2} \cdot \frac{1}{s_i} \tag{2}$$

where $s_i \in \{\frac{1}{48}, \frac{1}{24}, \frac{1}{12}, \frac{1}{6}, \frac{1}{3}\}$ in the presented case. The drastic effect of reduction in $D_{cv}$ on inference time is shown in Table 2.

## 3.4 Refinement Network

As discussed in Section 3.3, the $disp^{i+4}$ is produced at scale $\frac{1}{3}$. It typically has some edge gradient effects due to the multi-scale approach. These effects are especially visible on object boundaries and on areas where there is occlusion. To get rid of these artefacts, a final refinement block is applied. We use a stacked hourglass refinement layer as proposed by AANet+ [23]. We calculate the photometric consistency error [1] and use that as an additional input to the refinement network, along with the left and right images. The refinement network then hierarchically up-samples the prediction at $\frac{1}{3}$ to input resolution and produces geometrically consistent and continuous disparity maps covering thin areas.

## 3.5 Loss Function

The loss function that is typically used with the disparity regression layer is smooth $L1$ loss which we adopt in our pipeline too. This is because smooth $L1$ loss is quite robust against outliers and noise, and therefore helps against disparity discontinuities. We define our loss as:

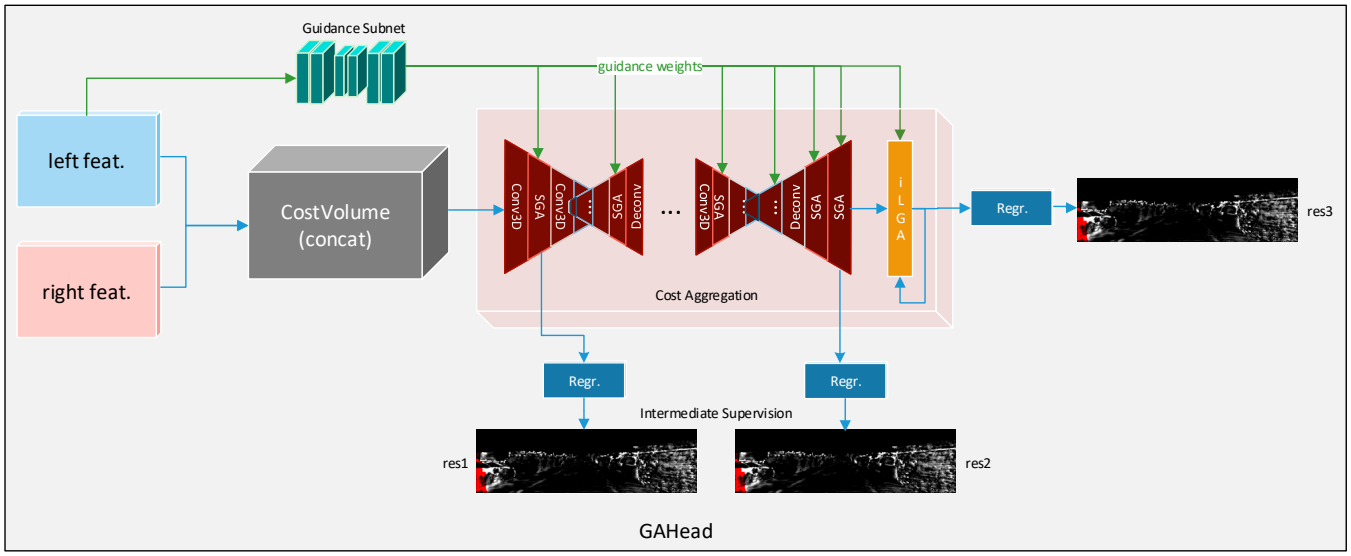
$$\hat{L}(\hat{d}, d) = \frac{1}{N} \sum_{i=1}^{N} L1(|\hat{d}_i - d_i|) \tag{3}$$

$$L1(x) = \begin{cases} x - 0.5, & if \quad x \geq 1 \\ x \cdot 0.5, & if \quad x < 1 \end{cases} \tag{4}$$

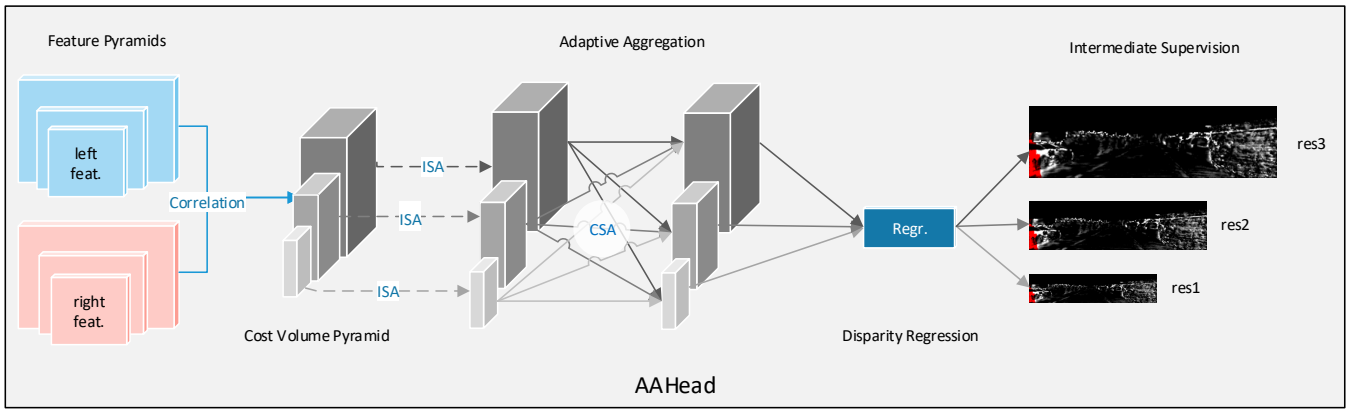
Here, $|\hat{d}_i - d_i|$ is the absolute error in the predicted disparities and $N$ is the number of ground truth label pixels. The final loss function is the weighted sum of losses over all predictions:

$$L = \sum_{i=1}^{M} \lambda_i \cdot \hat{L}_i \tag{5}$$

where $\lambda_i$ is a scalar weight and $M$ is the total number of predictions for intermediate supervision.




**(a) The prediction head adapted from GANet [29].**




**(b) The prediction head adapted from AANet [23].**

**Figure 3. A more detailed view of the prediction heads as used in our iCFR. They can optionally contain a guidance subnet as in GANet [29] (a) and support additional supervision with intermediate disparity regression. The inputs are the left and (warped) right image features at the corresponding scale from the feature extractor. These are used to build either a concatenation or correlation cost volume. The cost volume and the guidance weights are then passed to the cost aggregation or any kind of matching network. In our best performing network, ISA and CSA from AANet [23] (b) are used for aggregation. Finally, a softmax regression layer estimates the final disparity or the residual disparity based on the refined cost volume.**

## 3.6 GAHead

We apply the iCFR algorithm as described in Section 3.3 on state-of-the-art GANet [29] to significantly reduce its memory footprint and inference time. To this end we replace the prediction head in our network with the GANet matching network *i.e.* the prediction head now consists of concatenation based 4D symmetric cost-volume, guidance subnet, cost aggregation network (SGA and LGA layers) and disparity regression network. We call it *GAHead* and it is visualized in Figure 3a. The guidance weights are calculated by the multi-scale guidance subnet on top of the feature extraction network which are reshaped and normalized for use in cost aggregation.

*3.6.1 Improvements to the GAHead.* To push the GAHead performance even further we propose some improvements. Firstly, we replace a 3D convolution layer with an additional SGA layer, with additional guidance weights. This is because SGA layers are much faster than convolution layers and capture the disparity directly [10, 29]. Secondly, several LGA layers can be applied before and after passing it to the disparity regression layer to refine smaller details and fine structures in the final disparity map. However, instead of using multiple local guidance weights for each layer, we propose cost filtering using the same weights multiple times *i.e.* in an iterative manner. This is inspired by the idea of CSPN [4]. The filters are shared between iterations, thus keeping the number of learnable parameters constant. With each iteration, more details of

**Table 1. Evaluations of FRSNet with different design settings using the GAHead and the AAHead on the Sceneflow dataset. Evaluation metrics are average end point error (EPE) and 1 px threshold outlier rate (ER).**

| with GAHead | | | | | with AAHead | | | |
|---|---|---|---|---|---|---|---|---|
| Sym-CV | Improvements | HG-Ref | EPE [px] | ER [%] | Sym-CV | HG-Ref | EPE (px) | ER (%) |
| - | - | - | 7.06 | 51.9 | - | - | 4.91 | 43.1 |
| ✓ | - | - | 3.34 | 22.1 | ✓ | - | 2.08 | 23.2 |
| ✓ | ✓ | - | 1.01 | 18.2 | ✓ | ✓ | **0.98** | **16.9** |
| ✓ | ✓ | ✓ | **0.93** | **17.5** | | | | |

the image are revealed which in turn improve the per-pixel depth estimation results. The number of iterations is experimentally determined with the best EPE obtained for $i = 3$, without introducing too much overhead in run-time.

### 3.7 AAHead

We also apply the iCFR algorithm on state-of-the-art AANet [23] to reduce FLOPs and increase its scalability. To this end, we replace the prediction head with the adaptive aggregation module as described in AANet [23], which consists of intra/cross-scale aggregation operations. So the prediction head (AAHead, see Figure 3b), now consists of a correlation-based 3D symmetric cost volume, an adaptive aggregation network and disparity regression. Since the AAHead works with feature pyramids, for each scale of the iCFR algorithm a nested feature pyramid at $\{\frac{1}{3}, \frac{1}{6}, \frac{1}{12}\}$ sub-resolutions is constructed. The coarsest scale $\frac{1}{48}$ is omitted in this case because of the bottleneck in the adaptive aggregation module. To compensate for this reduction, the $D_{cv}$ is set to 24 according to Eq. 2 to keep the $D_{max}$ close to 192. The adaptive aggregation network employs several modulated deformable convolution layers to aggregate the cost at different scales which has a higher receptive field than normal convolution operation [31]. Reduction in FLOPs due to lower resolutions and a much shallower cost volume results in a scalable and much more efficient overall model with even better sub-pixel accuracy and pixel outlier rate than the original AANet as shown in Table 2.

## 4 EXPERIMENTS

We perform exhaustive experiments on our FRSNet using the Sceneflow and KITTI 2015 datasets. The first proposed by Mayer *et al.* [17], is a large dataset of synthetic images with dense ground truth. The KITTI dataset [8] contains real-world outdoor images but with sparse ground truth. We report end-point-error (EPE) and 1-pixel outlier rate (ER) on the Sceneflow dataset and EPE and D1 rate on KITTI datasets. The network is implemented in PyTorch. Our final model uses Adam ($\beta_1 = 0.9, \beta_2 = 0.999$) as optimizer. We employ a three-stage training strategy for all our experiments. 1) Training on Sceneflow dataset, 2) Fine-tuning on KITTI dataset, 3) Optionally, fine-tuning on the (dense) pseudo-ground truth as proposed in AANet [23]. This third step is required to produce visually consistent disparity maps for upper regions of KITTI images where there is no ground truth disparity available (*c.f.* Figure 4). We set the $D_{cv}$ value to 12 and 24 for experiments with GAHead and AAHead, respectively. Training is done for 30 epochs with a batch size of 32 on two NVIDIA V100 32GB GPUs, with the crop-size of $240 \times 576$ on Sceneflow dataset and for 1000 epochs with a batch size of 16

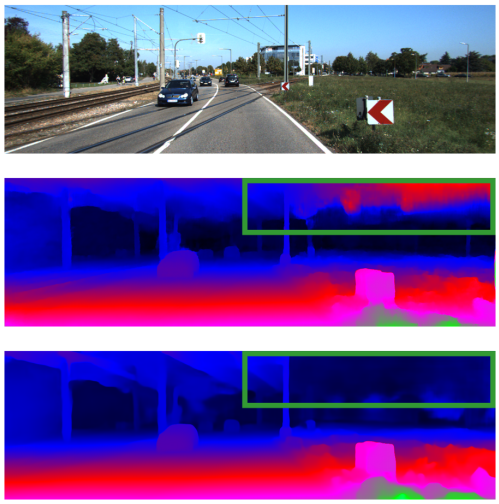


**Figure 4. Visual comparison of the results when training with (bottom) and without (middle) pseudo-GT. Using the pseudo-GT produces much more consistent results, *i.e.* without extraneous disparity values on the areas where there is no GT especially in the sky regions. The quantitative effect of this training strategy is shown in Table 3.**

on the KITTI datasets with the crop size of $384 \times 1248$. The third fine-tuning stage is carried out for a maximum of 8 epochs. The learning rate begins at 0.001 with a schedule for halving at 400th, 600th and 800th epoch in the fine-tuning phase. All the input image channels are normalized by subtracting their means and dividing their standard deviations. The loss weights $\lambda_i$ in Eq. 5 for multiple predictions are set to $[0.2, 0.4, 0.6, 1.0]$ for the three (two intermediate + final) predictions of the finest pyramid scale and the ultimate prediction at input resolution of the refinement network, in this order.

### 4.1 Ablation Study

To validate the performance of the proposed iCFR algorithm on the proposed FRSNet, we perform several experiments on the Sceneflow test set and on our KITTI validation set of 15 image pairs, which are obtained by splitting the training set for KITTI 2015 dataset randomly. The ablation experiments involve using the GAHead and AAHead in FRSNet and evaluating the effectiveness of different improvements proposed in Section 3. Table 1 shows the overall results of the ablation study for both predictions heads on the Sceneflow test set. In both cases, it highlights the importance of

**Table 2. Comparison of the state-of-the-art GANet [29] and AANet [23] to the proposed FRSNet with corresponding prediction heads. Our proposed network is much more scalable in terms of GPU memory, FLOPs and the number of parameters. The inference time is also considerably reduced *i.e.* 49× faster than the GANet, making it real-time, without a drastic increase in the EPE and D1. Compared to the AANet [23] it actually performs better by 0.05 px in EPE and 31% better D1 on our validation set for KITTI 2015. All values are the mean results of running these models for an inference of 100 image pairs, on a GTX 1080Ti. For each model, the comparison stops at the highest resolution that fits into GPU memory.**

| Method | Resolution | $D_{cv}$ | $D_{max}$ | Params | Mem | FLOPS | EPE [px] | D1 [%] | Time [ms] |
|---|---|---|---|---|---|---|---|---|---|
| GANet [29] | KITTI | 48 | 192 | 6.5M | 6.2G | 2.2T | 0.54 | 1.80 | 7402 |
| FRSNet-GA | KITTI | 4 | 186 | 6.5M | 1.4G | 373G | 0.74 | 2.80 | 150 |
| | HD | 4 | 186 | 6.5M | 5.7G | 718G | - | - | 452 |
| | 4K | 4 | 186 | 6.5M | 10G | 1.14T | - | - | 991 |
| AANet [23] | KITTI | 64 | 192 | 8.4M | 4.4G | 575G | 0.55 | 2.03 | 62 |
| | HD | 64 | 192 | 8.4M | 7.8G | 793G | - | - | 191 |
| FRSNet-AA | KITTI | 8 | 180 | 3.1M | 1.4G | 245G | 0.50 | 1.40 | 61 |
| | HD | 8 | 180 | 3.1M | 3.8G | 457G | - | - | 173 |
| | 4K | 8 | 180 | 3.1M | 9.9G | 981G | - | - | 692 |

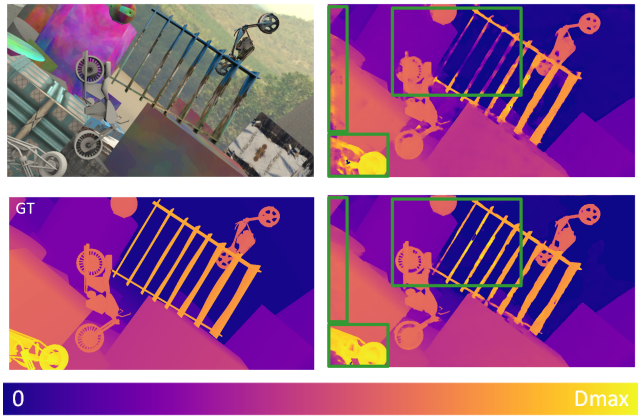



**Figure 5. Visual comparison on the Sceneflow test set, showing the impact of the refinement block. From top-left to bottom-right: Input reference images, prediction without refinement, GT disparity map, and refined prediction. Fine details are effectively recovered.**

having a symmetric cost volume (*Sym-CV*) with negative disparity hypotheses which decreases the EPE and ER by almost half. With GAHead, the introduction of the proposed improvements lower the error metrics even further. It was noted that replacing a higher resolution 3D convolution layer in cost aggregation with a SGA layer not only reduces the EPE and ER but also reduces the inference time by 70 ms.

Using the AAHead, without the final refinement layer yields relatively worse predictions than with GAHead. However, with a final hourglass refinement block (*HG-Ref*) as proposed in AANet+ [23] both variants of our model reach sub-pixel accuracy in EPE with the best ER. Similar improvements are seen while evaluating on the KITTI 2015 validation dataset. The positive impact of the refinement block is illustrated in Figure 5. A quantitative evaluation is given in Table 3 in which the hourglass refinement block (*HG-Ref*) is also compared against an alternative refinement module of SDRNet [1].

**Table 3. Evaluation of our FRSNet for different training datasets and different refinement modules. Evaluation metrics EPE and D1 rate calculated on our KITTI 2015 validation set.**

| KITTI 2015 | KITTI 2012 | Pseudo-GT | SDRNet [1]-Ref | HG-Ref | EPE (px) | D1 (%) |
|---|---|---|---|---|---|---|
| ✓ | - | - | - | - | 1.3 | 5.7 |
| ✓ | - | - | ✓ | - | 0.78 | 2.9 |
| ✓ | - | - | - | ✓ | 0.66 | 2.3 |
| - | ✓ | - | - | ✓ | 0.66 | 2.7 |
| ✓ | ✓ | - | ✓ | - | 0.56 | 2.0 |
| ✓ | ✓ | ✓ | ✓ | - | 0.54 | 1.7 |
| ✓ | ✓ | ✓ | - | ✓ | **0.50** | **1.4** |

## 4.2 Performance Comparison

Table 2 shows a comprehensive comparison between our network with GAHead and AAHead and the corresponding original networks on our KITTI 2015 validation set. There is a significant decrease in the number of parameters, memory consumption, FLOPs and inference time. Our model only requires 16% of the FLOPs required for the GANet-deep [29], while consuming 6× less memory. Similar improvement is seen compared to AANet [23] where the memory requirement is less than half while D1 being 31% better, with a comparable run-time. As also shown in Figure 6, our model can scale up to even close to 6K resolution while keeping the run-time below one second. These performance improvements are made possible because of the lower $D_{cv}$ value, which in turn means a shallower cost volume. Moreover, the iterative computations at smaller resolutions are bounded by at most twice the time required for the common operations at the highest resolution.

## 4.3 Benchmark Results

For benchmarking, we use our best performing network which is equipped with the AAHead and final hourglass refinement layer to evaluate on the KITTI 2015 test set on the official website. Table 4 shows the result of the evaluation in comparison to other models on the leader board. Compared to single-scale models our model

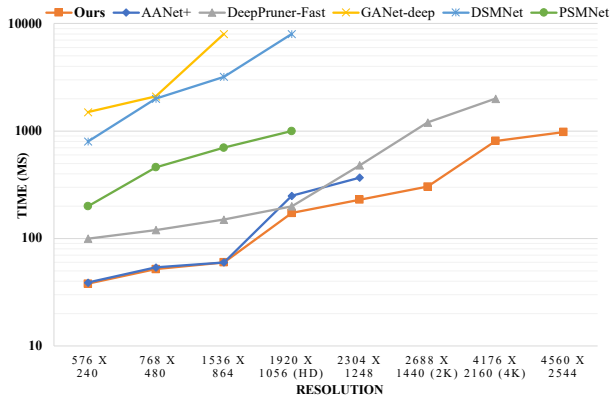



**Figure 6. Run-time of different stereo matching models for varying input resolutions. For all measurements, a GTX 1080Ti GPU (12 GB) is used. Each curve ends where the corresponding model can no longer fit into the GPU memory. Our FRSNet is around 120× and 15× faster than the GANet [29] and PSMNet [2], respectively. AANet+ [23] although having almost comparable run-time to ours, becomes too large to fit into the available GPU memory at just above HD resolution. DeepPruner [6], is 2.5× slower than our model, is less scalable in terms of memory footprint, and performs less accurately.**

**Table 4. Benchmark results of the D1 metric (in %) on the KITTI 2015 test set. Best numbers are given in bold, second-best are underlined. It is worth mentioning that the reported run-times are not measured in uniform settings, therefore a better comparison of run-time is provided by Table 2 and Figure 6.**

| Model | D1-Noc | | | D1-All | | | Time |
|---|---|---|---|---|---|---|---|
| | bg | fg | all | bg | fg | all | [ms] |
| GCNet [12] | 2.02 | 3.12 | 2.45 | 2.21 | 6.16 | 2.87 | 900 |
| PSMNet [2] | 1.71 | 4.31 | 2.14 | 1.86 | 4.62 | 2.32 | **410** |
| GANet [29] | **1.34** | 3.11 | **1.63** | **1.48** | 3.46 | **1.81** | 1800 |
| StereoNet [14] | - | - | - | 4.30 | 7.45 | 4.83 | **15** |
| DispNetC [17] | 4.11 | 3.72 | 4.05 | 4.32 | 4.41 | 4.34 | 60 |
| DeepPruner-fast [6] | 2.13 | 3.43 | 2.35 | 2.32 | 3.91 | 2.59 | 60 |
| AANet+ [23] | **1.49** | 3.66 | **1.85** | **1.65** | 3.96 | **2.03** | 60 |
| **FRSNet (Ours)** | 1.58 | **3.45** | 1.88 | 1.73 | **3.87** | 2.09 | 60 |

achieves comparable results while reducing the complexity and run-time significantly. Compared to real-time models with inference time less than 100 milliseconds, our model achieves state-of-the-art results standing right beside the AANet+ (the improved variant of the original AANet) in D1 metric while having a lower memory and computational footprint and better scalability at the same inference time. The results in Tables 4 and 5 for both the KITTI 2015 and Sceneflow datasets demonstrate that our method maintains a balance between accuracy and speed. The iCFR algorithm thus can be applied to any state-of-the-art stereo matching architecture to make it more efficient and scalable while retaining its accuracy.

Figure 7 shows a comparison of D1 error maps produced by the KITTI 2015 benchmark on the test set. Our best performing model exhibits lower errors overall, especially at occluded areas and image boundaries. The error maps reveal even smaller errors on smooth

**Table 5. Comparison of models on the Sceneflow dataset on our machine with a GTX 1080Ti GPU and using official code repositories. All the metrics are recorded after CUDA warm-up iterations, averaged on 100 inferences.**

| Models | EPE [px] | ≥ 3px [%] | Time [ms] |
|---|---|---|---|
| GCNet [12] | 2.51 | 9.34 | 950 |
| PSMNet [2] | 1.09 | **4.14** | **640** |
| GANet [29] | **0.78** | - | 7402 |
| StereoNet [14] | 1.10 | - | **15** |
| DispNetC [17] | 1.68 | 9.31 | 60 |
| DeepPruner [6] | 0.97 | - | 120 |
| AANet [23] | **0.87** | **3.44** | 62 |
| **FRSNet (Ours)** | 0.93 | 6.01 | 60 |

regions. The maps also show that our model produces state-of-the-art results and outperforms other real-time models with visually consistent and continuous disparity maps.

## 5 CONCLUSION

In this paper, a novel multi-scale stereo matching architecture is proposed using the iCFR algorithm with disparity residuals. We show that our architecture can be applied to any state-of-the-art model to boost its efficiency in terms of run-time, memory, and scalability. We validate this on two state-of-the-art networks namely GANet [29] and AANet [23]. Our best performing model performs 120x faster than the GANet and uses 5x less memory, using just 16% of FLOPs, while keeping the EPE and ER comparable. Our model scales until close to even 6K resolution on a GTX 1080Ti GPU, while keeping the inference time still below one second. The results show that by using iCFR, the trade-off between run-time and EPE can be optimized. Our best performing model produces state-of-the-art results on the Sceneflow test set as well as the KITTI 2015 benchmark while outperforming all the other real-time models in terms of accuracy.

Depending on the used prediction head and to keep the $D_{cv}$ low, the proposed architecture has a limitation on the number of scales (or the maximum down-sampling factor) to use. This is because of the relation between the cost aggregation technique, the $D_{max}$ and scales $s_i$ as shown in Eq. 2.

An interesting future work would be extending our architecture with already fast prediction heads to see how much further the performance improvement can be pushed. We also hope that the considerable reduction in the number of FLOPs will make our model suitable for edge computing devices with real-time performance requirements.

## ACKNOWLEDGMENTS

This work was partially funded by the Federal Ministry of Education and Research Germany under the project DECODE (01IW21001).

## REFERENCES

[1] Rohan Chabra, Julian Straub, Christopher Sweeney, Richard Newcombe, and Henry Fuchs. 2019. Stereodrnet: Dilated residual stereonet. In *Conference on Computer Vision and Pattern Recognition (CVPR).*

[2] Jia-Ren Chang and Yong-Sheng Chen. 2018. Pyramid stereo matching network. In *Conference on Computer Vision and Pattern Recognition (CVPR).*

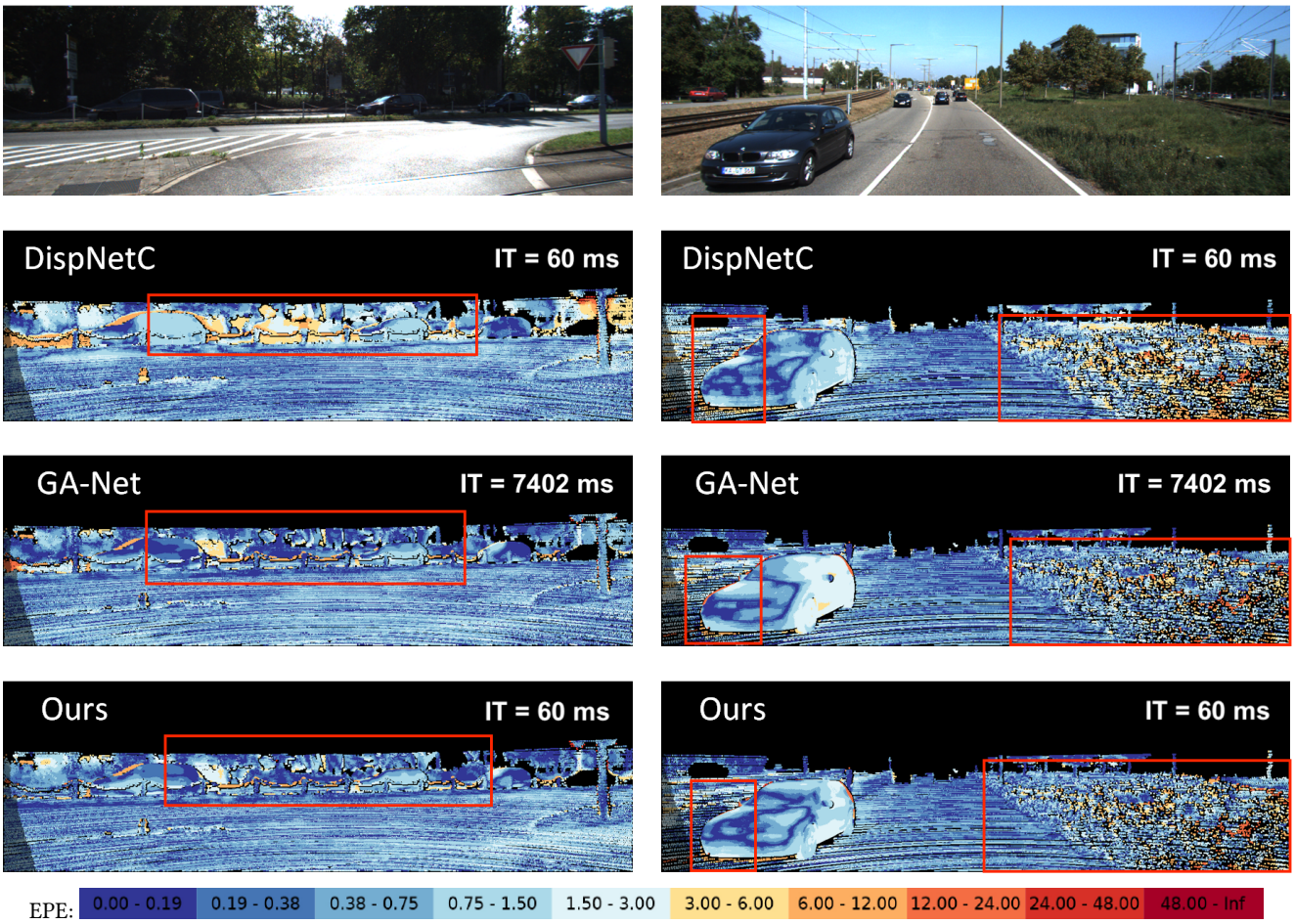



**Figure 7. Comparison of the D1 error maps on KITTI 2015 test set from top to bottom: DispNetC [17], GANet [29] and our FRSNet. Our model shows much smaller errors especially in occluded areas and at object boundaries as compared to the DispNetC. Compared to state-of-the-art GANet [29] our model has even lower errors in smooth areas while being significantly more efficient.**


[3] Rui Chen, Songfang Han, Jing Xu, and Hao Su. 2019. Point-based multi-view stereo network. In *International Conference on Computer Vision (ICCV)*.

[4] Xinjing Cheng, Peng Wang, and Ruigang Yang. 2019. Learning depth with convolutional spatial propagation network. *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2019).

[5] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. 2020. Hierarchical Neural Architecture Search for Deep Stereo Matching. *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[6] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. 2019. DeepPruner: Learning efficient stereo matching via differentiable patchmatch. In *International Conference on Computer Vision (ICCV)*.

[7] Lutz Falkenhagen. 1997. Hierarchical block-based disparity estimation considering neighbourhood constraints. In *International Workshop on SNHC and 3D Imaging*.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. 2020. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Heiko Hirschmuller. 2005. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[11] Michal Jancosek and Tomás Pajdla. 2009. *Segmentation based multi-view stereo.* Citeseer.

[12] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. 2017. End-to-end learning of geometry and context for deep stereo regression. In *International Conference on Computer Vision (ICCV)*.

[13] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. 2017. End-to-end learning of geometry and context for deep stereo regression. In *International Conference on Computer Vision (ICCV)*.

[14] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. 2018. StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *European Conference on Computer Vision (ECCV)*.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Yebin Liu, Xun Cao, Qionghai Dai, and Wenli Xu. 2009. Continuous depth estimation for multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. 2016. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Dongbo Min, Jiangbo Lu, and Minh N Do. 2011. A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy?. In *International Conference on Computer Vision (ICCV)*.

[19] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*.
[20] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. 2017. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *International Conference on Computer Vision Workshops (ICCVW)*.
[21] Johannes L Schonberger, Sudipta N Sinha, and Marc Pollefeys. 2018. Learning to fuse proposals from multiple scanline optimizations in semi-global matching. In *European Conference on Computer Vision (ECCV)*.
[22] Vladimir Tankovich, Christian Hane, Yinda Zhang, Adarsh Kowdle, Sean Fanello, and Sofien Bouaziz. 2021. HITNet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[23] Haofei Xu and Juyong Zhang. 2020. AANet: Adaptive aggregation network for efficient stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[24] Qingxiong Yang. 2012. A non-local cost aggregation method for stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[25] Zhichao Yin, Trevor Darrell, and Fisher Yu. 2019. Hierarchical discrete distribution decomposition for match density estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[26] Zehao Yu and Shenghua Gao. 2020. Fast-MVSnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[27] Jure Zbontar and Yann LeCun. 2015. Computing the stereo matching cost with a convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[28] Jure Zbontar and Yann LeCun. 2015. Computing the stereo matching cost with a convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[29] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. 2019. GA-Net: Guided aggregation net for end-to-end stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
[30] Xiaoxue Zhang and Zhigang Liu. 2014. A survey on stereo vision matching algorithms. In *World Congress on Intelligent Control and Automation (WCICA)*.
[31] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. 2019. Deformable convnets v2: More deformable, better results. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.