# DeepPruner: Learning Efficient Stereo Matching via Differentiable PatchMatch

Shivam Duggal[1], Shenlong Wang[1,2], Wei-Chiu Ma[1,3], Rui Hu[1] and Raquel Urtasun[1,2]
[1]Uber ATG, [2]University of Toronto, [3]Massachusetts Institute of Technology

## Abstract

*Our goal is to significantly speed up the runtime of current state-of-the-art stereo algorithms to enable real-time inference. Towards this goal, we developed a differentiable PatchMatch module that allows us to discard most disparities without requiring full cost volume evaluation. We then exploit this representation to learn which range to prune for each pixel. By progressively reducing the search space and effectively propagating such information, we are able to efficiently compute the cost volume for high likelihood hypotheses and achieve savings in both memory and computation. Finally, an image guided refinement module is exploited to further improve the performance. Since all our components are differentiable, the full network can be trained end-to-end. Our experiments show that our method achieves competitive results on KITTI and SceneFlow datasets while running in real-time at 62ms.*

## 1. Introduction

Stereo estimation is the process of estimating depth (or disparity) from a pair of images with overlapping fields of view. It is a fundamental building block for many applications such as robotics and computational photography. Despite many decades of research, stereo estimation of real-world scenes remains an open problem. State-of-the-art approaches still have difficulties tackling repetitive structures, texture-less regions, occlusions and thin objects. Furthermore, runtime is also a challenge. While real-time inference is required for many applications, it is hard to achieve without a significant compromise on accuracy.

Classic stereo approaches typically start by computing robust feature representations [38, 36, 31, 20, 41]. A cost volume is then computed for each pixel, encoding the similarity between its representation and the representation of all pixels along the corresponding epipolar line on the other image. Post-processing techniques [14, 30] are typically exploited for smoothing and noise removal. The lowest-cost disparity value is then picked for each pixel as the final prediction. Yet this pipeline is usually computationally demanding, due to the size of the solution space and the use of sophisticated post-processing procedures. A plethora of acceleration approaches have been proposed toward achieving real-time performance, such as solution space pruning [6], coarse-to-fine cost volume computation [16] and efficient variational optimization [10]. Among them, PatchMatch [3] is one of the most popular techniques used to achieve competitive performance with real-time inference [5, 6].

More recently, with the flourishing of deep learning, networks inspired by traditional stereo matching pipelines have been designed. They achieve state-of-the-art results by learning deep representations [22, 23] and building deep cost volumes [15, 8, 16]. While these approaches can be accelerated with GPUs, memory consumption and computation requirement is still a concern for most architectures. An alternative is to design a regression network that takes a stereo pair as input and directly regresses disparity without explicit matching or cost volume construction. Unfortunately, there is typically a large performance gap between real-time methods and best-performing algorithms.

Our goal is to significantly speed up the runtime of current state-of-the-art deep stereo algorithms to enable real-time inference. We build our model based on two key observations: first, the search space for stereo matching is large, yet a lot of candidates can be confidently discarded without requiring full evaluation; second, due to the coherent nature of the world, adjacent pixels often possess similar disparities. This suggests that once we know the disparity of one pixel, we can effectively propagate such information to its neighbors.

With these intuitions in mind, we propose DeepPruner, a real-time stereo matching model. Specifically, we first leverage a novel differentiable PatchMatch algorithm to obtain a sparse representation of the cost volume. We then exploit this representation to learn which range to prune for each pixel. Finally, an image guided refinement module is exploited to further improve the performance. Since all components are differentiable, the full network can be trained in an end-to-end fashion. By progressively reducing the search space and effectively propagating such information, we are able to efficiently compute the cost volume for high likelihood hypotheses and significantly reduces both memory consumption and computational cost.

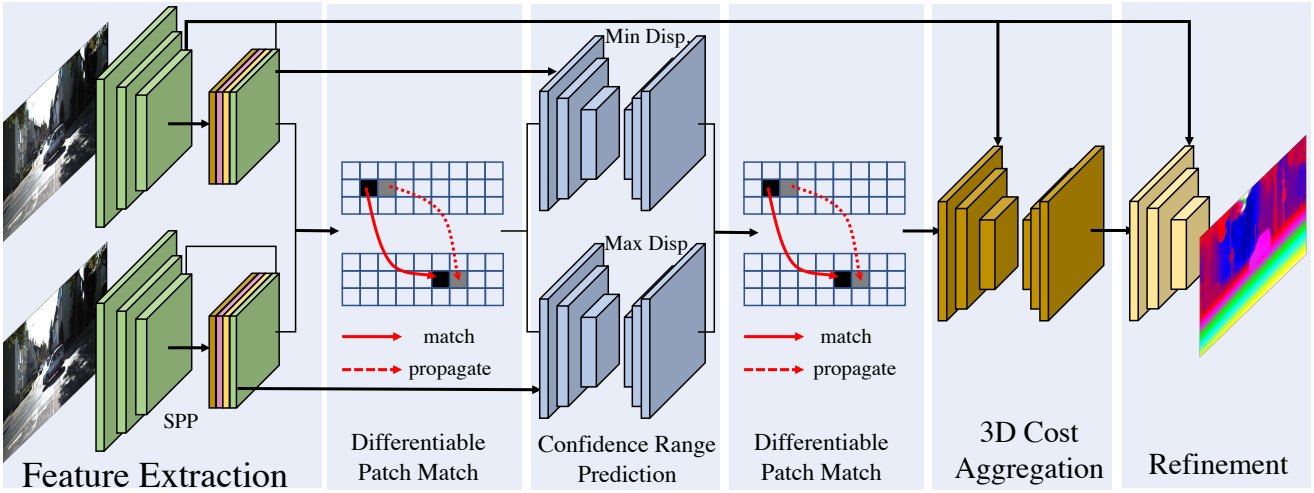


Figure 1: **Overview:** Given a pair of stereo images, we first extract deep multi-scale features. Then we exploit differentiable PatchMatch to estimate a small subset of disparities for each pixel and capitalize on confidence range predictor to further prune out the solution space. Unlike other approaches [8, 15] which operate on the entire disparity search range, we only aggregate the cost within the reduced search range. Finally, we leverage a light-weight network to refine the stereo output.

We demonstrate the efficiency and effectiveness of our approach on the challenging SceneFlow [23] and KITTI [11] datasets. Our model ranks second on SceneFlow while being 8 times faster than the best method [9]. Comparing to previous approaches on KITTI, DeepPruner achieves competitive performance to state-of-the-art methods [8, 33] and ranks first among all real-time models [23, 32]. To further showcase the robustness and generalizability of DeepPruner, we evaluate it on the Robust Vision Stereo Challenge [1]. Our model achieves state-of-the-art results on multiple datasets [11, 28, 26] and obtain first place on the overall ranking.

## 2. Related work

**Classical Stereo Matching:** Estimating disparities (depth) from stereo images has been studied for decades [2]. A stereo algorithm typically consists of the following three steps [27]: compute a pixel-wise feature representation, construct the cost volume, and final post-processing. As the pixel representation plays a critical role in the process, researchers have exploited a variety of representations, from the simplest RGB values of the surrounding pixels to more discriminative local descriptors such as CENSUS [39], SIFT [19], and BRIEF [7]. Together with the carefully designed post-processing techniques, *e.g.* cost aggregation, semi-global matching [14], and Markov random fields [30, 34], they are able to achieve good performance on relatively simple scenarios.

**Deep Stereo Matching:** In order to further deal with more complex real world scenes, especially texture-less regions or reflective surfaces, modern approaches leverage CNNs to extract robust features [22] and conduct matching [40, 41]. While these techniques have demonstrated great performance in benchmarks [11], time-consuming post-processing is still required. With this in mind, researchers propose to directly regress sub-pixel disparities from the given stereo images [35, 9, 29]. By implementing the full traditional stereo pipeline as neural network layers, these models can be trained in an end-to-end fashion and are able to conduct inference completely on GPU, which drastically improves the efficiency. Unfortunately, due to the large size of the cost volume and aggregation, the memory consumption and the required computation are still very high, rendering the models impractical [15, 8]. In this paper, we build our work upon [15, 8]. Instead of searching for the full disparity space, we exploit a novel differentiable version of PatchMatch to learn to prune out unlikely matches and reduce the complexity of the search space. Our model is able to run in real-time while maintaining comparable performance.

**PatchMatch:** The seminal work of PatchMatch (PM) was proposed by Barnes *et al*. in 2009 [3]. It was originally introduced as an efficient way to find dense correspondences across images for structural editing. The key idea behind it is that, a large number of random samples often lead to good guesses. Additionally, neighboring pixels usually have coherent matches. Therefore, once a good match is found, we

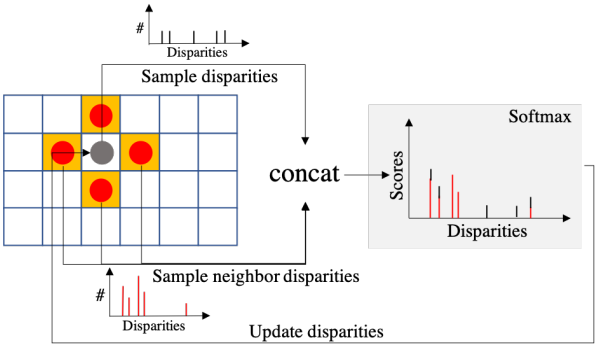


Figure 2: **Illustration of the differentiable patch match operations.**

can efficiently propagate the information to the neighbors. Due to its effectiveness in pruning out the search space, PatchMatch has drawn wide attention across the community [17, 12, 4, 5] and has been extended and applied in multiple domains. For instance, Korman and Avidan [17] incorporate the idea of image coherence into Locality Sensitivity Hashing and significantly improve the speed. He and Sun [12] combine KD-trees with PatchMatch to perform more efficient neighbor matching. PatchMatch has also been applied in the stereo setting for fast correspondence estimation [21] and slanted plane fitting [6]. To improve sub-pixel accuracy, Besse *et al*. [5] further combines PatchMatch with particle belief propagation and extend it to a continuous MRF inference algorithm. Specifically, [5] exploits PatchMatch to overcome the infeasibility of searching over continuous output space. Note that if the MRF has only unary terms, it reduces to a k-particle generalized PatchMatch [4].

Our pruning module is largely inspired by [5]. We first implement the particle PatchMatch operations as neural network layers and unroll them in a recurrent fashion. We then predict the disparity confidence range as to approximate the marginal distribution of each pixel. Through efficient sampling and propagation, we are able to prune out the solution space effectively and significantly speed up inference. Importantly, all our operations are differentiable, and thus our approach can be learned end-to-end.

**Real-time Stereo:** Besides our work, there has been several concurrent efforts pushing towards real-time deep learning based stereo estimation [16, 37]. Our work is different from theirs since we adaptively prune out the search space for each region. In contrast, they employ a fixed, coarse-to-fine procedure to iteratively find the match.

## 3. Learning to Prune for Stereo Matching

Our aim is to design an efficient stereo algorithm that not only produces reliable and accurate estimations, but also runs in real-time. Towards this goal, we present a simple yet effective solution that combines deep learning with PatchMatch to prune out the potentially large search space and significantly speeds up inference.

We start our discussion by describing the feature extraction backbone. We then briefly review the PatchMatch algorithm and show that it can be naturally incorporated into neural networks to prune out the search space and speed up cost volume construction. Finally, we describe how to aggregate the cost, refine the estimation, and perform end-to-end learning. We refer the reader to Fig. 1 for an illustration of our approach.

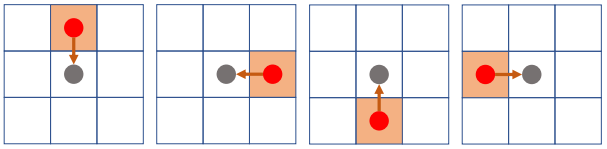

Figure 3: **One hot filter banks within the propagation layer.**

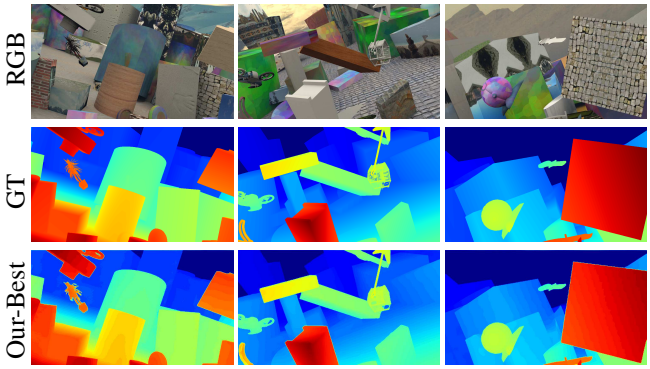


Figure 4: **Qualitative results on SceneFlow dataset.**

### 3.1. Feature Extraction

The goal of the feature extraction network is to produce a reliable pixel-wise feature representation from the input image. More formally, given a pair of stereo images $\{\mathbf{x}_0, \mathbf{x}_1\}$, we seek to learn a set of deep features $\mathbf{f}_0, \mathbf{f}_1$ that is useful for matching. Towards this goal, following [15, 8], we exploit a 2D convolutional neural network with a spatial pyramid pooling module [13, 42] as our backbone. Specifically, we employ four residual blocks and use $\times 2$ dilated convolution for the last block to enlarge the receptive field. We then apply spatial pyramid pooling to build a 4-level pyramid feature. Through multi-scale information, the model is able to capture large context while maintaining a high spatial resolution. The size of the final feature map is 1/4 of the original input image size. We share the parameters for the left and right feature network. Now that we have a reliable feature

representation for each pixel, the next step is to construct the cost volume.

### 3.2. Pruning Through Differentiable PatchMatch

Modern stereo approaches typically generate a cost volume over the full disparity space [8, 15, 9, 29]. The large search space not only increases memory consumption, but also intensifies the computational burden. For example, considering PSM-Net [8], 3D cost volume construction and aggregation takes more than 250 ms. These two operations themselves render the real-time applications infeasible. In this paper, we tackle this issue by designing an efficient PatchMatch-based pruner module that is able to predict a confidence range for each pixel, and construct a sparse cost volume that requires significantly less operations. This allows the model to focus only on high likelihood regions and save large amount of computation and memory. Unlike standard PatchMatch, our module is differentiable, making end-to-end learning possible. Importantly, as shown in our experiments, such confidence range is also a promising indicator of uncertainty and a foreseer of potential prediction error. This is important when depth estimates are used in downstream tasks.

**PatchMatch revisited:** Our pruner module is motivated by the elegant and classic PatchMatch algorithm [3]. PatchMatch methods [6, 5, 3, 4] typically consist of the following three steps[1]:

1. **Particle sampling**: generate $k$ random candidates;
2. **Propagation**: propagate particles to neighbors;
3. **Evaluation**: update best $k$ solutions by evaluating current and propagated particles;

Once the initialization is done (step 1,), the greedy approach iterates between step 2 and step 3 until convergence or a fix number of steps is reached. In practice, this often leads to good results without enumerating a,ll possibilities. Originally, $k$ was set to 1 [3]. But later on, generalized PatchMatch [4] draw connections to particle sampling methods and extend PatchMatch to utilize the top-$k$. This not only increases the expressive power but also enables faster convergence.

**Differentiable PatchMatch:** In this work, we unroll generalized PatchMatch as a recurrent neural network, where each unrolling step is equivalent to each iteration of the algorithm. This is important as it allow us to train our full model end-to-end. Specifically, we design the following layers:

1. **Particle sampling layer**: for each pixel $i$, we randomly generate $k$ disparity values from the uniform distribution over predicted/pre-defined search space;
2. **Propagation layer**: particles from adjacent pixels are propagated together through convolution with a predefined one-hot filter pattern (see Fig. 3), which encodes the fact that we allow each pixel to propagate particles to its 4-neighbours.
3. **Evaluation layer**: for each pixel $i$, matching scores are computed by taking the inner product between the left feature and the right feature: $s_{i,j} = \langle \mathbf{f}_0(i), \mathbf{f}_1(i + d_{i,j}) \rangle$ for all candidates $j$. The best $k$ disparity value for each pixel is carried towards the next iteration.

Our architecture design has one particle sampling layer at the bottom, and then iterates through propagation and evaluation layers recurrently. As the $\arg\max$ operator during evaluation is not differentiable, we replace it with a soft version [15]:

$$\hat{d}_i = \frac{\sum_j s_{i,j} \cdot d_{i,j}}{\sum_j s_{i,j}}. \quad (1)$$

Fig. 2 depicts the computation graph of one recurrent step, which combines propagation and evaluation. In practice, rather than allowing each particle to reside in the full disparity space, we divide the search space into $k$ intervals, and force the $i-$th particle to be in a $i-$th interval. This guarantees the diversity of the particles and helps improve accuracy for later computations, which we show in our experiments. Since all operations are differentiable, we can directly back-propagate through all unrolling steps and train the model in an end-to-end fashion.

**Confidence range prediction:** The original search space for all pixels is identical. However, in practice, for each pixel, the highly probable disparities lie in a narrow region. Using the small subset of disparities estimated from the PatchMatch stage, we have sufficient information to predict the range in which the true disparity lies. We thus exploit a confidence range prediction network to adjust the search space for each pixel. The network has a convolutional encoder-decoder structure. It takes the sparse disparity estimations from the differentiable PatchMatch, the left image and the warped right image (warped according to the sparse disparity estimations) as input and outputs a confidence range $\mathcal{R}_i = [l_i, u_i]$ for each pixel $i$.

The confidence range prunes out the space of unlikely matches, allowing the expensive cost-volume construction and aggregation to happen only at a few disparity values.

### 3.3. Cost Aggregation and Refinement

**Cost aggregation:** Based on the predicted range in the pruning module, we build the 3D cost volume estimator and

[1]For simplicity, we omit the local random resampling applied to the current particles.

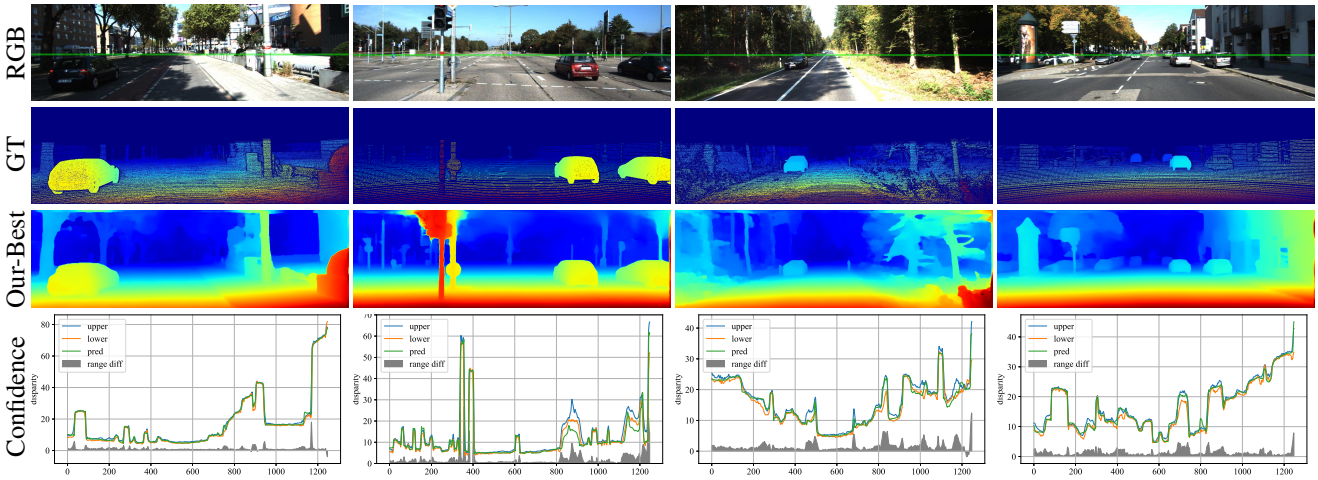


Figure 5: **Visualization of confidence range predictor.** At the bottom row, we show the predicted search range for pixels along the horizontal green line on validation images shown at the top row. Blue and orange line represent the upper and lower bound of the search range, respectively, and the grey shades depict the range size. For most pixels, DeepPruner predicts a very small search range, thereby allowing efficient cost aggregation. Higher search ranges often happen at boundary pixels, or pixels that are occluded in one view.

conduct spatial aggregation. Following common practice [8, 15], we take the left image, the warped right image and corresponding disparities as input and output the cost over the disparity range at the size $B \times R \times H \times W$, where $R$ is the number of disparities per pixel. Compared to prior work [8, 15], our $R$ is more than 10 times smaller, making this module very efficient. Soft-$\arg\max$ defined in Eq. 1 is again used to predict the disparity value $\mathbf{y}_{\text{cost}}$, so that our approach is end-to-end trainable.

**Refinement:** We utilize a lightweight fully convolutional refinement network to further boost the performance. The network takes left image convolutional features from the second residual block of the feature network and the current disparity estimation $\mathbf{y}_{\text{cost}}$ as input. It then outputs the finetuned disparity prediction $\mathbf{y}_{\text{refine}}$. The low-level feature information serves as a guidance to reduce noise and improve the quality of the final disparity map, especially on sharp boundaries.

### 3.4. End-to-end Learning

Our network is end-to-end differentiable. We use back-propagation to learn the parameters. Given the GT disparity $\mathbf{y}$, the total loss function is defined as follows:

$$\ell_s(\mathbf{y}_{\text{cost}} - \mathbf{y}_{\text{gt}}) + \ell_s(\mathbf{y}_{\text{refine}} - \mathbf{y}_{\text{gt}})$$

$$+\gamma\{\ell_{\text{lower}}(\mathbf{l} - \mathbf{y}_{\text{gt}}) + \ell_{\text{upper}}(\mathbf{u} - \mathbf{y}_{\text{gt}})\}$$

where the standard smooth-$\ell_1$ loss is applied over the disparity prediction in cost aggregation stage and final refinement stage respectively. Thus we define:

$$\ell_s(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

This loss has the advantage of being differentiable everywhere like the $\ell_2$ loss, but more robust to outlier like the $\ell_1$ loss. Moreover, the loss over upper and lower bound of the range is defined as a boomerang shape unbalanced smooth-$\ell_1$ loss:

$$\ell_{\text{lower}}(x) = \begin{cases} (1-\lambda)\ell_s(x) & \text{if } x > 0 \\ \lambda\ell_s(x) & \text{otherwise} \end{cases}$$

$$\ell_{\text{upper}}(x) = \begin{cases} \lambda\ell_s(x) & \text{if } x > 0 \\ (1-\lambda)\ell_s(x) & \text{otherwise} \end{cases}$$

with $0 < \lambda < 0.5$. $\gamma$ is the balancing scalar. Note that $\ell_{\text{upper}}$ encourages the upper-bound prediction to be closer to but preferably larger than GT disparity; whereas the $\ell_{\text{lower}}$ pushes the lower-bound prediction to be closer to but preferably smaller than GT disparity.

## 4. Experiments

We compare our approach against the best performing algorithms [41, 15, 9, 25, 29, 8] and the real-time models [32, 23]. Specifically, we evaluate two variants of our method, namely `DeepPruner-Best` and `DeepPruner-Fast`. `DeepPruner-Best` downsamples the cost volume by 4 times while `DeepPruner-Fast` downsamples it by 8. The rest of the two models remain the same.

| | GC-Net [15] | SegStereo [35] | CRL [25] | PDS-Net [33] | PSM-Net [8] | CSPN [9] | Our-Best | DispNetC [23] | Our-Fast |
|---|---|---|---|---|---|---|---|---|---|
| EPE | 2.51 | 1.45 | 1.32 | 1.12 | 1.09 | **0.78** | 0.86 | 1.68 | **0.97** |
| Runtime | 900 ms | 600 ms | 470 ms | 500 ms | 410 ms | 500 ms | **182 ms** | **60 ms** | 62 ms |

Table 1: **Quantitative results on SceneFlow Dataset.** Our approach ranks $2^{nd}$ among all competing algorithms. Our fast model is 8 times faster than prior art and improves the performance of previous real-time model by $40\%$.

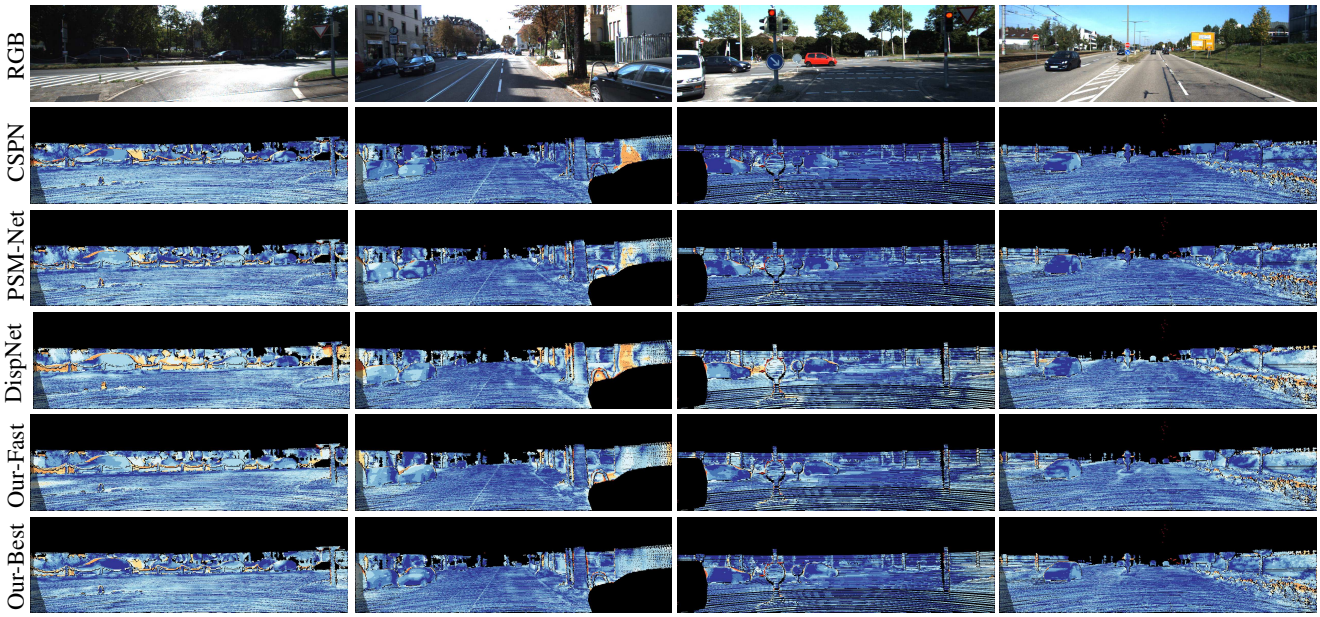


Figure 6: **Qualitative Results on KITTI 2015 Test Set.** Orange corresponds to erroneous prediction.

| | Inference | Noc (%) | | | All (%) | | |
|---|---|---|---|---|---|---|---|
| Methods | Runtime | *bg* | *fg* | *all* | *bg* | *fg* | *all* |
| Content-CNN [22] | 1000 ms | 3.32 | 7.44 | 4.00 | 3.73 | 8.58 | 4.54 |
| MC-CNN [41] | 67000 ms | 2.48 | 7.64 | 3.33 | 2.89 | 8.88 | 3.89 |
| GC-Net [15] | 900 ms | 2.02 | 3.12 | 2.45 | 2.21 | 6.16 | 2.87 |
| CRL [25] | 470 ms | 2.32 | 3.68 | 2.36 | 2.48 | 3.59 | 2.67 |
| PDS-Net [33] | 500 ms | 2.09 | 3.68 | 2.36 | 2.29 | 4.05 | 2.58 |
| PSM-Net [8] | 410 ms | 1.71 | 4.31 | 2.14 | 1.86 | 4.62 | 2.32 |
| SegStereo [35] | 600 ms | 1.76 | 3.70 | 2.08 | 1.88 | 4.07 | 2.25 |
| EdgeStereo [29] | 700 ms | 1.72 | 3.41 | 2.00 | 1.87 | 3.61 | 2.16 |
| CSPN [9] | 500 ms | **1.40** | **2.67** | **1.61** | **1.51** | **2.88** | **1.74** |
| DeepPruner-Best | **182 ms** | 1.71 | 3.18 | 1.95 | 1.87 | 3.56 | 2.15 |
| MAD-Net [32] | **20 ms** | 3.45 | 8.41 | 4.27 | 3.75 | 9.2 | 4.66 |
| DipsNetC [23] | 60 ms | 4.11 | 3.72 | 4.05 | 4.32 | 4.41 | 4.34 |
| DeepPruner-Fast | 61 ms | **2.13** | **3.43** | **2.35** | **2.32** | **3.91** | **2.59** |

Table 2: **Quantitative Results on KITTI 2015 Test Set. Top:** Our model acheives comparable performance to state-of-the-art models while being significantly faster. **Bottom:** Comparing to other real-time methods (*e.g.* DispNet), our stereo estimation is much more precise.

In this section, we first describe our experimental setup. Next we evaluate our approach on challenging public benchmarks. Finally we comprehensively study the characteristic of our model.

### 4.1. Dataset

**SceneFlow:** As proposed in [23], this is a synthetic dataset consisting of dense ground truth disparity maps for 35454 training and 4370 testing stereo pairs with dimensions as (H=540,W=960). We used End-Point-Error (EPE) as the evaluation metric for SceneFlow dataset.

**KITTI 2015:** This is a real world dataset consisting of 200 training and 200 testing stereo pairs with dimensions (H=376,W=1240). The ground truth disparities are obtained from Lidar points collected using Velodyne HDL-64E laserscanner. Unlike the prior version KITTI 2012, dense ground disparities are present for the dynamic scenes. The evaluation metrics used are same as that provided by the benchmark i.e percentage of outliers.

### 4.2. Implementation Details

The model was trained end-to-end on randomly cropped image patches of size (H=256,W=512) using Adam($\beta_1$=0.9, $\beta_2$=0.999) as the optimizer. Color normalization was done on the input images using ImageNet statistics(mean and std) as the sole pre-processing step. All models were trained on 4 Nvidia-TitanXp GPUs. For all datasets, we use the hyper-parameter $\lambda = 0.315$, $\gamma = 2.4$ in our loss function. We

| Module | Feature Extraction | PatchMatch-1 | Confident Range | PatchMatch-2 | Cost Aggregation | RefineNet |
|---|---|---|---|---|---|---|
| DeepPruner-Best | 54 ms | 20 ms | 61 ms | 13 ms | 32 ms | 3 ms |
| DeepPruner-Fast | 28 ms | 5 ms | 16 ms | 3 ms | 8 ms | 4 ms |

Table 3: **Runtime Breakdown:** As PatchMatch and Confidence Range Predictor significantly reduce the search space, we only conduct cost aggregation on a small subset of disparities, thereby being much faster.

| Network Component | | | | | | Inference | KITTI 2015 (%) | | | SceneFlow |
|---|---|---|---|---|---|---|---|---|---|---|
| Feat. Extr. | PM-1 | CRP | PM-2* | CA | RefineNet | Runtime | *bg* | *fg* | *all* | EPE |
| ✓ | ✓ | | | ✓ | ✓ | 120 ms | 2.05 | 3.57 | 2.28 | 0.982 |
| ✓ | ✓ | ✓ | | ✓ | ✓ | 172 ms | 1.65 | 3.27 | 1.90 | 0.868 |
| ✓ | ✓ | ✓ | ✓ | ✓ | | 178 ms | 2.04 | 3.86 | 2.32 | 1.283 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 182 ms | 1.61 | 2.90 | 1.8 | 0.858 |

Table 4: **Contributions of each network component:** Confidence range predictor (CRP) and PatchMatch (PM) significantly prune out the solution space, thus enable better cost aggregation (CA). RefineNet further improves the estimation by incoporating more visual guidance. (*: PM-2 is only used during SceneFlow pre-training.)

only computed loss over pixels with ground truth disparities between 0 and 192 during training. Evaluation is conducted over the all pixels regardless of their disparity values.

For sceneflow dataset, we trained the model from scratch with a batch size of 16 for 64 epochs. The initial learning rate was set to 0.001 and was decayed by 0.0003 after every 20 epochs.

For KITTI dataset, we combined KITTI 2012 and KITTI 2015 image pairs, resulting in a total of 394 training image pairs. We reserved out 40 images from the total 394 images for validation. We then used the pre-trained Sceneflow model and finetuned it for another 1040 epochs. After 800 epochs, all batch norm layers were switched to eval mode, i.e the running mean and std statistics were kept fixed for further training. We used a batch size of 16 for `DeepPruner-Best` model, while `DeepPruner-Fast` model was trained with a batch size of 64. We used an initial lr of 0.0001 and reduced it once, after 500 epochs to 0.00005. For submission to the KITTI test benchmark, we re-trained the model on all the 394 training images for 1040 epochs.

### 4.3. Experimental Results

**SceneFlow:** As shown in Tab. 1, our method outperforms most approaches by a large margin and achieves the second best results. Comparing to state of the art [9], our best model is 2.5 times faster and our fast model is over 8 times faster. Comparing against the real-time approach [23], our method reduces the end-point-error by $40\%$ with almost the same runtime. Fig. 4 depicts the qualitative results. DeepPruner captures both large disparities and small objects, and is able to produce sharp estimation on image boundaries.

**KITTI:** Tab. 2 showcases the performance and runtime of all competing algorithms on KITTI stereo benchmark. `DeepPruner-Best` achieves comparable performance to state-of-the-art approaches while being significantly faster. Comparing to real-time stereo models, `DeepPruner-Fast` reduces the outlier ratio by more than $40\%$. Fig. 6 visualize a few stereo results on test set. DeepPruner produces competitive estimation among various scenarios.

### 4.4. Analysis

**Ablation study:** To understand the effectiveness of each component in DeepPruner, we evaluate our model with different configurations. As shown in Tab. 4, confidence range predictor is of crucial importance to our model. With the help of refinement network, we can further capture the sharp edges as well as the fine-grained details, and improve the overall stereo estimation.

**Visualizing confidence range predictor:** The goal of confidence range predictor is to prune out the space of unlikely matches and ensure the expensive cost volume operations only happen at a few disparity values. To understand the efficacy of the predictor, we visualize the predicted search range as well as the GT disparity for pixels along a horizontal line. As shown in Fig. 5, our confident range is quite small in most cases, which greatly reduce the computation and memory burden of cost volume prediction.

**Uncertainty:** The range prediction can also be considered as a measurement of confidence/uncertainty level — the larger the range, the more uncertain the model is. To validate this hypothesis, we compare the predicted confidence range (*i.e.,* max range minus min range) against the disparity error map over several validation images. As shown in Fig. 7, the uncertainty (predicted range) map and the error

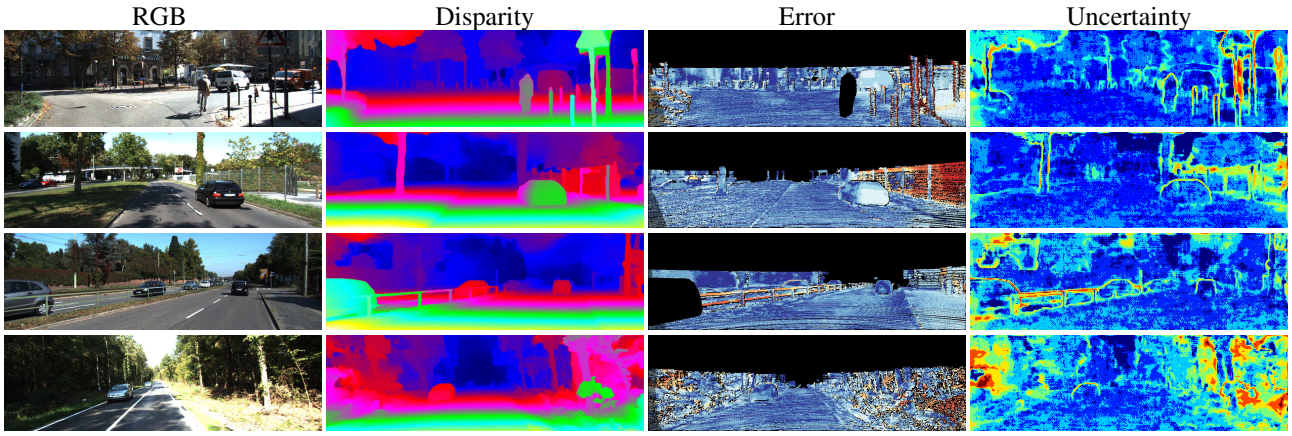


Figure 7: **Uncertainty vs Error.** As can be seen, uncertainty correlates highly with the error, thereby being a good indicator of the potential errors.

map are highly correlated, suggesting that it could be a good indicator of the potential errors. To further verify this, we track the change in the metric by gradually removing uncertain pixels, starting from most uncertain ones. By removing 6% of uncertain pixels, we improve the outlier ratio by 38%. This clearly indicates that our high confidence regions have very low error whereas most of the error happens at low confidence regions.

| Method | KITTI2015 | | MiddleburyV3 | | ETH3D | | ROB |
|---|---|---|---|---|---|---|---|
| | Rank | D1-all (3px) | Rank | 4x | Rank | 4x | Overall-Rank |
| PSMNet_ROB [8] | 2 | 2.31 | 11 | 29.2 | 6 | 0.54 | 5 |
| DN-CSS_ROB [1] | 8 | 2.94 | 3 | **19.6** | 2 | 0.38 | 3 |
| iResNet_ROB [18] | 3 | 2.71 | 6 | 22.1 | 3 | 0.40 | 2 |
| DeepPruner_ROB | **1** | **2.23** | 7 | 21.9 | **1** | **0.34** | **1** |

Table 5: **Comparison against top 3 approaches on Robust Vision Challenge.** Our proposed approach achieves the highest overall rank.

**Robustness and generalizability:** To corroborate how well our model generalizes across different scenarios, we evalute our model on the Robust Vision Challenge [1]. Specifically, we fine-tune our sceneflow-pretrained model on KITTI [24], ETH3D[28], and MiddleburyV3 [26] jointly and report results on all three datasets. As shown in Tab. 5, DeepPruner achieves the highest rank on two datasets and ranks first on the overall ranking. It is also able to capture the fine-grained geometry of various scenes (see Fig. 8).

**Runtime and memory analysis:** We benchmark the runtime of each component in the model during inference in Tab. 3. As PatchMatch and confidence range predictor progressively reduce the possible solution space, we only need to perform cost aggregation among a small subset of disparities. The model is thus significantly faster. To further demonstrate the efficiency of our model, we compare our memory consumption against previous full cost volume approach [8]. For a pair of full-size KITTI stereo images, PSM-Net [8] takes up to 4351 MB memory during inference. In contrast, our `DeepPruner-Best` and `DeepPruner-Fast` merely consume 1161 MB and 805 MB memory respectively. The storage requirements are less than one fourth of [8], which shows the potential to be integrated in mobile computing platform.

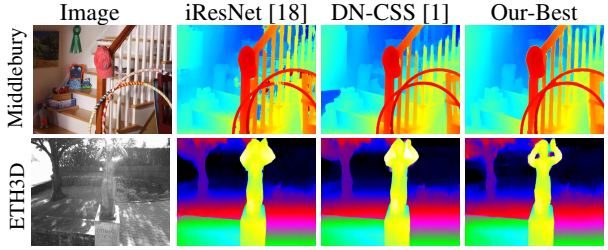


Figure 8: **Qualitative Results on ROB Challenge:** Our method captures the fine-grained geometry of the scenes. See the stair armrest and the arms of the sculpture.

## 5. Conclusion

In this paper we show how to exploit the fact that we can quickly prune parts of the cost volume for each pixel without requiring to fully evaluate its matching score. Towards this goal, we developed an end-to-end trainable network that exploits a novel differentiable PatchMatch as part of its internal structure. Our experiments show that our model achieves best performance among real-time methods and comparable results with the best performing methods while remaining several times faster. In the future we plan to apply our approach to both optical flow and scene flow tasks.

## References


[1] http://www.robustvision.net/.

[2] Stephen T Barnard and Martin A Fischler. Computational stereo. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1982.

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *TOG*. ACM, 2009.

[4] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, Sept. 2010.

[5] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. *IJCV*, 2014.

[6] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows.

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, 2010.

[8] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[9] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *arXiv*, 2018.

[10] David Ferstl, Christian Reinbacher, Rene Ranftl, Matthias Rüther, and Horst Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *ICCV*, pages 993–1000, 2013.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[12] Kaiming He and Jian Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[14] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *TPAMI*, 2008.

[15] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. 2017.

[16] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *ECCV*, 2018.

[17] Simon Korman and Shai Avidan. Coherency sensitive hashing. *TPAMI*, 2016.

[18] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. 2018.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[20] David G Lowe et al. Object recognition from local scale-invariant features. In *ICCV*, volume 99, pages 1150–1157, 1999.

[21] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *CVPR*, 2013.

[22] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016.

[23] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[24] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015.

[25] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV Workshop on Geometry Meets Deep Learning*, Oct 2017.

[26] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, 2014.

[27] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[28] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017.

[29] Xiao Song, Xu Zhao, Hanwen Hu, and Liangji Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. *arXiv*, 2018.

[30] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *TPAMI*, 2008.

[31] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE TPAMI*, 32(5):815–830, 2010.

[32] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *CVPR*, 2019.

[33] Stepan Tulyakov, Anton Ivanov, and François Fleuret. Practical deep stereo (pds): Toward applications-friendly deep stereo matching. In *NEURIPS*. 2018.

[34] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014.

[35] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. Segstereo: Exploiting semantic information for disparity estimation. In *ECCV*, 2018.

[36] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewénius, and David Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE TPAMI*, 2009.

[37] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. 2019.

[38] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *CSSP*, 2009.

[39] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, 1994.

[40] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015.

[41] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016.

[42] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.