SSGP: Sparse Spatial Guided Propagation for Robust and Generic Interpolation

René Schuster[1] Oliver Wasenmüller[1] Christian Unger[2] Didier Stricker[1]
[1]DFKI - German Research Center for Artificial Intelligence [2]BMW Group
firstname.lastname@{dfki,bmw}.de

Abstract

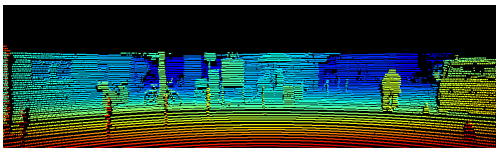
Interpolation of sparse pixel information towards a dense target resolution finds its application across multiple disciplines in computer vision. State-of-the-art interpolation of motion fields applies model-based interpolation that makes use of edge information extracted from the target image. For depth completion, data-driven learning approaches are widespread. Our work is inspired by latest trends in depth completion that tackle the problem of dense guidance for sparse information. We extend these ideas and create a generic cross-domain architecture that can be applied for a multitude of interpolation problems like optical flow, scene flow, or depth completion. In our experiments, we show that our proposed concept of Sparse Spatial Guided Propagation (SSGP) achieves improvements to robustness, accuracy, or speed compared to specialized algorithms.

1. Introduction

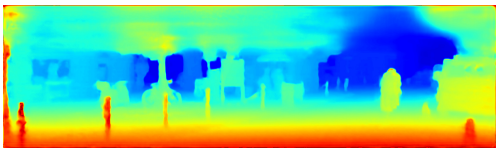
The problems of interpolation and extrapolation have a long history in mathematics and computer science. In high-level computer vision, interpolation finds its application in various problems like motion estimation in 2D (optical flow) [1, 2, 11, 14, 15, 20, 30, 31, 33, 43, 50], 3D (scene flow) [36, 37], or depth completion [6, 17, 26, 40, 41]. These methods in turn are applied in robot navigation, advanced driver assistance systems (ADAS), surveillance, and many others.

The strategies of previous work are quite distinct for motion field interpolation and depth completion. While the first focuses on hand-crafted models and piece-wise patches extracted from edge information, the latter fully relies on deep neural networks often considering image information insufficiently. With the learning capabilities and inherent parallelism of the data-driven approach, we want to further push the limits of motion field estimation towards higher accuracy and speed. At the same time, we extend and combine previous ideas from depth completion into a model that works equally well on different domains and applications. This exposes novel challenges like effective mechanisms for handling of sparse data with different patterns or densities, efficient strategies for guidance from dense image information, or suitable fusion of heterogeneous data (*e.g.* image and depth feature representations).

(a) Input image



(b) LiDAR measurements (visually enhanced)



(c) Densified depth with SSGP

Figure 1. We propose Sparse Spatial Guided Propagation (SSGP), a deep network for interpolation of sparse data. Here, an example of depth completion on KITTI [10] data is shown. Our full evaluation conducts experiments on more data sets and different types of input.

To solve the aforementioned challenges, we propose Sparse Spatial Guided Propagation (SSGP), which is the combination of spatially invariant, image dependent *convolutional propagation* and *sparsity-aware convolution*. This key concept is used in a *generic* sparse-to-dense encoder-decoder with *full image guidance* at every stage. Our overall contribution consists of the following:

- A unified architecture which performs sparse-to-dense interpolation in different domains, *e.g.* interpolation of opti-

cal flow, scene flow, or depth.

- A proper architectural design that leads to excellent robustness against noisy input or changes in the input density.
- Appropriate image guidance to resolve the dependency of previous flow interpolators on edge maps.
- A modification of existing spatial propagation that saves a vast amount of trainable parameters and improves generalization.
- Exhaustive experiments to validate all the above claims and to compare to state-of-the-art where in several cases SSGP produces top results.

2. Related Work

Sparse-to-Dense Motion Estimation. The interpolation of sparse points to a dense motion field dates back to at least [11, 30]. A practical approach for large displacement optical flow is introduced by EPICFlow [33]. The authors make use of image edges computed with SED [48] to find local edge-aware neighborhoods of previously computed, sparse flow values. Based on these neighborhoods, an affine 2D transformation is estimated to interpolate the gaps. Later, this concept is improved by RICFlow [14] to be more robust by using small superpixels and RANSAC in the estimation of the transformation. SFF [36] and SFF++ [37] take both interpolators for optical flow and transfer them to the scene flow setup. Throughout this work, we will refer to the interpolation modules of SFF and SFF++ as *EPIC3D* and *RIC3D* respectively. SemFlow [43] extends the above concepts for interpolation of optical flow by the use of deeply regressed semantic segmentation maps. These maps replace the edge information used in EPIC or RIC to improve the measure of similarity of connected neighborhoods of input matches. However, this approach is heavily dependent on semantic segmentation algorithms and thus not suitable for all domains and data sets. Lastly, InterpoNet [50] is another recent approach that considers deep neural networks for the actual interpolation task. Yet, InterpoNet still requires an explicit edge map as input.

In contrast to all interpolation modules mentioned, our network performs dense interpolation at full resolution for a multitude of problems (*i.e*. it is not restricted to optical flow or scene flow) and utilizes a trainable deep model (*i.e*. it is not subjected to hand-crafted rules or assumptions and provides significantly better run-times). Additionally, the existing approaches highly depend on an intermediate representation of the image (edges, semantics). SSGP operates on the input image directly and resolves this dependency.

Depth Completion. Most recent related work (especially in the area of deep learning) is concerned with depth completion. In this field, literature differentiates between unguided and guided depth completion. The latter utilizes the reference image for guidance. In the setup of guided depth completion, novel questions arise which are also highly relevant for this work, *e.g*. how to deal with sparse information in neural networks or how to combine heterogeneous feature domains. SparseConvNet [41] introduces sparsity invariant CNNs by normalizing regular convolutions according to a sparsity mask. This work has also introduced the Depth Completion Benchmark to the KITTI Vision Benchmark Suite [10]. Later, another strategy for the handling of sparsity was introduced by confidence convolution [9]. In this case, the authors replace the binary sparsity mask with a continuous confidence volume that is used to normalize features after convolution.

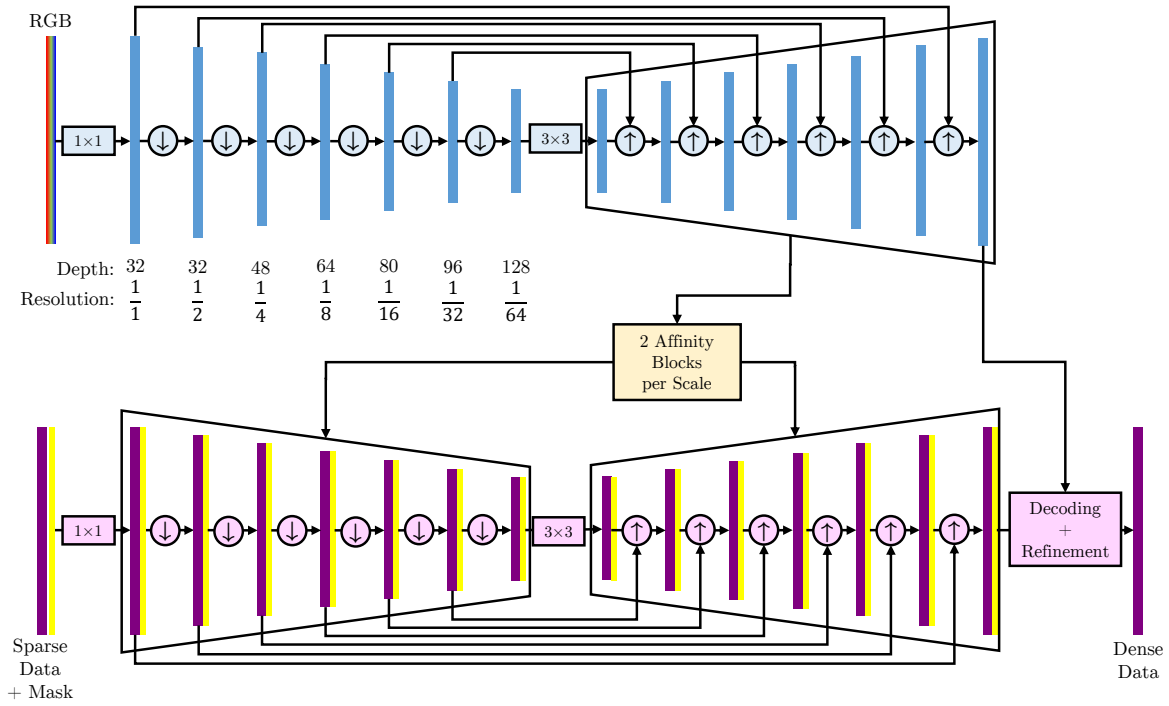
Another promising strategy is the use of spatially variant and content dependent kernels in convolutional networks [23, 45]. This idea is successfully used by [25] for semantic segmentation and later by CSPN [6] for the refinement of already densified depth maps. Most recently, GuideNet [40] has applied the same idea for the densification of sparse depth maps itself. In all cases, the idea is to predict per-pixel propagation kernels based on the image (or a feature map) directly instead of learning a spatially invariant set of kernels that is likewise applied to every pixel of the input.

We will make use of the two latterly presented concepts, namely awareness and explicit handling of sparsity as well as learning of spatially-variant and image-dependent convolutions. Both ideas will be combined in our novel, sparsity-aware, image-guided interpolation network that uses our new Sparse Spatial Guided Propagation (SSGP) module.

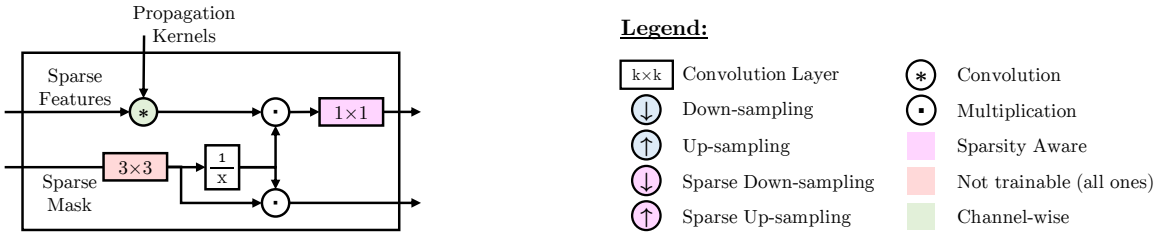
Other Interpolation Tasks. Lastly, there are more computer vision problems that are remotely related to our work, *e.g*. image inpainting which is also a problem of interpolation. However, for image inpainting the challenge usually lies within the reconstruction of the texture. For the interpolation of geometry or motion, the expected result is piece-wise smooth and thus the problem is rather to find semantically coherent regions. Still, related ideas can also be found in the field of image inpainting, where *e.g*. in [24] partial convolutions are used, which is the same idea for handling of sparsity as in [41]. Similarly, the task of super-resolution could also be posed as an interpolation problem with a regular pattern of sparse input. Though theoretically, our method is directly applicable to this family of problems, super-resolution goes beyond the scope of this paper and might be easier to be solved with other approaches.

3. Interpolation Network

As motivated earlier, we will use a deep neural network for the task of sparse-to-dense interpolation. The network has to be equipped with an appropriate mechanism for sparsity, otherwise the considerably large gaps in the used sparse-to-dense motion estimation pipelines can lead



(a) Overall network architecture showing the RGB and the sparse-to-dense codec.



(b) Our novel sparse spatial propagation module.

Figure 2. An overview of our network architecture (a) as well as a close-up view on our sparse spatial propagation module (b) which is used in the down- and up-sampling blocks of the sparse-to-dense codec.

to significantly deteriorated feature representation in these regions. For the same reason of large gaps in motion fields (contrary to *e.g.* depth completion where LiDAR measurements follow a predictable pattern of rotated scan lines), the network architecture has U-Net [34] structure. This way, even large gaps will be effectively closed after a few levels of the encoder, leading to a dense representation at the bottleneck. Additionally, to inject a maximal amount of guidance through the entire sparse-to-dense codec, the image information is used to compute spatially variant propagation kernels that are applied for densification by convolutional propagation in the sparse encoder, and for guided upsampling in the dense decoder. These guidance kernels are computed from the RGB image within a feature pyramid network with skip connections, for high expressiveness and accurate localization.

In summary, the interpolation network consists of four components. Firstly, the RGB codec for computation of image-dependent and spatially-variant propagation kernels (Section 3.1). Secondly, a sparse spatial propagation module that is likewise used within the encoder and decoder of the sparse-to-dense codec (Section 3.2). Thirdly, the u-shaped sparse-to-dense network that applies the propagation module for guidance and considers sparsity throughout (Section 3.3). Lastly, a dense refinement module to further improve the dense result. The combination of all elements – our sparse-to-dense interpolation network – is visualized in Figure 2.

3.1. RGB Codec

The purpose of the RGB codec is to provide a well-shaped feature representation of the image that fits the ac-

cording level of the sparse codec. Therefore it mimics the shape of the sparse codec and has the same number of levels l in the encoder and decoder as the interpolator. The image gets pre-processed by a regular 1×1 convolution and is then passed through l down-sampling blocks. Each consists of four 3×3 convolutions where the third convolution applies a stride of 2 to sub-sample the representation. After one additional convolution at the bottleneck, the representation of lowest resolution is passed through l up-sampling blocks. Again, each of these blocks consists of four 3×3 convolutions, but this time the second one is a transposed convolution with a stride of 2 for up-sampling. In addition after up-sampling, the intermediate feature representation gets concatenated with the next higher resolved level of the encoder, *i.e*. regular skip connections to re-introduce localization into the feature maps. In this architecture, the number of output channels is gradually increased as the spatial resolution is reduced which is a common practice for low resolution feature embeddings. In our setup, we use $l = 6$ pyramid levels with fully symmetric feature depth of 32, 32, 48, 64, 80, 96, and 128. An overview of the RGB codec is shown in Figure 2a.

Finally, we branch two affinity blocks from each level of the decoder to predict the spatially-variant, content-dependent kernels for each scale. One affinity block consists of two convolutional layers. One layer is used for pre-transformation, and one to predict a single $K \times K$ kernel per pixel for propagation in the sparse-to-dense codec. Please note, that different sets of propagation kernels are predicted for the encoder and the decoder of the sparse codec, *i.e*. weights are not shared for the two affinity blocks at each level of the RGB decoder. For reasons of memory consumption and computational efficiency, our propagation kernels have a size of $K = 3$. Contrary to existing work [6], our network uses a single, flat affinity map independent of the number of feature channels to propagate. This reduces the total number of parameters significantly and effectively diminishes over-fitting during fine-tuning on small data sets.

3.2. Sparse Spatial Propagation

The previously computed multi-scale feature maps, affinity maps, and propagation kernels are now used within our sparse spatial propagation module. Consider an arbitrarily shaped $H \times W \times C$ feature representation $\mathcal{S}$ of the sparse input along with a binary sparsity mask $\mathcal{M}$ of shape $H \times W \times 1$ and a feature representation $\mathcal{F}$ of the guidance image of the same spatial size (and potentially a different number of feature channels). The affinity block of the previous section will transform the image features $\mathcal{F}$ into a set of propagation kernels $\mathcal{K}$ of the shape $H \times W \times 1 \times K^2$. For the sake of affinity and propagation, the center pixel of the propagation kernels is fixed to 1, *i.e*. isolated sparse points will not be altered. These kernels are then applied in a channel-wise $K \times K$ convolution with the sparse representation $\mathcal{S}$ to spread the information into the neighborhood according to the image features. In GuideNet [40] one set of kernels is predicted for each feature channel of the sparse input, which leads to the necessity of depth-wise separable convolutions [7]. Other than that, we predict a single affinity map, which results in the natural use of depth-wise convolution for practicability and efficiency. After channel-wise spatial propagation, a 1×1 convolution is performed to mix the propagated input dimension and expand (or compress) the representation to a new feature depth. Further and in contrast to existing methods using convolutional spatial propagation, we explicitly model sparsity-awareness in our propagation module. Towards this end, we adopt the idea of sparse convolution from [41] and utilize the sparsity mask $\mathcal{M}$ to normalize the propagated features. By that, only valid information is spread according to the guidance image to fill in gaps. Formally, the output of the sparse spatial convolution of $\mathcal{S}$ with $\mathcal{K}$ for a single channel c and pixel is

$$\tilde{\mathcal{S}}_c = \frac{\sum_{i,j \in \mathcal{W}} \mathcal{S}_{c,i,j} \cdot \mathcal{K}_{i,j}}{\sum_{i,j \in \mathcal{W}} \mathcal{M}_{i,j}}, \tag{1}$$

where $\mathcal{W}$ is the $k \times k$ window around the pixel under consideration. The normalization and the propagation kernel are independent of the feature channel, *i.e*. there are only a single 1-channel mask $\mathcal{M}$ and single set of kernels $\mathcal{K}$ for the entire feature volume. This relationship is also visualized in Figure 2b. The entire concept expands directly to arbitrary batch sizes.

3.3. Image-guided Sparse-to-Dense Codec

The RGB codec and the sparse spatial propagation module enable an efficient way to introduce image guidance to our interpolation network. All convolutions of the sparse-to-dense codec make use of the sparse convolution as presented by [41]. Sparsity masks are used throughout the entire sparse codec which makes it easy to verify that full density is reached by the end of the decoder by the latest (usually already at the bottleneck), *i.e*. all pixels have been filled with information from the initially valid points. As with the RGB codec, we pre-process the sparse input with a sparse 1×1 convolution. Then, l sparse down-sampling blocks are applied. These blocks consist of our sparse spatial propagation module that applies the spatial guidance kernels from the RGB decoder, followed by a 1×1 convolution to complete the depth-wise separation of the spatially variant guidance. The last step within this block is a sparse average pooling layer with a kernel size of 3×3 and a stride of 2 to perform the sparse sub-sampling. Again, a single 3×3 convolution is applied at the bottleneck. Starting at lowest resolution from the bottleneck, l guided up-sampling blocks are passed through. As with the down-sampling, the first part of these blocks is the depth-wise separated sparse spatial

propagation. Then, the feature representation along with its validity mask are up-sampled using nearest-neighbor interpolation to avoid mixture with invalid pixels in case some are still remaining. Lastly, skip connections are established from the next higher resolution of the sparse encoder. The skipped encoder features are summed up with the decoder features to avoid re-introduction of sparsity into the feature representation and merged in another 3×3 convolution.

At full input resolution of the decoder pyramid, we perform one additional sparse spatial guided propagation, followed by three more convolutions for final decoding. The first two of these three are of size 3×3, the other is 1×1. The last two have linear activation to allow a final prediction of negative motions. We are aware that, theoretically, the two linear activated convolutions could be folded into a single one. However, we found that explicit separation leads to a faster convergence initially, probably due to better initialization by separation. Another advantage of using sparse convolution is that (especially during the decoding) no negative boundary effects are introduced, because the sparsity mechanism can treat padded areas as invalid.

3.4. Dense Refinement

At the end of the sparse-to-dense codec, a dense result in the respective target domain is already obtained. However, we follow the idea of CSPN [6] and further refine the result using spatial propagation for filtering. Since the RGB codec provides already a strong feature representation, we can transform these features into affinity maps for each output channel using a single 3×3 convolution. The kernels extracted from the affinity maps are further transformed to introduce stability as in CSPN [6]. The dense results are then refined during 10 iterations of spatial propagation.

3.5. Data, Training, and Implementation Details

Data Sets. For real applications, realistic data is required. However, labeling real world data with reference displacement fields is non-trivial and sometimes even impossible. Therefore, only a limited amount of suitable data sets is available. Additionally, these data sets are small in size, *i.e*. in the number of distinct images. This work will mainly use the KITTI 2015 data set [28] to cover realistic scenarios which only provides 200 annotated images for scene flow and optical flow. To overcome this issue, we will make use of synthetically generated data, namely the FlyingThings3D (FT3D) data set [27]. It provides approximately 2500 sequences, with 10 images each, of 3D objects flying in front of a random background image. This data set is large enough for deep training, but lacks variation in the scenes and realism. Still, it has been shown to be irreplaceable for pre-training [16, 27, 35, 39]. Next to KITTI and FT3D, Sintel [3] provides a trade-off between realism and size, though only for optical flow. Sintel comprises 23 sequences of 20 to 50 frames each. Additionally, we use HD1K [19] for extended experiments with interpolation of optical flow. For depth completion, the KITTI Benchmark Suite [10, 41] offers a larger and yet more realistic data set that provides labels for about 45000 stereo image pairs.

For all results in Section 4, we follow the common recommendation and perform our experiments on a randomly selected validation split which is not used for training. In particular these sets are the 20 sequences 4, 42, 46, 65, 92, 94, 98, 106, 115, 119, 121, 124, 146, 173, 174, 181, 184, 186, 190, 193 on KITTI, the original *val_selection_cropped* split from the KITTI depth completion data, the sequences *alley_2*, *ambush_4*, *bamboo_2*, *cave_4*, *market_5* for Sintel that sum up to 223 frames, and the sequences 0, 5, 15, 16, 18, 19, 27, 31 for HD1K.

Details. For large size data sets like FT3D, it is infeasible to compute the actual sparse input of existing sparse-to-dense pipelines, due to the high run-times of several seconds up to one minute per frame. Instead and because FT3D is only used for pre-training, a randomized sparsification process is introduced to simulate the sparse or non-dense input for interpolation. Additionally, random Gaussian noise ($\sigma = 2$ px) is added to all remaining valid pixels to simulate inaccuracies of a real matching process. For our experiments on optical flow and scene flow interpolation, we first train our network on FT3D [27]. The KITTI depth completion data set is sufficiently large to train on it directly. We pre-train for 1 million iterations which corresponds to approximately 64 epochs. Afterwards, we start training on the respective target domain and task with the pre-trained weights for initialization. For pre-training, photometric image augmentation is applied as in [8]. The objective for training depends on the specific interpolation problem at hand. For motion fields, the average Euclidean distance between predicted $\hat{\mathbf{p}}$ and ground truth $\mathbf{p}$ motion vectors is minimized. This loss function is equally used for optical flow and scene flow. For single valued depth, we optimize the mean squared error between ground truth d and prediction $\hat{d}$. Except for the two final linearly activated layers, we use ReLU activation [12] for all convolutional layers. ADAM [18] with an initial learning rate of 10^{-4} is used. The learning rate is continuously reduced with an exponential decay rate of 0.8 after every 10 % of the total number of steps. Due to hardware constraints, we are limited to a batch size of 1 for all our experiments. For training stability and improved generalization, we normalize all input of our network according to the respective image and sparse statistics to zero mean and unit variance.

4. Experiments and Results

Three sets of experiments are presented. The first one is an ablation study on the different components of the archi-

tecture to clarify our contributions and validate the impact. Then, we demonstrate the robustness of SSGP in terms of noisy input, wrong input, changes of density of the input, and padding artifacts. Lastly, SSGP is compared to state-of-the-art on various data sets and interpolation tasks.

For flow interpolation, the metrics under considerations are the end-point error (EPE) in image space, and the KITTI outlier error rate (KOE) giving the percentage of pixels that exceed an EPE of 3 px and deviate more than 5 % from the ground truth. Both metrics are likewise applied in our experiments on scene flow and optical flow. For depth completion, we use the default mean absolute error (MAE) and the root mean squared error (RMSE) as measure.

To obtain the sparse input for our experiments with optical flow, we use the prominent FlowFields (FF) [1] or its extension FlowFields+ (FF+) [2] along with their competitor CPM [15]. There has also been a longer history of sparse matching techniques in optical flow [13, 44]. However latest interpolation approaches [14, 33] have shown that these have been superseded by the FlowFields family or CPM. Their matching concept has been extended to a stereo camera setup to predict scene flow correspondences in SceneFlowFields (SFF) [36] and further to a multi-frame setup in SceneFlowFields++ (SFF++) [37]. To the best of our knowledge, these are the only approaches which have tested the sparse-to-dense approach for scene flow. For the problem of depth completion, sparse input is obtained directly from a LiDAR sensor.

4.1. Ablation Study

Part of our contributions is the combination of sparsity-awareness and spatial propagation for full guidance into an end-to-end interpolation network. Therefore, in this section our approach is compared to equivalent networks that differ only conceptually from our design. All the results of the ablation study are reported in Table 1. As a first step, we will validate that the fusion of image data into the sparse target domain (image guidance) is beneficial, especially when image data is available anyways. Towards that goal, we evaluate an *unguided* version of the sparse-to-dense codec, *i.e.* the input image is not used at all and the RGB branch is removed. Whenever the ablation removes our Sparse Spatial Guided Propagation, we replace it with a spatially invariant 3×3 convolution. We also test different variants of guidance. We remove guidance from either the encoder or decoder of the sparse-to-dense codec and compare to our fully guided approach. It is obvious that guidance improves the results significantly. Furthermore, guidance in the encoder alone (*enc*) performs not as good as in later stages of the network (*dec*), or during all stages (*full*). The latter two variants perform on a par, but we argue that full guidance improves results in difficult scenarios without much additional computational effort.

Table 1. Ablation study. We compare different concepts for sparse-to-dense interpolation of LiDAR measurements on the validation split of KITTI data. Mean absolute error (MAE) [mm], root mean squared error (RMSE) [mm], number of parameters ($\times 10^6$) and floating point operations ($\times 10^9$) are presented.

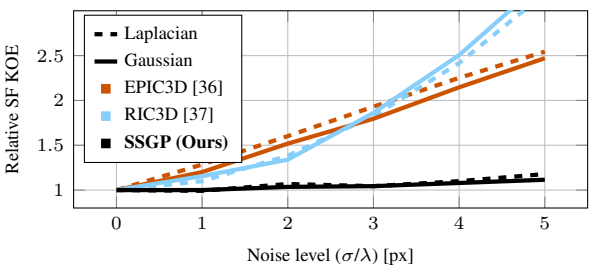
Guide	Sparse	Flat	Refine	MAE	RMSE	Params	FLOPs
none	yes	yes	no	356	1171	**0.93**	**41.2**
enc	yes	yes	no	312	1013	4.32	148.5
dec	yes	yes	no	289	**953**	4.47	149.5
full	yes	yes	no	**288**	957	4.61	156.9
enc	no	no	no	280	929	6.49	250.1
full	**yes**	**no**	no	276	915	10.14	382.4
full	**no**	**no**	no	270	910	10.14	381.3
full	**yes**	**yes**	no	288	957	**4.61**	156.9
full	**no**	**yes**	no	**267**	**908**	**4.61**	**155.8**
full	yes	no	**yes**	260	892	10.15	384.7
full	no	no	**yes**	251	881	10.15	383.6
full	yes	yes	**yes**	260	910	**4.61**	159.2
full	no	yes	**yes**	**248**	**877**	**4.61**	**158.1**

Next, we compare networks that use regular convolution wherever our design uses sparse convolution (*sparse*) and networks which compute either a full affinity volume for guidance or a single affinity map (*flat*). Because LiDAR measurements have a quite regular pattern across all samples, the network variants without sparse convolution perform in general slightly better than our versions with sparse convolution. Anyways, we will show in Section 4.2 that sparse convolution introduces higher robustness in case this property is not fulfilled. The flat versions reduce the network size and computational complexity by more than 50 % without much loss of accuracy. In fact, the version with flat guidance and regular convolutions performs the best. In later experiments with smaller data sets, we found the impact of flat guidance to be even more beneficial to reduce over-fitting. Lastly, we show that dense *refinement* improves the results for all variants with very little increase in number of parameters or FLOPs.

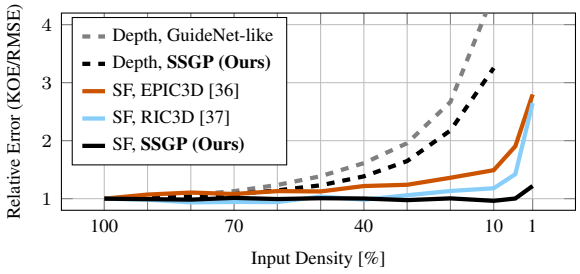
The fifth row in Table 1 represents a setup which is conceptually comparable to GuideNet [40], *i.e.* guidance is only used in the encoder, the network is not sparsity-aware, and guided propagation uses the full affinity volume. We call this setup *GuideNet-like*.

4.2. Robustness

In this section, the robustness of SSGP is demonstrated. We evaluate SSGP when the input is deteriorated with random noise, and when the density is reduced by random sampling. Both results are presented in Figure 3. For the experiment with noisy input, we add random Gaussian or Laplacian noise with zero mean and different values of standard deviation σ and exponential decay λ to all valid points of the sparse input. We then perform scene flow interpolation



(a) Results for scene flow interpolation when the input is superposed with different types and levels of random noise.



(b) Relative increase of errors when the input for different interpolation tasks is uniformly sparsified.

Figure 3. Experiments on the robustness of SSGP. We alter the input with additive Gaussian and Laplacian noise (a) or random sparsification for depth completion and interpolation of scene flow (b). Our novel architecture is most robust to any type or level of degradation.

and compare the relative increase of outliers for different levels of noise and different interpolation approaches with respect to the unaltered input. Figure 3a clearly shows, that our SSGP is extremely robust even to very noisy input. The outlier rate is maintained almost constant, while the competing methods perform considerably worse even for small amounts of additive noise.

In a second experiment, we also validate that the contribution of sparse convolution during guided propagation and the rest of the sparse-to-dense codec introduces higher invariance to the level of sparsity. Towards this end, we perform depth completion and scene flow interpolation with randomly sparsified input. Results are presented in Figure 3b. The increase of errors for the sparsity-aware model is about 50 % less when considering very sparse depth measurements. For SSGP on scene flow (*SF*), the impact of sparisfication is neglectable until 1 % of the original density. Note that all models are trained on the full input density. This improved robustness applies also to changes in the pattern of the input, *e.g.* when the LiDAR measurements are sparsified non-uniformly.

As additional indicator for the robustness of SSGP, we measure the *outlier rejection rate* (ORR), *i.e.* the percentage of input that is classified as scene flow outlier before interpolation, but is corrected during interpolation. For input from SFF and SFF++, EPIC3D achieves ORRs of 51.2 % and 40.3 %, RIC3D achieves 64.2 % and 55.7 %, and our SSGP yields ORRs of **67.6 %** and **56.7 %**.

We also compare the errors at boundary regions of the image to show the robustness of sparse convolution to padding. While the *GuideNet-like* variant obtains an MAE and RMSE of 186 and 505 mm in regions which are less than 10 px away from the image boundary, our full setup of SSGP achieves **140** and **448 mm**.

4.3. Interpolation

Scene Flow. As first application to our interpolation network, we use matches from SFF [36] and SFF++ [37] (using the SDC feature descriptor [38]) for interpolation of dense scene flow. The results are computed on the KITTI data set [28] and are compared to EPIC3D [36] and RIC3D [37] which are the heuristic two-stage interpolators of SFF and SFF++ respectively. Both use additional edge information of the scene. Results are given in Table 2.

Table 2. Evaluation of *scene flow* interpolation on our validation split of the KITTI scene flow data set. KITTI outliers (KOE) [%], end-point error (EPE) [px], and run time [s] are reported.

Input	Method	D0 KOE	D0 EPE	D1 KOE	D1 EPE	OF KOE	OF EPE	SF KOE	SF ΣEPE	Run time
SFF	EPIC3D [36]	12.83	1.88	17.80	11.49	29.62	112.1	31.72	125.4	1.0
	RIC3D [37]	9.88	1.92	13.94	2.79	**15.44**	8.42	**17.45**	13.10	3.8
	SSGP (Ours)	**9.06**	**1.33**	**13.93**	**1.83**	20.67	**5.04**	25.19	**8.20**	**0.19**
SFF++ + SDC	EPIC3D [36]	6.74	1.30	10.83	1.96	15.65	6.23	17.91	9.49	1.0
	RIC3D [37]	5.91	1.29	**7.24**	1.53	**9.80**	3.33	**11.50**	6.15	3.8
	SSGP (Ours)	**5.71**	**1.04**	9.89	**1.45**	12.39	**3.00**	16.61	**5.50**	**0.19**

Our approach achieves competitive performance to previous methods, though being significantly faster. Especially for interpolation of initial disparity (*D0*), SSGP outperforms the baselines. Further, SSGP performs comparatively well in the EPE metric, which was also the objective function during training.

Optical Flow. For the experiments related to optical flow, we have multiple data sets to evaluate on, namely KITTI [28], HD1K [19], and Sintel [3]. We evaluate our method and state-of-the-art for two kinds of input matches generated from FF+ [2] and CPM [15]. Our approach will be compared to EPICFlow [33], RICFlow [14], and InterpoNet [50]. Note, that all three methods use additional edge information, while we feed the raw image to our network. A visual comparison for a cropped frame of KITTI is presented in Figure 4. In this example, SSGP presents a globally consistent result, even in the static part of the scene, where small deviations have most impact in the visualization. Our approach shows the most accurate and sharp object contours, even though it is not provided with precomputed edge information. This highlights the capabilities of the full guidance strategy. In fact, our approach is able to reject wrong matches in shadows of the vehicles during

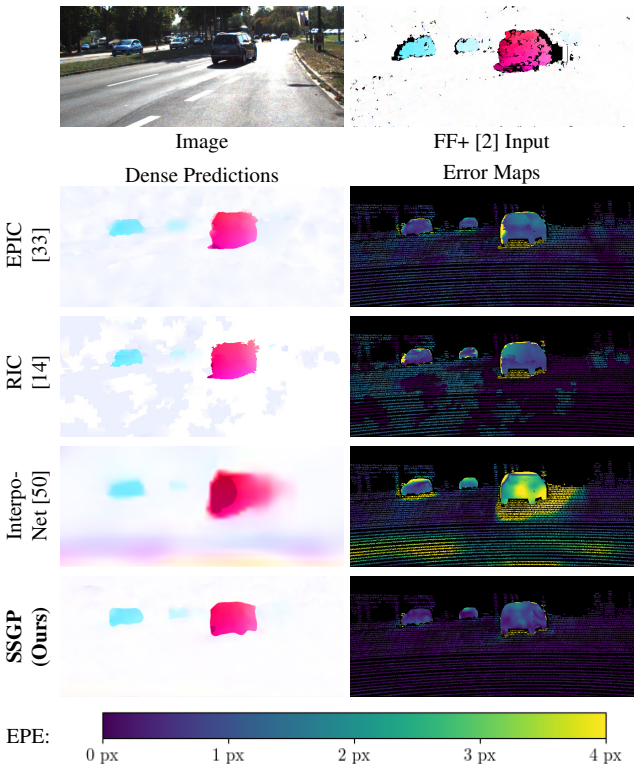


Figure 4. Visual comparison of optical flow interpolation on the KITTI data set.

interpolation.

Table 4 compares quantitative results over our entire validation sets. It is to highlight that SSGP cuts the end-point error on KITTI by about half in our comparison. On KITTI also, the outlier rates of SSGP beat all previous work. For completeness and fairness, we have to mention that we are using the publicly available pre-trained weights of InterpoNet [50] that have been fine-tuned on Sintel with input from DF [29] and on KITTI with matches from FlowFields [1]. However, this indicates that InterpoNet is not very robust to changes of the input. On Sintel, our approach is on par with InterpoNet, but lacks behind the other methods. This is due to the limited variance between scenes which makes it hard to train a deep model on Sintel. Yet on HD1K, our SSGP outperforms state-of-the-art in all metrics while also being faster.

Table 4. Evaluation of interpolation of *optical flow*. We test on our validation splits of the KITTI, HD1K, and Sintel data sets. Outlier rates (KOE) [%], end-point error (EPE) [px], and run time [s] are reported.

		KITTI		HD1K		Sintel clean		Sintel final		
Input	Method	KOE	EPE	KOE	EPE	KOE	EPE	KOE	EPE	Run time
CPM [15]	EPICFlow [33]	24.39	10.04	5.43	1.11	9.98	**3.84**	13.94	**5.76**	0.4
	RICFlow [14]	21.98	9.91	5.02	1.09	**9.17**	4.05	**13.60**	5.88	2.8
	InterpoNet [50]	40.38	12.81	12.3	2.36	14.94	4.75	18.09	6.24	0.3
	SSGP (Ours)	**20.26**	**5.02**	**4.32**	**0.83**	14.97	5.63	20.33	7.27	**0.16**
FF+ [2]	EPICFlow [33]	23.97	11.34	5.55	1.21	11.25	**5.05**	15.99	**7.26**	0.4
	RICFlow [14]	20.46	10.17	4.88	1.07	**10.59**	5.59	**15.82**	8.19	2.8
	InterpoNet [50]	37.08	11.34	13.1	2.35	16.49	5.7	20.51	7.64	0.3
	SSGP (Ours)	**20.34**	**5.21**	**4.54**	**0.85**	16.53	6.55	22.20	8.43	**0.16**

Depth Completion. SSGP can also be used for the completion of sparse LiDAR measurements. We train the entire architecture from scratch on the KITTI depth completion data set [41] and compare our results to state-of-the-art in Table 3. Our network again achieves a competitive result on yet another challenge, indicating its broad applicability. A visual example of an interpolated depth map is given in Figure 1. We further notice that RIC3D [37], a top-performing method for interpolation of scene flow, performs considerably worse than any other approach. This shows, that even though RIC3D is not a learning-based method, it has a strong dependency on properly selected hyper-parameters.

5. Conclusion

SSGP successfully combines sparsity-aware convolution and spatially variant propagation for fully image guided interpolation. The network design is applicable to diverse sparse-to-dense problems and achieves competitive performance throughout all experiments, beating state-of-the-art in interpolation of optical flow and in terms of EPE. A flat affinity map can be used for spatial guidance equally well as a full affinity volume, drastically reducing the overall network size. This strategy for guidance resolves the dependency on explicitly pre-computed edge information resulting in even more accurate interpolation boundaries with a globally consistent output that preserves fine details. SSGP is especially robust to variations of the sparsity pattern and to noise in the input.

Acknowledgement: This work was partially funded by the BMW Group and partially by the Federal Ministry of Education and Research Germany under the project VIDETE (01IW18002).

Table 3. Comparison of methods for depth completion on the KITTI benchmark [41]. We report mean average error (MAE [mm]), root mean squared error (RMSE [mm]), and run time [ms] for the best performing, published methods using image guidance out of more than 90 total submissions. Values in gray are computed on the validation split.

	GuideNet [40]	CSPN++ [5]	FuseNet [4]	DeepLiDAR [32]	MSG-CHN [22]	Guide&Certainty [42]	PwP [46]	CrossGuidance [21]	Sparse-to-Dense [26]	NConv-CNN [9]	DDP [47]	**SSGP (Ours)**	Spade [17]	DFineNet [49]	CSPN [6]	RIC3D [37]
MAE	219	209	221	227	220	215	235	254	250	233	**204**	245	235	304	279	588
RMSE	**736**	744	753	758	762	773	777	807	815	830	833	838	918	945	1020	2477
Run time	140	200	90	70	**10**	20	100	200	80	20	80	140	70	20	1000	1400

References

[1] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *International Conference on Computer Vision (ICCV)*, 2015.

[2] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012.

[4] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *International Conference on Computer Vision (ICCV)*, 2019.

[5] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. CSPN++: Learning context and resource aware convolutional spatial propagation networks for depth completion. *Conference on Artificial Intelligence (AAAI)*, 2020.

[6] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *European Conference on Computer Vision (ECCV)*, 2018.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[9] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Confidence propagation through cnns for guided sparse depth regression. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

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

[11] David Gibson and Michael Spann. Robust optical flow estimation based on a sparse motion trajectory set. *Transactions on Image Processing (TIP)*, 2003.

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AIStats)*, 2011.

[13] Kaiming He and Jian Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] Yinlin Hu, Yunsong Li, and Rui Song. Robust interpolation of correspondences for large displacement optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with CNNs: Depth completion and semantic segmentation. In *International Conference on 3D Vision (3DV)*, 2018.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*, 2015.

[19] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016.

[20] Manuel Lang, Oliver Wang, Tunc Aydin, Aljoscha Smolic, and Markus Gross. Practical temporal consistency for image-based graphics applications. *Transactions on Graphics (ToG)*, 2012.

[21] Sihaeng Lee, Janghyeon Lee, Doyeon Kim, and Junmo Kim. Deep architecture with cross guidance between single image and sparse lidar data for depth completion. *IEEE Access*, 2020.

[22] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[23] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep joint image filtering. In *European Conference on Computer Vision (ECCV)*, 2016.

[24] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision (ECCV)*, 2018.

[25] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[26] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[27] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[29] Moritz Menze, Christian Heipke, and Andreas Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition (GCPR)*, 2015.

[30] Mircea Nicolescu and Gérard Medioni. Layered 4d representation and voting for grouping from motion. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2003.

[31] Maria Oliver, Lara Raad, Coloma Ballester, and Gloria Haro. Motion inpainting by an image-based geodesic amle method. In *International Conference on Image Processing (ICIP)*, 2018.

[32] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI)*, 2015.

[35] Rohan Saxena, René Schuster, Oliver Wasenmüller, and Didier Stricker. PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation. In *Intelligent Vehicles Symposium (IV)*, 2019.

[36] René Schuster, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. SceneFlowFields: Dense interpolation of sparse scene flow correspondences. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[37] René Schuster, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. SceneFlowFields++: Multiframe matching, visibility prediction, and robust interpolation for scene flow estimation. *International Journal on Computer Vision (IJCV)*, 2020.

[38] René Schuster, Oliver Wasenmüller, Christian Unger, and Didier Stricker. SDC - Stacked dilated convolution: A unified descriptor network for dense matching tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[40] Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. Learning guided convolutional network for depth completion. *arXiv preprint arXiv:1908.01238*, 2019.

[41] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *International Conference on 3D Vision (3DV)*, 2017.

[42] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy LiDAR completion with RGB guidance and uncertainty. In *International Conference on Machine Vision Applications (MVA)*, 2019.

[43] Xianshun Wang, Dongchen Zhu, Yanqing Liu, Xiaoqing Ye, Jiamao Li, and Xiaolin Zhang. SemFlow: Semantic-driven interpolation for large displacement optical flow. *IEEE Access*, 2019.

[44] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *International Conference on Computer Vision (ICCV)*, 2013.

[45] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[46] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *International Conference on Computer Vision (ICCV)*, 2019.

[47] Yanchao Yang, Alex Wong, and Stefano Soatto. Dense depth posterior (DDP) from single image and sparse range. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] Yilun Zhang, Ty Nguyen, Ian D Miller, Steven Chen, Camillo J Taylor, Vijay Kumar, et al. DFineNet: Ego-motion estimation and depth refinement from sparse, noisy depth input with rgb guidance. *arXiv preprint arXiv:1903.06397*, 2019.

[50] Shay Zweig and Lior Wolf. InterpoNet, a brain inspired neural network for optical flow dense interpolation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.