

CROSS SPECTRAL IMAGE RECONSTRUCTION USING A DEEP GUIDED NEURAL NETWORK

Frank Sippel, Jürgen Seiler, and André Kaup

Friedrich-Alexander-Universität Erlangen-Nürnberg
Multimedia Communications and Signal Processing
Cauerstraße 7, 91058 Erlangen, Germany

ABSTRACT

Cross spectral camera arrays, where each camera records different spectral content, are becoming increasingly popular for RGB, multispectral and hyperspectral imaging, since they are capable of a high resolution in every dimension using off-the-shelf hardware. For these, it is necessary to build an image processing pipeline to calculate a consistent image data cube, i.e., it should look like as if every camera records the scene from the center camera. Since the cameras record the scene from a different angle, this pipeline needs a reconstruction component for pixels that are not visible to peripheral cameras. For that, a novel deep guided neural network (DGNet) is presented. Since only little cross spectral data is available for training, this neural network is highly regularized. Furthermore, a new data augmentation process is introduced to generate the cross spectral content. On synthetic and real multispectral camera array data, the proposed network outperforms the state of the art by up to 2 dB in terms of PSNR on average. Besides, DGNet also tops its best competitor in terms of SSIM as well as in runtime by a factor of nearly 12. Moreover, a qualitative evaluation reveals visually more appealing results for real camera array data.

Index Terms— Multispectral Imaging, Image Reconstruction, Deep Learning

1. INTRODUCTION

Multispectral camera arrays[1, 2] and hyperspectral camera arrays[3] are capable of recording different spectral areas by employing multiple cameras. In contrast to other multispectral and hyperspectral imaging devices, this approach has the advantage of yielding a high resolution in spatial and temporal dimension as well as being cost-efficient and flexible regarding the used filters. The recorded channels do not necessarily have to lie in the visible wavelength area, but can also

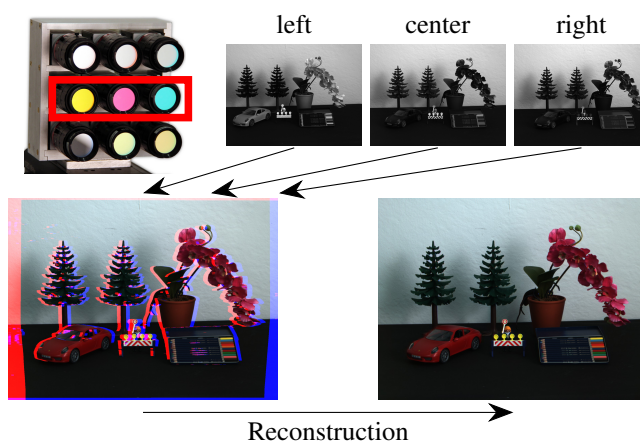


Fig. 1. A multispectral camera array (CAMSI) with three corresponding recorded images of the center row (top). Additionally, the registered multispectral image with missing pixels (bottom left) and the reconstructed image (bottom right) are depicted.

contain information about areas in the infrared or ultraviolet area of the spectrum. These areas of the spectrum are particularly interesting for classification problems. Multispectral and hyperspectral cameras can be used in medicine to classify the degree of burn [4], in recycling to sort materials [4], in forensics by determining the age of blood [5], or in agriculture by discriminating between parts of fields that need water and fertilizer and those that are healthy [6].

Genser et al. [7] introduced a multispectral camera array consisting of nine cameras. However, a reconstruction pipeline to reconstruct a consistent multispectral datacube is necessary. The goal of this processing pipeline is to reconstruct a consistent multispectral center view by mapping all peripheral cameras to the center perspective, see Fig. 1. Then, each pixel of the multispectral datacube shows exactly the same object in different spectral bands.

This reconstruction pipeline consists of multiple separate

The authors gratefully acknowledge that this work has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 491814627.

Source code: <https://github.com/FAU-LMS/dgnet>.

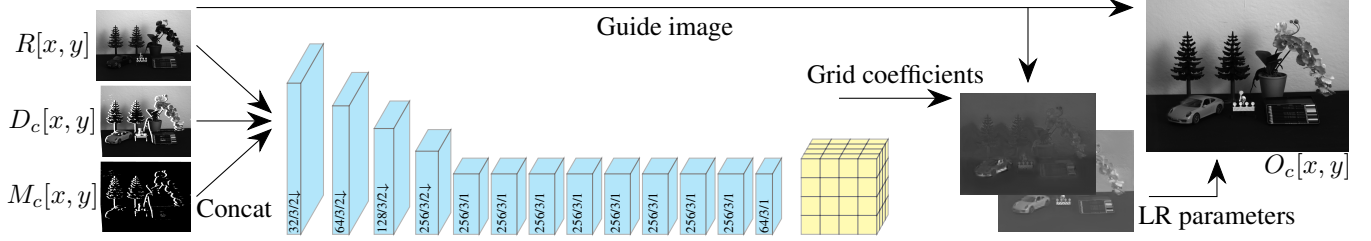


Fig. 2. The architecture of the proposed network. Blue blocks are convolutional layers with parameters channels/kernel size/stride.

steps. First, a calibration procedure is necessary, since the cameras within the camera array as well as the sensors within the camera housing are not perfectly aligned. For that, a checkerboard pattern is placed in front of the camera, and the images are calibrated to this depth. Afterwards, since the displacement of the pixels of the peripheral cameras is depth-dependent, a cross spectral depth estimation is performed, e.g., by [8]. Subsequently, the pixels of the peripheral cameras can be warped to the center view based on this depth map. However, due to occlusions the center camera sees pixels that the individual peripheral cameras do not see. Hence, a cross spectral reconstruction process is necessary that fills in these missing pixels.

Fortunately, this is not a pure inpainting problem as tackled by classical algorithms [9, 10] or neural networks [11, 12, 13], since the complete center view is available as reference image. However, this reference image lies in a different spectral band, thus only the structure can be used as guiding information for the reconstruction process. Algorithms from literature that also use this guiding information are [7] and recently also [14]. These algorithms are all based on classical image processing techniques like guided filtering and non-local filtering. However, these methods are not able to establish global relationships between guide pixels and valid pixels in the distorted view. Furthermore, these methods typically fail in reconstructing high frequency areas of the image, since they are relying on enough similar reference pixel being locally available. Moreover, until now there are no guided reconstruction methods based on neural networks. A big challenge of training a neural network for this task is that there is too little multispectral and hyperspectral data available to train such a network properly.

2. PROPOSED NETWORK

The basic idea is to estimate a low dimensional cube of linear regression coefficients through a convolutional neural network. Afterwards, the cube is sliced into linear regression parameters for each pixel using a guide image. Finally, these linear regression coefficients are used on the guide image to obtain the final result.

2.1. Architecture

In Fig. 2, the network architecture of the proposed Deep Guided Neural Network (DGNet) is depicted. The input images with pixel coordinates (x, y) are the reference image $R[x, y]$, i.e., the center view, the distorted image $D_c[x, y]$, i.e., channel c of a warped peripheral view, and the mask $M_c[x, y]$, which indicates pixels that need to be reconstructed. The missing pixels in $D_c[x, y]$ are set to white as well. These images are concatenated and put into downscaling convolutional layers with stride 2. The number of of downscaling layers depends on how many pixels should be within one spatial bin. For example, for a bin size of 16×16 , 4 downscaling layers are needed. For every downscaling layer, the number of channels is doubled. Afterwards, nine convolutional layers without any scaling are applied, which yield cubes of size $G_w \times G_h \times (8 \cdot G_l)$, where G_l is the desired amount of luma bins in the resulting linear regression cube [15, 16]. This also increases the perceptual field, which is necessary for large areas of missing pixels. Then, a final convolutional layer is applied, which transforms the previous number of channels into the proper number of channels for the linear regression bilateral grid with dimensions $G_w \times G_h \times (2 \cdot G_l)$. For this layer, only weights are applied and no activation is used. Therefore, this layer can be interpreted as a linear combination of features for determining the linear regression coefficients of the grid. Finally, this tensor is split into two to yield one low-resolution cube for the linear parameter $A_c^l[x, y, z]$ of size $G_w \times G_h \times G_l$ and one low-resolution cube for the linear regression bias $B_c^l[x, y, z]$ of the same size.

The next step is to use the reference image $R[x, y]$ of size $W \times H$ to slice the coefficients to two high resolution images of linear regression coefficients A_c^h and B_c^h . For that, trilinear interpolation is used

$$A_c^h[x, y] = \sum_{i,j,k} \theta(s_x x - i) \theta(s_y y - j) \theta(s_z R[x, y] - k) \cdot A_c^l[i, j, k], \quad (1)$$

where $\theta(\cdot) = \max(1 - |\cdot|, 0)$ is the linear interpolation kernel, s_x and s_y are the ratios of the spatial dimensions of the grid with respect to the high-resolution image, i.e., $s_x = G_w/W$. Similarly, s_z is the ratio of luma bins to the maxi-

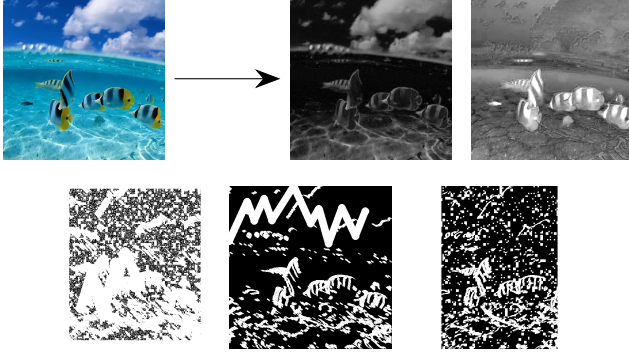


Fig. 3. In the first row, two generated spectral grayscale images from an RGB image are shown. In the bottom row, three masks are depicted. Note that the edge mask is generated using the RGB image of the first row.

imum possible intensity value of the grayscale reference image. Of course, the same interpolation is also performed to yield the high-resolution bias B_c^h . The output image of the network is calculated by applying these coefficients to the reference image

$$O_c[x, y] = A_c^h[x, y] \cdot R[x, y] + B_c^h[x, y]. \quad (2)$$

The resulting multispectral channel $I_c[x, y]$ is calculated by only using the reconstructed image for masked pixels

$$I_c[x, y] = (1 - M_c[x, y]) \cdot D_c[x, y] + M_c[x, y] \cdot O_c[x, y]. \quad (3)$$

2.2. Data Augmentation

A big challenge to train this network is that there is not enough multispectral data available to yield diverse scenes. Thus, we suggest a smart data augmentation using RGB data, since there are many large size RGB databases with thousands of different scenes. Furthermore, realistic masks needs to be generated.

2.2.1. Spectral Image Generation

The main goal here is not to perfectly generate spectral data out of these images, but rather imitate that the objects have different textures for different spectral bands. To be consistent across the whole object, the idea of this data augmentation is to map random grayscale values to a certain number of colors in the image. This is done by transforming the RGB image to an HSV image and use the hue value to assign colors. Between these colors, a linear interpolation is performed for consistency reasons. Moreover, random intensities for white and black are generated to also use the saturation and value of the HSV image. Again, linear interpolation is used. Subsequently, the grayscale image is normalized to the full intensity range and a random exposure between 0.2 and 1.5 is applied to also cover over- and underexposed cases. Finally, synthetic

white Gaussian noise is added and the image is clipped to the intensity range.

2.2.2. Mask Generation

Realistic masks also need to be generated for training. To that end, five different patterns are implemented. First, the classic stroke mask known from inpainting papers such as [11] is used. Moreover, a random pixel loss mask and a random block loss mask are used to simulate cases, where the depth map is noisy. Furthermore, a border mask is provided, which just cuts off random portions of random borders of the image. This also happens in reality, since the cameras are spatially distributed and thus cannot see all borders of the center camera. Finally, an edge mask can be applied. Typically, missing pixels occur on edges of objects, because often there is a bigger disparity difference between foreground and background. Therefore, an edge mask is calculated by extracting edges from the RGB image using a Canny edge detector. These thin edges are extended to a random width in a random direction by calculating starting and ending points of lines using the gradient map and setting all pixels on these lines to be reconstructed. In general, more diverse masks will make the network generalize better as long as enough image content is shown to estimate the coefficients. Examples for generated images and masks are shown in Fig. 3.

2.3. Configuration

Since only convolutional layers are used, the network can handle any image size. The input images need to be padded such that all cube bins cover full image areas. Otherwise, the border linear regression coefficients will not be properly estimated. The desired spatial bin size is set to 16×16 , while there are $G_l = 32$ luma bins. A weighted l_1 -loss is used as loss function

$$L(\hat{I}_c[x, y], O_c[x, y], M_c[x, y]) = \frac{1}{WH} \sum_{x,y} \begin{cases} \alpha \cdot |O_c[x, y] - \hat{I}_c[x, y]|, & \text{if } M_c[x, y] = 1 \\ |O_c[x, y] - \hat{I}_c[x, y]|, & \text{else,} \end{cases} \quad (4)$$

where α is the weight factor for missing pixel positions. This weight factor is set to 10 to put more importance to missing regions. Moreover, the cells in the grid should be steered towards reconstructing missing parts of the image rather than already existing parts. However, it is still important that the whole image is reconstructed such that the network learns how images look like. As optimizer, Adam with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$ is picked. The network is trained for 64 epochs using the Places2 database [17], which contains approximately 1.8 million RGB images. The learning rate starts at 0.0001 and is halved at epochs 20, 32, 40, 48, and 56.

Table I. Average PSNR in dB and SSIM of different reconstruction methods on different databases [18, 19, 7].

Database		FF[12]	EC[11]	FSR[10]	NOCS[14]	CSR[7]	DGNet
PSNR	HyViD[18]	33.72	29.40	34.32	35.29	38.98	40.95
	MiBu[19]	28.85	27.38	30.12	30.94	33.64	35.43
	CAMSI[7]	28.57	28.40	28.21	29.78	29.26	30.57
SSIM	HyViD[18]	0.942	0.869	0.951	0.962	0.974	0.989
	MiBu[19]	0.923	0.895	0.931	0.934	0.949	0.952
	CAMSI[7]	0.896	0.896	0.901	0.906	0.907	0.913

Table II. Single-thread CPU runtime of all methods in seconds.

FF[12]	EC[11]	FSR[10]	NOCS[14]	CSR[7]	DGNet	DGNet on GPU
2960	391	855	97.8	162	8.27	0.34

3. EVALUATION

The evaluation is done quantitatively as well as qualitatively. Our novel DGNet is compared against the inpainting methods Free Form Inpainting (FF) [12], EdgeConnect (EC) [11], Frequency Selective Reconstruction (FSR) [10], and the guided reconstruction methods Cross-Spectral Reconstruction (CSR) from [7] and Non-local Cross-Spectral Reconstruction (NOCS) [14].

3.1. Synthetic Data

To evaluate DGNet, the HyViD database [18] is used, which contains 7 scenes, each with 30 frames. The images have 31 hyperspectral channels and are rendered from a camera array very similar to the one in Fig. 1. Moreover, ground-truth depth data is available. Hence, the database can be used to evaluate realistic cross spectral reconstruction problems. For that, the 31 hyperspectral channels are synthetically filtered to nine different bandpasses, including a red, green and blue component.

The ground-truth depth is used for the reconstruction pipeline to warp the images accordingly. In Table I, the results are summarized in terms of average PSNR and SSIM, which indicates that our novel neural network-based method outperforms the reference algorithms. Moreover, the best of three runtimes using a single thread CPU implementation is shown in Table II, which are evaluated on the first frame of the scene *family house*. DGNet outperforms the reference algorithms by a factor of at least 11.8. When executed on an RTX 3090 GPU, the runtime is decreased to 340 ms, corresponding to a factor of 287 in comparison to the second fastest method NOCS.

3.2. Real Data

To evaluate real-world performance, an evaluation based on the Middlebury 2006 database [19], which contains 21 scenes, as well as on the database of [7], which are both much smaller



Fig. 4. Reconstruction results of spectrally guided methods as (false) color images combining three spectral bands. Left column: center row of camera array (red, green, blue). Right column: top row of camera array (infrared filters at 750 nm, 850 nm and 950 nm). Zoomed areas are depicted using red rectangles.

than the synthetic database, is shown in Table I. Note that the Middlebury database only contains RGB images. Hence, the red channel of the left camera, the green channel of the center camera and the blue channel of the right camera are chosen to simulate a linear camera array. Again, the proposed DGNet outperforms all reference methods. Additionally, another image was recorded using CAMSI [7] for qualitative evaluation. In Fig. 4, false color images of the cross spectral methods and our novel network are depicted. As indicated, the novel DGNet produces fewer artefacts than its competitors.

4. CONCLUSION

This paper introduced a neural network-based method for reconstructing occluded pixels using a guide image, e.g., when employing a multispectral camera array. For that, a cube of grid coefficients for linear regression is estimated in spatial as well as intensity direction. Afterwards, this cube is sliced into two high-resolution linear regression parameter images using the guide image. Finally, the linear regression coefficients are applied to this guide image to yield a final estimate of the peripheral view. It was shown that this method outperforms the state of the art by up to 2 dB in terms of PSNR on multispectral camera array data as well as in terms of SSIM. The novel neural network also provides a significant speedup and yields visually more accurate images.

5. REFERENCES

- [1] S. Dandriofosse, A. Carlier, B. Dumont, and B. Mercatoris, “Registration and fusion of close-range multi-modal wheat images in field conditions,” *Remote Sensing*, vol. 13, no. 7, 2021.
- [2] F. Huang, P. Lin, R. Cao, B. Zhou, and X. Wu, “Dictionary learning- and total variation-based high-light-efficiency snapshot multi-aperture spectral imaging,” *Remote Sensing*, vol. 14, no. 16, 2022.
- [3] Á. Gómez Manzanares, D. Vázquez Moliní, A. Alvarez Fernandez-Balbuena, S. Mayorga Pinilla, and J. C. Martínez Antón, “Measuring high dynamic range spectral reflectance of artworks through an image capture matrix hyperspectral camera,” *Sensors*, vol. 22, no. 13, 2022.
- [4] M. Moroni, A. Mei, A. Leonardi, E. Lupo, and F. Marca, “PET and PVC Separation with Hyperspectral Imagery,” *Sensors*, vol. 15, no. 1, pp. 2205–2227, Jan. 2015.
- [5] G. J. Edelman, T. G. van Leeuwen, and M. C. G. Aalders, “Hyperspectral imaging of the crime scene for detection and identification of blood stains,” in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XIX*, Sylvia S. Shen and Paul E. Lewis, Eds. International Society for Optics and Photonics, 2013, vol. 8743, p. 87430A, SPIE.
- [6] M. Cardim Ferreira Lima, A. Krus, C. Valero, A. Barrientos, J. del Cerro, and J. J. Roldán-Gómez, “Monitoring plant status and fertilization strategy through multi-spectral images,” *Sensors*, vol. 20, no. 2, 2020.
- [7] N. Genser, J. Seiler, and A. Kaup, “Camera Array for Multi-Spectral Imaging,” *IEEE Transactions on Image Processing*, vol. 29, pp. 9234–9249, 2020.
- [8] N. Genser, A. Spruck, J. Seiler, and A. Kaup, “Deep learning based cross-spectral disparity estimation for stereo imaging,” in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2536–2540.
- [9] P. Getreuer, “Total Variation Inpainting using Split Bregman,” *Image Processing On Line*, vol. 2, pp. 147–157, July 2012.
- [10] N. Genser, J. Seiler, and A. Kaup, “Spectral Constrained Frequency Selective Extrapolation for Rapid Image Error Concealment,” in *Proc. 25th International Conference on Systems, Signals and Image Processing (IWS-SIP)*. June 2018, pp. 1–5, IEEE.
- [11] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, “Edgeconnect: Structure guided image inpainting using edge prediction,” in *Proc. IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 3265–3274.
- [12] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, “Free-form image inpainting with gated convolution,” in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4470–4479.
- [13] G. Liu, F. A. Reda, K. J. Shih, T. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *Proc. European Conference on Computer Vision (ECCV)*, Sept. 2018, pp. 85–100.
- [14] F. Sippel, J. Seiler, and A. Kaup, “Spatio-spectral image reconstruction using non-local filtering,” in *Proc. International Conference on Visual Communications and Image Processing (VCIP)*, 2021, pp. 1–5.
- [15] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff, “Bilateral guided upsampling,” *ACM Trans. Graph.*, vol. 35, no. 6, Dec. 2016.
- [16] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, “Deep bilateral learning for real-time image enhancement,” *ACM Trans. Graph.*, vol. 36, no. 4, July 2017.
- [17] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.
- [18] F. Sippel, J. Seiler, and A. Kaup, “Synthetic hyperspectral array video database with applications to cross-spectral reconstruction and hyperspectral video coding,” *J. Opt. Soc. Am. A*, vol. 40, no. 3, pp. 479–491, Mar. 2023.
- [19] D. Scharstein and C. Pal, “Learning conditional random fields for stereo,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.