2016 15th IEEE International Conference on Machine Learning and Applications

Infrared Colorization Using Deep Convolutional Neural Networks

Matthias Limmer*, Hendrik P.A. Lensch†
*Daimler AG, Ulm, Germany
†Department of Computer Graphics, Eberhard Karls Universität, Tübingen, Germany

***Abstract*—This paper proposes a method for transferring the RGB color spectrum to near-infrared (NIR) images using deep multi-scale convolutional neural networks. A direct and integrated transfer between NIR and RGB pixels is trained. The trained model does not require any user guidance or a reference image database in the recall phase to produce images with a natural appearance. To preserve the rich details of the NIR image, its high frequency features are transferred to the estimated RGB image. The presented approach is trained and evaluated on a real-world dataset containing a large amount of road scene images in summer. The dataset was captured by a multi-CCD NIR/RGB camera, which ensures a perfect pixel to pixel registration.**

I. Introduction

In advanced driver assistance systems, cameras are often used as a sensor for object detection and augmentation (e.g. to alarm the driver of obstacles or possible threats). Near-infrared cameras have two advantages over regular RGB cameras. First, color or infrared filters are not applied to the sensor, thus not diminishing its sensitivity. Second, infrared light beams, which are invisible to the human eye, can be used to illuminate the scene in low light conditions without blinding other road users. The NIR images, produced by cameras without color filters, are grayscale. Since the IR-cut filter has been removed, they possess an appearance different from images with an IR-cut filter. Using such images in augmenting systems decreases the user acceptance, because their look does not agree with human cognition [1]. It is more difficult for users to orientate on images that lack color discrimination or contain wrong colors. Integrating a second sensor only for display purposes increases the size of the hardware components and the price of the final product. For this reason, transforming the NIR image into a natural looking RGB image, like in Fig. 1, is desirable.

Transforming a grayscale NIR image into a multichannel RGB image is closely related to *Image Colorization*, where regular grayscale images are colorized, and *Color Transfer*, where color distributions are transferred from one RGB image to another. Both techniques, however, are not simply applicable for colorizing NIR images. They often contain multiple cues, including various optimization, feature extraction and segmentation algorithms, and have certain prerequisites. Colorization, for example, leverages the fact, that the luminance is given by the grayscale input, and therefore only estimates the chrominance. NIR colorization requires estimating both the luminance and the chrominance. On the other hand, color transfer methods are often tailored to transform multi-channel input into multi-channel output. The reduced dimensionality of single-channel NIR images renders many color transfer methods ineffective because they often require inter-color distinction to produce reasonable results.

Fig. 1: An NIR image (Left) is colorized (Right) by the approach described in this paper. Best viewed in color.

This paper proposes an integrated approach based on deep-learning techniques to perform a spectral transfer of NIR to RGB images. A deep multi-scale *Convolutional Neural Network* (CNN) performs a direct estimation of the low frequency RGB values. A postprocessing step that filters the raw output of the CNN and transfers the details of the input image to the final output image is the only additional cue in the proposed approach. Sophisticated neural network architectures with a dedicated bypass path are trained on sunny summer rural road scenes. The images are from a specialized mulit-CCD camera providing pixel-to-pixel registered NIR and RGB images. Extensive numerical experiments demonstrate significantly better results, compared to existing colorization and color transfer methods, and the potential of the proposed approach.

II. Related Work

Color transfer methods analyze the color distribution of an input image and fit them to a target color distribution, taken from one or more target images. This can be done globally for the whole image [2], [3] or locally for image patches or segments [4]–[6]. Especially the approaches of [5] and [6] are showing impressive results in mapping day images to night images [5] or summer images to winter

978-1-5090-6167-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICMLA.2016.114

images [6]. Both approaches determine color transfers for small input patches by finding similar image patches in a sample image pair, which conducts a desired transformation (e.g. frames from a time-lapse video at different times of day).

Colorization methods are used for transferring colors to grayscale images. Typical approaches need to perform three main steps. First, the image is segmented into regions that are supposed to receive the same color. Second, for each region a target color palette is retrieved. Third, the retrieved color palettes for each region are used to determine their respective chrominance.

Approaches [7]–[9] use *scribbles*[1] to determine the color palette and the coarse region segmentation. Other approaches, such as [10]–[12], do not rely on user guidance. In these approaches, patches of an input image are matched to patches of a reference image or reference colorization by using feature extraction and matching. This operates fully automatically if fitting target images are given. However, it is also the greatest drawback of those approaches since a fitting reference image database and retrieval mechanism needs to be implemented.

The approach of [13] uses a *Multi Layer Perceptron* (MLP) to colorize natural images. They exploit *Scene Labels* of the used *SUN* dataset [14] as an additional feature as well as extracted *DAISY* features [15] to perform better class specific colorizations. This approach performs a fully automatic and integrated colorization, but requires scene labels as an input and these are generally not provided for arbitrary sets of images.

Recent approaches [16]–[19] leverage deep CNNs for automatic image colorization. While [16] and [17] train their networks to directly estimate chrominace values, [18] and [19] quantize the chrominance space into discrete colors and perform a logistic regression. [16], [18] and [19] initialize their networks with publicly available pre-trained models and adapt them to perform the colorization task. Contrary to [17] and [19], who introduce their own network topologies, [16] and [18] append *Hypercolumns* [20] to their topologies. All these approaches show that deep CNNs are suitable to perform automatic image colorizations. Due to their design of estimating the chrominance only, they are not directly applicable to colorize NIR images.

The approach proposed in this paper uses CNNs to perform an automatic integrated colorization from a single channel NIR image. Chrominance as well as luminance is reliably inferred for the majority of natural objects in a summerly road scenery setup. Contrary to previous approaches, though, additional complex processing cues or hand-crafted features, like scene labels, are not utilized.

The remainder of the paper is structured as follows: Section III describes the proposed approach, while extensive experiments are conducted in Section IV. The results are discussed in Section V and concluded in Section VI.

[1]Scribbles are colored strokes on an image, which mark roughly which area receives which color. They are generally given by a user.

III. Approach

The recent increase in the computational power of general purpose GPUs resulted in larger CNN architectures surpassing state-of-the-art performances in various classification tasks [21]–[25]. This indicates that inference problems such as image colorization can reach superior performance if deep CNNs are used.

This paper proposes a method using deep multi-scale CNNs to colorize infrared images. The architecture is inspired by [21] and combines it with the multi-scale scheme from [26]. While the training of the network is patch-based, the inference is image-based using techniques from [27] and [28] to preserve the image resolution.

A. Deep Multiscale CNNs

A recent trend for CNNs is the usage of many convolution layers with small convolution kernels and relatively few pooling layers. This increases the total amount of nonlinearities in the network. Furthermore, the computational complexity of each convolution layer is decreased due to small kernel sizes [21]. To increase the scale invariance of the approach proposed in this paper multiple scales [26] of the same input data are processed concurrently and lastly combined to one output (e.g. by a *fully connected* layer).

B. Framework

The proposed approach consists of three steps: First, preprocessing is performed to build a normalized image pyramid. Second, the color is inferred by using a CNN. Third, postprocessing is completed by filtering the raw output of the network and transferring the details from the input image. A schematic overview of all processing steps is depicted in Fig. 2.

C. Preprocessing

The preprocessing component is a two-step approach. First, an image pyramid of n_l levels is constructed. Every level reduces each image dimension by a factor of 0.5. Second, all pyramid levels are normalized to zero mean unit variance in a local neighborhood. Applied to all pixels of an image, this produces an image of enhanced texture. We denote the normalized image in the first level of the image pyramid by I'_1. Byproducts of the normalization are an image of local standard deviations I_σ and a mean filtered image I_μ. While I_μ can be considered the low frequency component of image I, so the Hadamard product of $I'_1 \circ I_\sigma = I_h$ can be considered the high frequency component of I.

D. Inference

Each level of the input image pyramid is fed to its own branch of the neural network. The branches are fused in a final fully-connected layer, which is also the output layer of the network. Though each branch is structurally identical,

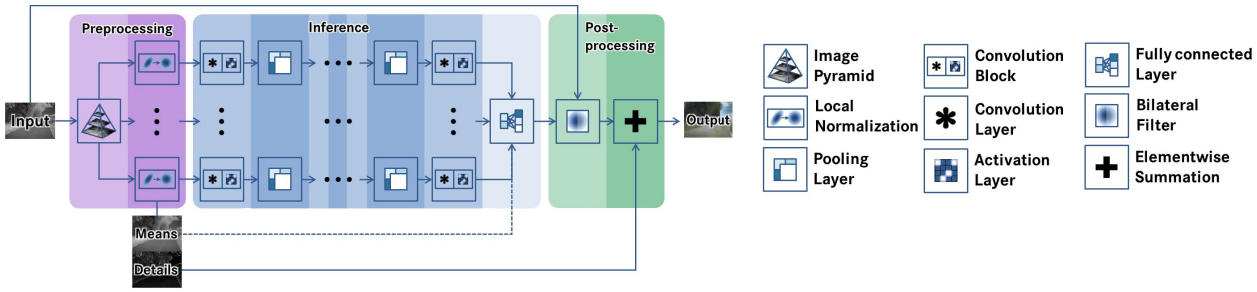


Fig. 2: A detailed view of all processing steps of the proposed approach: The colored blocks denote the various processing steps. The arrows indicate a data flow between the processing units. All processing units are depicted in a legend on the right. Convolution and activation layers are not explicitly used, because they are included in the convolution blocks. Each block consists of a fixed amount of convolution and activation layers. The amount of pyramid levels can variable and therefore also the amount of branches in the inference component.

the corresponding layers do not share their weights with each other. Each branch overall consists of n_c convolution layers and n_p max-pooling layers. The pooling layers are distributed between the convolution layers so that each *convolution layer block* has the same amount of convolution layers. The activation function of the convolution layers is the ReLU function: $\mathrm{ReLU}(x) = \max(0, x)$. The size of the filter bank n_f is the same inside each convolution layer block and is doubled after each pooling layer. The kernel size of the convolution kernels n_k is the same in all convolution layers. A peculiarity of the proposed network architecture is an optional bypass connection of the corresponding values of I_μ to the final fully connected layer. Note that a patch-based application requires an extraction of the correctly sized input patches roi_i for each scale. This is due to the reduction of resolution by using *valid convolutions*[2] and pooling layers.

E. Postprocessing

The raw output of the inference step E_μ shows visible noise, caused by inaccurate pixel-wise estimations. The subsampling property of the pooling layers combined with the correlation property of the convolution layers amplify this effect. Since source pixel locations of the subsampled and convolved coherent output feature maps are not adjacent in the original image (c.f. [27], [28]), evaluating every pixel position results in an interleaving of these separate coherent output maps. If those maps differ in local regions, the interleaving introduces a checkerboard pattern, as can be seen in Fig. 3(a). This *coherence gap* s_π between pixels of coherent maps can be calculated by building a product of all strides in the neural network $s_\pi = \prod s_{l,i}$ where $s_{l,i} \in S_l$ defines the strides of all layers in scale l. For a network, which contains three 2×2 pooling layers, the coherence gap is $s_\pi = 2^3 = 8$ in both image dimensions. Postprocessing is necessary to remove this incoherence and recover the lost details.

The postprocessing consists of two steps: First, the raw output after the inference E_μ step is filtered by a *Joint Bilateral Filter* using the *Bilateral Grid* [29]. Second, the filtered image is augmented by the high frequency details I_h from the input image to produce the final colorized image.

[2]Valid convolutions do not pad the image before convolving.

(a) Raw Output (b) Bilateral Filtered (c) With Details

Fig. 3: Results of the postprocessing steps. The checkerboard pattern is clearly visible in the raw network output (a). It is filtered by a bilateral filter (b) to remove the noise. After adding the detail image, textures and distinct object boundaries are visible (c).

The joint bilateral filter using Gaussians has two major parameters: the spatial domain standard deviation σ_g and the range domain standard deviation σ_f. While σ_g steers the spatial size of the blur, σ_f steers the sensitivity to edges in the filtering process. A large σ_g increases the area of the blur, while a small σ_g does not reduce the noise. Likewise, a small σ_f increases the edge sensitivity and a large σ_f increases the smoothing effect. The joint bilateral filter also utilizes a guidance channel on which the range domain filter is computed. In the proposed approach, the input image I is used as the guidance channel, because it contains less noise than the raw network output and all necessary surface edges to conduct an edge aware smoothing. The result after the filtering step is displayed in Fig. 3(b).

Compared to the raw output E_μ, object contours and edges are clearly visible, but the textures of the surfaces are still missing. By using the detail component of the input image I_h, textures can be partially recovered (see Fig 3(c)).

Fig. 4: Example image pairs of the used dataset. The left sides display the RGB and the right sides the corresponding NIR images.

IV. Experiments

In the following, the approach proposed in this paper is trained and evaluated on real-world images. Data acquisition has been performed using a two-CCD camera[3]. The sensor splits NIR and RGB wavelengths and distributes them to dedicated CCDs. This ensures pixel to pixel registration and temporal synchronization between the channels. Due to the application scope of the approach proposed in this paper a dataset was assembled accordingly. Video sequences were recorded during several sunny summer days in a time range of one month resulting in approximately 5 h of video material with 30 frames per second. These sequences have then been equally sampled into 38.495 image pairs and show a variation of road scenes, incorporating rural areas as well as highways. The target RGB images are white balanced and demosaiced. Finally, 32.795 image pairs were used for training and 800 for evaluation. The set of image pairs for evaluation is disjoint to those for training taken from different recording days and showing different tracks. The native resolution of the sensor is 1024×768 pixels with a bit-depth of 10 bits per pixel. Fig. 4 shows various example image pairs from the dataset.

A. Color Transfer and Colorization

Fig. 5 shows the results of various color transfer and colorization methods applied on one example from our dataset Fig. 5(a). For algorithms Fig. 5(d) - Fig. 5(f) the corresponding RGB channel Fig. 5(h) was used as the target image. Note that Fig. 5(h) would not be present in realistic applications.

Fig. 5(b) displays a user-guided colorization from [7] and Fig. 5(c) an automatic colorization from [12]. Both approaches are able to colorize the sky, but fail for trees and grass. This is mainly caused by not estimating the correct luminance.

[3]jAi AD-080CL: http://www.jai.com/en/products/ad-080cl

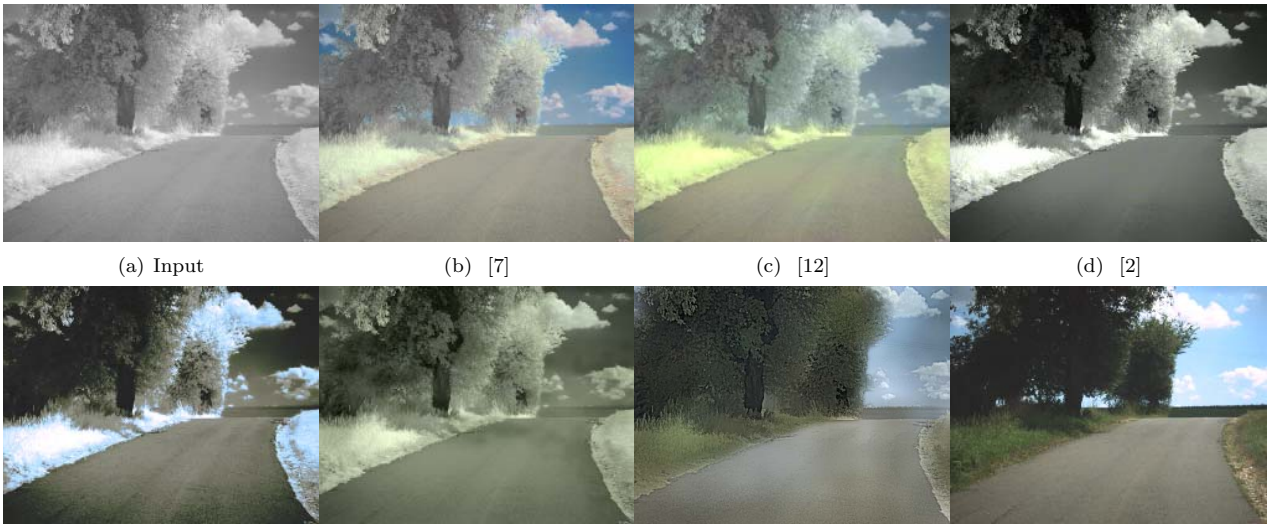
Name	n_l	n_c	n_p	bypass	roi_i
`topo-1-9-2`	1	9	2	-	46
`topo-1-9-2-bp`	1	9	2	✓	46
`topo-1-8-3`	1	8	3	-	68
`topo-1-8-3-bp`	1	8	3	✓	68
`topo-1-12-3`	1	12	3	-	98
`topo-1-12-3-bp`	1	12	3	✓	98
`topo-3-9-2`	3	9	2	-	46
`topo-3-9-2-bp`	3	9	2	✓	46
`topo-3-8-3`	3	8	3	-	68
`topo-3-8-3-bp`	3	8	3	✓	68
`topo-3-12-3`	3	12	3	-	98
`topo-3-12-3-bp`	3	12	3	✓	98

TABLE I: Network topologies, evaluated in this paper. The *bypass* column indicates topologies, where the values of I_μ are bypassed to the fully connected layer.

Fig. 5(d), 5(e) and 5(f) show the results of two global [2], [3] and one local color transfer method [5]. These approaches are not able to transfer different colors to different objects at all. They are apparently not suitable for colorizing NIR images. For comparison, the result of the approach proposed in this paper is shown in Fig. 5(g). Note that this colorization was performed without the knowledge of the target image.

B. Network Topologies

Not all parameters mentioned in Section III-B are evaluated. The kernel size n_k of the convolution layers was fixed to a small size of 3×3 pixels and the kernel size of the max pooling to 2×2 pixels. This is chosen analogous to [21], who discovered that a stack of convolution layers with small kernels benefits the classification accuracy greatly compared to singular convolution layers with big kernels. To counteract the increasing computational complexity and memory consumption of the multi-scale analysis, the filter bank of the first convolution block n_{f_1} is set to a fixed amount of 16 filters and doubled after each pooling layer. That leaves the amount of convolution layers n_c,



(a) Input (b) [7] (c) [12] (d) [2]

(e) [3] (f) [5] (g) Ours (h) Target

Fig. 5: Colorization and Color Transfer methods applied to an NIR image (a) using the corresponding RGB image (h). (b) is the result of [7], a user guided colorization method, using scribbles. (c) is the result of [12], an automatic colorization method. (d) shows the result of [2], a simple global color transfer method, while (e) shows a more sophisticated global color transfer from [3]. (f) is the result of [5], a local color transfer method. Our result is displayed in (g). It was colorized without using (h) as training data.

pooling layers n_p and scales n_l variable. We further examine the effect of bypassing the mean image I_μ as an input to the final fully connected layer. The evaluated configurations are summarized in Table I. Parameters n_c and n_p are chosen according to the following criteria. Better results are not expected from configurations smaller than `topo-1-9-2`. Configurations `topo-*-12-3` are deeper versions of `topo-*-9-2` with one additional pooling layer and convolution block per scale. Additionally, intermediary configurations `topo-*-8-3` containing 3 pooling layers and 4 convolution blocks with 2 convolution layers each were investigated. The input patch size of `topo-*-8-3`, $roi_i = 68$ lies between those of `topo-*-9-2`, $roi_i = 46$ and `topo-*-12-3`, $roi_i = 98$.

All network topologies were trained with a similar scheme. The trainable parameters Θ are initialized from a Gaussian random distribution. Stochastic gradient descent using the backpropagation algorithm [30] is performed to minimize the *mean squared error* (MSE) between the pixel estimates $\mathcal{F}(p'_i, \Theta)$ of the i^{th} normalized pixel p'_i and the corresponding pixels q_i of the mean filtered RGB image T_μ:

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{D} \sum_{i \in D} ||\mathcal{F}(p'_i, \Theta) - q_i||^2 \qquad (1)$$

where D describes the number of pixels in the dataset. In each epoch, patches of multiple random pixel locations from a random subset of images are extracted and fed to the network. The initial learning rates η for each experiment were chosen empirically by performing mini-trainings and choosing the best performing η. Then full trainings of 10000 epochs were conducted with the selected learn rates using a linear annealing and a momentum of 0.9.

Name	RMSE	S-CIELAB
`topo-1-9-2`	0.199 ± 0.049	13.36 ± 3.71
`topo-1-8-3`	0.196 ± 0.041	13.33 ± 3.19
`topo-1-12-3`	0.194 ± 0.043	12.58 ± 3.17
`topo-1-9-2-bp`	0.151 ± 0.030	10.29 ± 2.87
`topo-1-8-3-bp`	0.146 ± 0.027	9.65 ± 2.40
`topo-1-12-3-bp`	0.153 ± 0.029	10.36 ± 2.45
`topo-3-9-2`	0.167 ± 0.047	11.21 ± 3.63
`topo-3-8-3`	0.166 ± 0.047	11.64 ± 3.44
`topo-3-12-3`	0.155 ± 0.047	10.76 ± 3.31
`topo-3-9-2-bp`	0.149 ± 0.036	10.48 ± 3.08
`topo-3-8-3-bp`	0.137 ± 0.040	9.57 ± 3.25
`topo-3-12-3-bp`	**0.130** ± 0.043	**8.88** ± 3.15

TABLE II: The average RMSE and S-CIELAB [31] and their standard deviation for various network topologies. Best performing topology is `topo-3-12-3-bp`.

The topologies from Table I have been evaluated with respect to the RMSE and *S-CIELAB* [31] image quality measures. The S-CIELAB is a subjective measure, designed for measuring the image quality from a human perspective. The raw network outputs of the 800 images from the evaluation dataset are individually evaluated by both measures before any postprocessing has been applied. Table II displays the average values and standard deviations of the evaluations for each topology. Fig. 6 shows the evaluation results of the RMSE graphically. The solid lines display the medians and the transparent area their respective interquartile distance. The best performing network architecture for both measures is `topo-3-12-3-bp`. It has

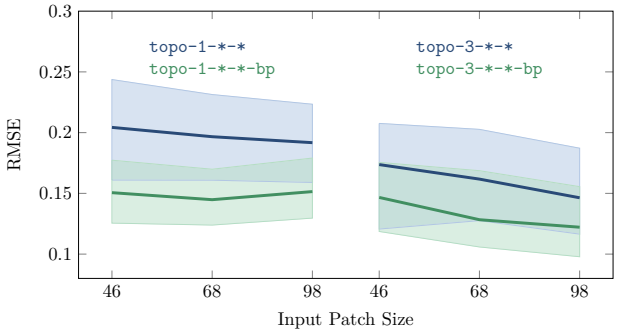


Fig. 6: Medians (solid line) and interquartile range (transparent area) of the RMSE over the evaluation dataset for different network topologies.

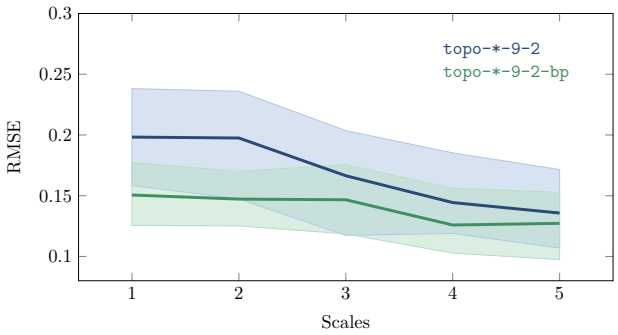


Fig. 7: Medians (solid line) and interquartile range (transparent area) of the RMSE over the evaluation dataset for network topologies `topo-*-9-2` and `topo-*-9-2-bp` for different scales.

the biggest input patch size, 3 scales and the bypass path. Although `topo-1-12-3-bp` is behaving otherwise, network architectures tend to perform better if their input patch size increases.

This also seems to be the case for topologies with more scales. Additional topologies (`topo-[1..5]-9-2` and `topo-[1..5]-9-2-bp`) were trained to examine the influence of the amount of scales to the performance. Fig. 7 shows that an increase of scales is beneficial to the performance up to a certain threshold. More than 4 scales result in almost no improvement (cf. `topo-5-9-2`) or even slight loss of performance (cf. `topo-5-9-2-bp`). The reason for that behavior is the final resolution of scale 5. From an initial resolution of 1024×768 pixels of the input image, scale 5 has a resolution of 64×48 pixels. With a patch size $roi = 46$, the image of scale 5 is almost completely contained by the patch. In that case, scale 5 cannot provide any useful information, which benefits a distinguished inference for different pixel locations.

A positive effect, though, can be recognized for all topologies when using the bypass path. It seems that the mean filtered image I_μ serves as a prior to T_μ, because it contains information that is not present in the normalized input images.

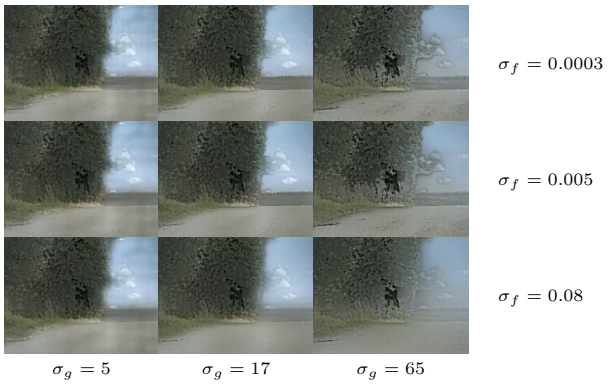


Fig. 8: The effect of different parameterizations of the bilateral filter on the output image of `topo-3-12-3-bp` after adding the details. The rows show the effect of different range parameters σ_f, while the columns show the effect of different spatial parameters σ_g.

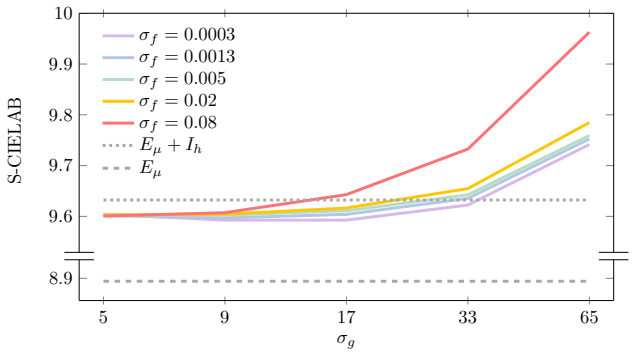


Fig. 9: The average S-CIELAB errors over all evaluation images of `topo-3-12-3-bp` for the parameters (σ_g, σ_f) of the bilateral filter after adding the detail layer I_h. Each graph displays the result of one σ_f in relation to various σ_g. For comparison, E_μ (dashed line) displays the error of the raw CNN-output and $E_\mu + I_h$ (dotted line) the error of the raw output after adding the high frequency details.

C. Postprocessing

In the postprocessing step, the parameterization of the bilateral filter has a great impact on the final visual appearance. Fig. 8 shows the visual influence of various instances of σ_g and σ_f after details have been added to the estimated image. The column on the left (all with $\sigma_g = 5$) still shows strong noise deriving from the coherence gap. The column on the right (all with $\sigma_g = 65$), however, shows a severe *color bleeding* effect resulting from the great spatial blur of the bilateral filter. The range parameter σ_f, though, appears to have a minor effect on the final result. This is reflected in Fig. 9, which displays the average S-CIELAB error over the evaluation dataset for `topo-3-12-3-bp` in relation to the parameters of the bilateral filter. The average error of E_μ (dashed line) is increased by adding the detail layer I_h (dotted line), since and addition of details was not considered in the training of the CNN. The bilateral filter, though, is able to reduce this increased error, by choosing σ_g and σ_f wisely. Fig. 9 shows that small σ_f perform better, but keep the error

Fig. 10: NIR images (Left) are colorized (Middle) in comparison to their target images (Right). The used topology is `topo-3-12-3-bp` with $\sigma_g = 17$ and $\sigma_f = 0.005$.

in the same range for reasonable σ_g. Values of $\sigma_g \leq 17$ are mainly smoothing the noise induced by the coherence gap for this topology. Values of $\sigma_g > 17$ start to have increasingly negative effects, because the smoothing of bigger areas result in a color bleeding effect. In these cases the error surpasses the error of $E_\mu + I_h$.

V. Discussion

The proposed approach is able to colorize NIR images of summerly road scenes fully automatically by leveraging the advantages of a deep neural network. It cannot reconstruct all information that is not included in single channel NIR images, though. Many artificial objects, such as cars, buildings, etc. are falsely colorized because their appearance does not correlate with a specific color (c.f. Fig. 11(a)). There are also items that are invisible in the NIR image due to the filtering of the RGB wavelengths. Fig. 11(b) shows an example, where the green light of an LED traffic light is absent from the NIR image. The induction of model information including additional scene label features might help in some cases, but effects, such as the missing signal of a traffic light, are not recoverable.

VI. Conclusion and Future Work

This paper presented an integrated approach to transfer the color spectrum of an RGB image to an NIR image. The transfer is performed by feeding a locally normalized image pyramid to a deep multi-scale CNN, which directly estimates RGB values. Using the mean filtered input image as an additional input to the final fully connected layer improves the performance greatly. The resulting raw output is then joint-bilaterally filtered using the input image as a guidance map. The details of the input image are added at the end to produce a naturally-colorized output image. The approach is only failing to colorize objects correctly, where object appearance and color do not correlate.

Future work will incorporate the extension of the training and evaluation dataset to other seasons of the year. This might result in training distinct colorizers for each of these seasons. To improve realism of the colorized image textures, future work will also contain the estimation of a correct detail layer. Changing the loss function to support multi-modal estimations in combination with a semantic segmentation might increase the vibrancy of objects that our algorithm is failing to colorize.

(a) The color recovery for a truck fails.

(b) The green lamp of an LED traffic light is not perceived by the NIR channel and is therefore not colorized.

Fig. 11: Situations, where the approach shows limitations.

Acknowledgment

Authors would like to thank Markus Thom and Roland Schweiger for valuable advises and input.

References

[1] A. Toet and M. A. Hogervorst, "Progress in color night vision," *Optical Engineering*, vol. 51, no. 1, 2012.

[2] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34 –41, 2001.

[3] F. Pitié and A. Kokaram, "The linear monge-kantorovitch linear colour mapping for example-based colour transfer," in *European Conference on Visual Media Production*, 2007.

[4] B. Wang, Y. Yu, and Y.-Q. Xu, "Example-based image color and tone style enhancement," in *Proceedings of ACM SIGGRAPH*, ser. SIGGRAPH '11. New York, NY, USA: ACM, 2011, pp. 64:1–64:12.

[5] Y. Shih, S. Paris, F. Durand, and W. T. Freeman, "Data-driven hallucination of different times of day from a single outdoor photo," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 200:1–200:11, 2013.

[6] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays, "Transient attributes for high-level understanding and editing of outdoor scenes," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[7] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 689–694, 2004.

[8] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1120–1129, 2006.

[9] B. Sheng, H. Sun, S. Chen, X. Liu, and E. Wu, "Colorization using the rotation-invariant feature space," *IEEE Computer Graphics and Applications*, vol. 31, no. 2, pp. 24–35, 2011.

[10] G. Charpiat, M. Hofmann, and B. Schölkopf, "Automatic image colorization via multimodal predictions," in *Proceedings of the European Conference on Computer Vision*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Marseille, France: Springer, 2008, pp. 126–139.

[11] J. Pang, O. Au, K. Tang, and Y. Guo, "Image colorization using sparse representation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1578–1582.

[12] A. Deshpande, J. Rock, and D. Forsyth, "Learning large-scale automatic image colorization," in *Proceedings of the International Conference on Computer Vision*, 2015.

[13] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the International Conference on Computer Vision*, 2015.

[14] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2012.

[15] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2008.

[16] D. Ryan, Tech. Rep., 2016. [Online]. Available: http://tinyclouds.org/colorize/

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *Proceedings of ACM SIGGRAPH*, vol. 35, no. 4, 2016.

[18] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," Tech. Rep. arXiv:1603.06668, 2016.

[19] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," Tech. Rep. arXiv:1603.08511, 2016.

[20] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2015, pp. 447–456.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2014, arXiv:1509.01951.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Tech. Rep. arXiv:1512.03385, 2015.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2015.

[24] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," Tech. Rep. arXiv:1511.00561, 2015.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2015.

[26] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[27] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *Proceedings of the IEEE International Conference on Image Processing*, 2013, pp. 4034–4038.

[28] M. Thom and F. Gritschneder, "A theory for rapid exact signal scanning with deep multi-scale convolutional neural networks," Tech. Rep. arXiv:1508.06904, 2016.

[29] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278 –2324, 1998.

[31] X. Zhang, B. A. Wandell, and B. A. W, "A spatial extension of cielab for digital color image reproduction," 1996.