Self-Supervised Continuous Colormap Recovery from a 2D Scalar Field Visualization without a Legend

Hongxu Liu, Xinyu Chen, Haoyang Zheng, Manyi Li, Zhenfan Liu, Fumeng Yang, Yunhai Wang, Changhe Tu, and Qiong Zeng

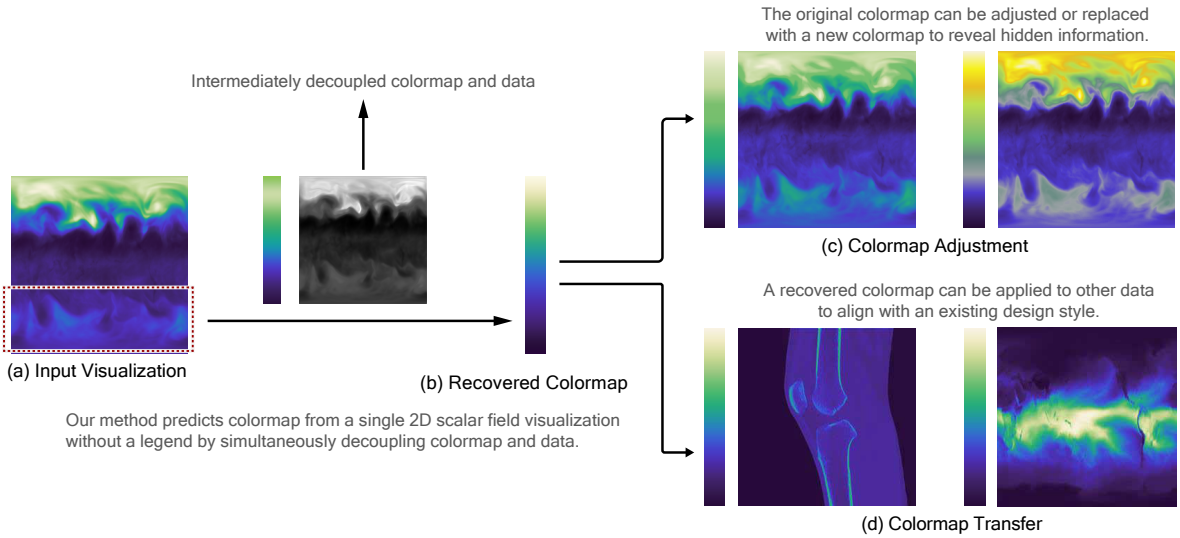


Fig. 1: **Colormap recovery from a single visualization**. (a) shows a visualization with subtle variations in the red dotted region. (b) Our method successfully predicts the original colormap by simultaneously decoupling colormap and data embedded in the visualization. (c) Replacing the original colormap with two newly designed colormaps better reveals spatial variations in the highlighted region. (d) The recovered colormap can be applied to new data for a consistent design style.

Abstract—Recovering a continuous colormap from a single 2D scalar field visualization can be quite challenging, especially in the absence of a corresponding color legend. In this paper, we propose a novel colormap recovery approach that extracts the colormap from a color-encoded 2D scalar field visualization by simultaneously predicting the colormap and underlying data using a *decoupling-and-reconstruction* strategy. Our approach first separates the input visualization into colormap and data using a decoupling module, then reconstructs the visualization with a differentiable color-mapping module. To guide this process, we design a reconstruction loss between the input and reconstructed visualizations, which serves both as a constraint to ensure strong correlation between colormap and data during training, and as a self-supervised optimizer for fine-tuning the predicted colormap of unseen visualizations during inferencing. To ensure smoothness and correct color ordering in the extracted colormap, we introduce a compact colormap representation using cubic B-spline curves and an associated color order loss. We evaluate our method quantitatively and qualitatively on a synthetic dataset and a collection of real-world visualizations from the VIS30K dataset [9]. Additionally, we demonstrate its utility in two prototype applications—colormap adjustment and colormap transfer—and explore its generalization to visualizations with color legends and ones encoded using discrete color palettes.

Index Terms—Colormap, color design, 2D scalar field visualization, reverse engineering

1 Introduction

Color mapping is the process of assigning colors from a continuous colormap to data values through linear or non-linear mapping, creating a visualization where each color represents one or more data values. In this process, the continuous colormap is typically generated by linearly or non-linearly interpolating a sequence of colors. Color mapping is a powerful tool in scientific visualization (e.g., meteorology [61] and fluid dynamics [18]), enabling researchers to effectively represent 2D scalar fields and facilitate downstream analysis [12, 61].

The effectiveness of color mapping depends heavily on a number of design considerations, including data distribution, task requirements, and color choices [34,46,59,64]. Poorly designed colormaps can lead to misinterpretation [11], such as the rainbow colormap exaggerating data variations due to its abrupt hue changes [51]. This issue is prevalent in many real-world visualizations, including those found in published papers, online repositories, and informal reports [44], where the original data are not provided. For instance, in medical visualizations such as MRI imaging [51], a rainbow colormap can cause moderate values to appear as extreme due to perceptual discontinuities, misleading viewers and potentially affecting decision-making. Similarly, spatial patterns in the lower region of Fig. 1(a) are obscured. Moreover, many real-world visualizations fail to consider color vision deficiencies. Colormaps that rely on red-green contrasts, for example, can render important features invisible to users with red-green color blindness [3, 6, 40], thus limiting accessibility. These challenges highlight the need for effective colormap design and, more broadly, for techniques that can recover or replace poorly chosen colormaps in visualizations.

- *Hongxu Liu, Xinyu Chen, Haoyang Zhang, Manyi Li, Zhenfan Liu, Changhe Tu, and Qiong Zeng (corresponding author) are with Shandong University. E-mails: {liuhongxu_0227, zhenghaoyang1229, liuzhenfannn}@163.com, xinyu.chen0468@gmail.com, and {manyili, chtu, qiong.zn}@sdu.edu.cn.*
- *Fumeng Yang is with University of Maryland. E-mail: fy@umd.edu.*
- *Yunhai Wang is with Renming University of China. E-mail: cloudseawang@gmail.com.*

Colormap reverse engineering provides a potential solution to improve real-world visualizations [43, 44, 54, 63]. Its aim is to identify the colormap used and recover the mapping between colors and data values. This process typically relies on a color legend associated with the visualization to reconstruct the color-to-data mapping. The main challenge in these cases is to accurately detect colors and text within the legend using image processing and optical character recognition (OCR) techniques [10,43,44]. However, many real-world visualizations, particularly those in books or on web pages, lack a color legend [63], making this reverse engineering even more difficult. In such cases, accurate color extraction and correctly assigning data values to those colors are essential, making this a fundamentally underconstrained problem.

Several methods have been proposed to extract discrete color palettes from visualizations without a legend. For instance, some approaches calculate categorical similarity or spatial distance between colors, and use heuristic priors or clustering algorithms to group similar colors [8, 42, 62]. In a different approach, Yuan *et al.* [63] introduced a convolutional neural network that extracts continuous colormaps under the supervision of paired colormap and visualization examples. While it incorporates post-processing techniques such as Laplacian smoothing to refine extracted colormaps, such steps may introduce inaccuracies or noise. Additionally, Yuan *et al.*'s approach struggles with out-of-distribution visualizations—ones with color distributions different from the training data, potentially resulting in incorrect or artificial colors.

Different from previous methods, we propose an end-to-end **self-supervised colormap recovery network** that directly learns continuous colormaps from single visualizations without an accompanying legend. Our approach leverages the intrinsic relationship between visualizations, colormaps, and data, following a *decoupling-and-reconstruction* strategy. For simplicity and applicability, we focus on linear mapping between colormaps and data among different mapping strategies [13, 58, 60]. Our network operates in two stages. First, **an auto-encoder** decomposes the input visualization into a decoupled colormap and data. Second, **a differentiable colormapping module** reconstructs the visualization using the recovered colormap and data. The network is trained in a supervised manner, where the training data consists of pairs of visualizations, their corresponding colormap, and data. A *reconstruction loss* between the original and reconstructed visualizations is used to guide the training optimization process. This also allows the network to fine-tune the recovered colormap during inferencing with the input of a single visualization. To guarantee colormap smoothness, we use a compact representation based on cubic B-spline curves [7, 41]. To preserve the color order, we also introduce a color order loss that penalizes random color ordering. By explicitly decoupling the colormap from the data and incorporating a self-supervised fine-tuning strategy, our method achieves robust colormap recovery even for out-of-distribution visualizations.

We compare our method to the state-of-the-art colormap recovery method [63] quantitatively and through a perceptual study, based on synthetic and real-world visualizations. We generate the synthetic visualizations using colormaps from various sources [37, 46, 48, 63] to encode data from mathematical functions [37] and the ECMWF public datasets [1]. The real-world visualizations came from the VIS30K dataset [9]. We also conduct an ablation study to verify the importance of our core *decoupling-and-reconstruction* components, including the cubic B-spline colormap representation, colormap recovery loss functions, and the fine-tuning procedure during inference. Finally, we show how to extend our approach to general visualizations with legends or discrete color palettes, and present the practical value of our method through two prototype applications: colormap adjustment and colormap transfer. Our main contributions are as follows:

- We propose **a decoupling-and-reconstruction network** using two design considerations extracted from the literature; our network supports self-supervised fine-tuning for colormap recovery from single visualization images lacking a legend.
- We introduce **a compact colormap representation** using cubic B-spline curves and a color order loss to ensure a visually smooth colormap with appropriate ordering. The use of cubic B-spline curves results in smoother transitions between colors, achieving at least a 29.6% lower mean squared error (MSE) compared to traditional linear colormap representations.
- We show **the effectiveness** of our proposed network. Our model outperforms baseline approaches with an MSE improvement of at least 19.1%; it received a 26.5% higher rating in a perceptual study concerning the visual quality of recovered colormaps.
- We demonstrate the practical usage of our approach in two application prototypes—**colormap adjustment** and **colormap transfer**—and extend it to accommodate general visualizations.

To facilitate future research, we have open-sourced our data, code, models, and interactive tool at `https://osf.io/gb2tx/?view_only=4d2a269b59bc4144b2806ec2d1a34e11`.

2 Related Work

Considerable efforts in the visualization community have been made to improve the effectiveness and efficiency of color design. For a complete investigation of color design in visualization, we refer readers to prior surveys [5, 56, 66]. In this section, we review colormap generation and optimization, colormap extraction, and image recoloring approaches.

2.1 Colormap Generation & Optimization

Common color design considerations, such as color name distance [22, 47] and perceptual uniformity [7], can either be directly modeled as colormap generation rules, or integrated with data distributions and visualization tasks [19,31,36,39,57]. For example, Gramazio *et al.* [19] provided a color design tool, *Colorgorical*, which incorporates user-assigned discriminability and preference into the process of color generation. Smart *et al.* [57] learned data-driven models to fit designer-crafted colormaps, and leveraged those models to generate an effective color palette from a single seed color. Lu *et al.* [31] integrated color generation and color assignment for visualizing categorical data, with color names and color discriminability as optimization constraints.

Another common challenge in color design is to improve a colormap. Automatic colormap optimization algorithms are used to enhance the original colormap to meet specific design considerations, e.g., colormap properties, data distributions, or task requirements [38, 40, 58, 59, 64]. For example, Thompson *et al.* [58] generated and adjusted colormaps to highlight prominent values based on heuristic data analysis rules. Nuñez *et al.* [40] enhanced colormaps while maintaining perceptual uniformity and linearity, especially for viewers with defective color vision. Tominski *et al.* [59] and Zeng *et al.* [64] adjusted control point positions within colormaps to better match the colormap to data histograms or spatial data variations. Additionally, Nardini *et al.* [38] presented a framework for local and global colormap optimization, adhering to colormap properties such as uniformity and color order.

The colormaps resulting from these methods are used to *forward* encode data in visualizations, typically presented with a legend. However, many real-world visualizations lack such contextual information, making them difficult for users to interpret or reuse. In this work, we reverse-engineer colormaps from real-world visualizations without legends, and provide applications (e.g., colormap adjustment and colormap transfer) that enable users to interactively explore the underlying colormap embedded within the visualization images.

2.2 Colormap Extraction

Extracting visual specifications from visualization images is a typical reverse engineering process. A visualization image is used as input, and the extracted visual specifications and data values are the output. The key challenge here is to recognize chart elements (e.g., text, characters, and words in bar charts [24, 32, 33, 35, 43, 54]) and infer their relationships. For example, Poco and Heer [43] used text analysis to infer textual elements and deduce their correlations with graphical mark types. Luo *et al.* [32] unified deep neural networks and rule-based methods in a single framework, which includes key point and chart type detection, data range extraction, and chart object detection. However, these methods primarily address simple and discrete charts, without considering continuous color mappings in scalar field visualizations.

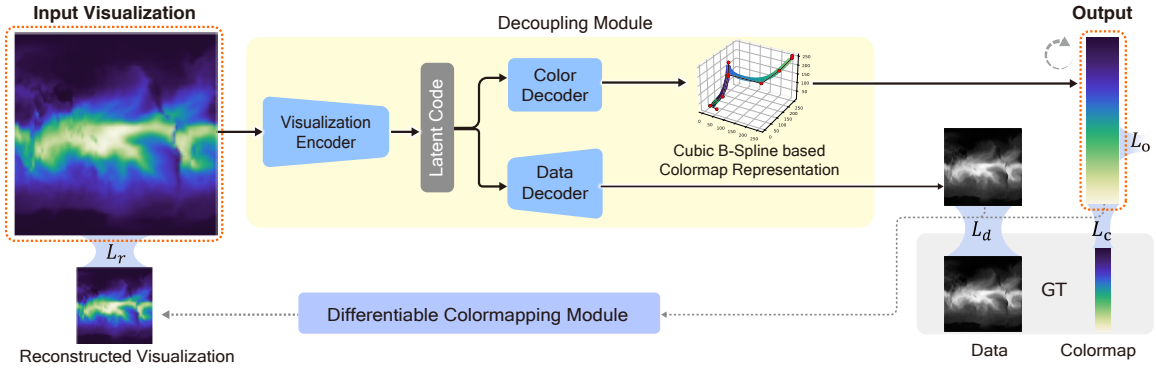


Fig. 2: **Overview of our colormap recovery network**. It consists of a *decoupling module* and a *differentiable colormapping module*. The decoupling module includes a visualization encoder to represent the input as a latent code, a color decoder to recover the colormap, and a data decoder to recover the underlying data. To ensure accurate colormap recovery, we employ four loss functions: a **color fidelity loss** L_c to measure differences between the recovered and an original colormap, a **color order loss** L_o to promote visually smooth color transitions, a **data loss** L_d to measure differences between the recovered data and the ground-truth data, and a **reconstruction loss** L_r to assess the similarity between reconstructed and input visualizations. During training, the network is trained on paired visualizations with known colormaps and data. During inferencing, the network operates on a single visualization without the latter information.

One line of exploration is to incorporate additional information into visualization images to facilitate data recovery. For example, Zhang *et al.* [65] embedded visual information into QR codes and placed them in unimportant areas of the visualization image. Fu *et al.* [17] encoded subtle chart information (e.g., coarse and fine marks) and concealed them using carefully designed opacity. However, these methods rely on altering the current design of visualizations and cannot be directly applied to existing visualizations.

The other line of work focuses on colormap recovery. Poco *et al.* [44] proposed a semi-automatic method to extract colormaps. It first detects and classifies the legend as either discrete or continuous, then uses OCR techniques to identify colors and extract the accompanying text. The extracted color and labeled text are further integrated to predict the colormap. However, this method assumes the visualization includes an accurate legend, while many real-world visualizations lack well-defined legends, have incomplete or misleading labels, or use inconsistent color mappings. In such cases, legend-based extraction might result in interpretation errors, highlighting the need for colormap recovery algorithms in visualizations without legends.

Our work focuses on inferring color mappings directly from a single visualization lacking a legend. Yuan *et al.* [63] shared similar goals to ours—recovering colormaps from single visualizations without legends. Their method extracts colormaps from a given visualization image using supervised deep neural networks trained on paired colormaps and visualizations. The process is followed by post-processing operations, including clustering for discrete colormaps and Laplacian smoothing for continuous ones, to enhance the quality of the extracted colormaps. However, their method has two major limitations. First, it separates the deep colormap extraction from the post-processing steps, which can lead to inaccurate colors and the loss of mappings between the original colormap and data. Furthermore, the approach relies on paired datasets and lacks mechanisms to ensure robust performance on unseen visualizations with color distributions different from those in the training dataset (as discussed further in Sec. 5.3).

2.3 Visualization Recoloring

Research on visualization recolorization directly enhances input visualizations without first recovering colormaps [14, 67–69]. For example, Elmqvist *et al.* [14] enhanced initial colors in user-defined regions of interest and applied a modified colormap to the data in those regions, but this method requires access to the original data. Zhou *et al.* [67] sharpened color-mapped visualizations by compensating the power spectrum for viewing distance. Following this work, Zhou *et al.* [68] further enhanced colors for high-dynamic range data by using tone mapping operators and overlaid glare for highlighting. The tone mapping operators preserved relationships between original data values and structures, while compressing the high-dynamic range into a suitable data range. However, these methods enhance the original colormap without recovering the original colors, limiting their applicability in restyling and editing. It is worth noting that some image style transfer methods treat color recovery as an intermediate step, recoloring images based on the semantic relationships between the recovered color and reference palettes [8]. However, these methods mainly focus on preserving semantic correlations between images and may distort the underlying data in visualizations.

3 Reverse-Colormapping Overview

3.1 Design Considerations

The color mapping process typically assigns colors from a continuous colormap to data values, either linearly or non-linearly [60, 64]. Such colormaps are usually constructed by interpolating a set of predefined colors, known as control points [60, 64]. To simplify the colormap recovery problem, we focus on *linear mapping* between data values and colors. Building on the color mapping foundations and insights of color design from the literature [7, 38, 39, 43, 44, 54, 64], we define two core requirements (CR) that our framework must satisfy for effective colormap recovery.

- **CR1. High fidelity**. The recovered colormap and data should faithfully reflect the underlying color distribution, data patterns, and information encoded in the original visualization.
- **CR2. Visual clarity**. The recovered colormap should exhibit smooth color transitions and maintain consistent sorting order. This ensures clear visual communication and avoids introducing artifacts that could hinder data interpretation.

3.2 Overview

Motivated by the guidelines identified above, we propose to recover colormaps from single 2D scalar field visualizations by simultaneously decoupling the colormap and data, ensuring that the newly reconstructed visualization closely resembles the input. The colormap recovery procedure involves estimating a continuous colormap for unknown data based solely on a set of colors given in the input visualization. This task is inherently underspecified, resulting in an infinite number of possible solutions. To address this issue, we propose to *simultaneously decouple the colormap and the data*, while *ensuring visual consistency* between the reconstructed and original visualizations using an end-to-end deep neural network (see Sec. 4). While our approach focuses on continuous colormaps, it can be easily extended to discrete colormaps with minimal adjustments (see Sec. 6.2).

4 Reverse-Colormapping Network

Our network employs a *decoupling-and-reconstruction* strategy to separate the colormap information from the underlying data within a 2D scalar field visualization, which includes a *decoupling module*

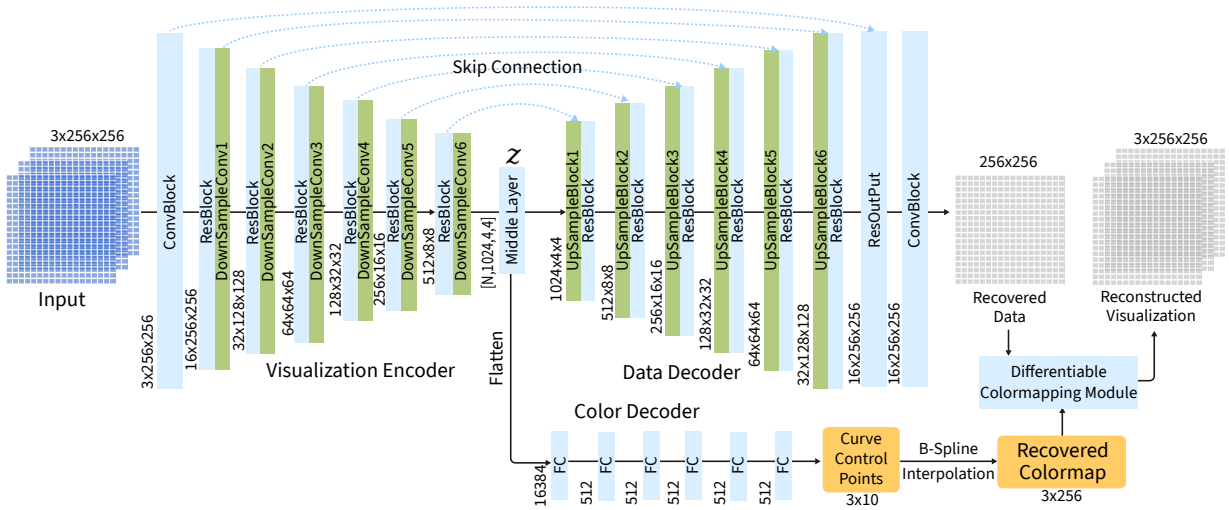


Fig. 3: **Network architecture of our decoupling-and-reconstruction network**.The visualization encoder and data decoder are implemented as a convolutional U-Net with skip connections [52]. The color decoder is an MLP network which outputs color curve control points, which are then transformed as a colormap and used to recover the visualization.

(see 4.2.2) and a *differentiable colormapping module* (see 4.2.3). Figure 2 shows the framework.

4.1 Mathematical Formulation

Given an input visualization $\boldsymbol{I} \in \mathbb{R}^{M\times H\times W}$ (where M is the number of visualization channels, H is the height, and W is the width), the goal of the colormap recovery network is to recover both the original colormap $\Xi \in \mathbb{R}^{n+1}$ (where $n+1$ is the number of control points in the colormap) and the underlying data $\boldsymbol{D} \in \mathbb{R}^{H\times W}$. The input visualization $\boldsymbol{I}$ is passed through the visualization encoder to produce a latent code $\boldsymbol{z}$, which is then fed into two separate decoder branches, satisfying **CR1**. The color decoder decouples $\boldsymbol{z}$ and recovers a set of control points $\hat{\boldsymbol{C}}$, which are used to generate a smooth colormap $\hat{\Xi}$ with a cubic B-spline interpolating function $\boldsymbol{S}$ (in line with **CR2**). The data decoder decouples $\boldsymbol{z}$ to recover the underlying data $\hat{\boldsymbol{D}}$. Then, the recovered colormap $\hat{\Xi}$ and data $\hat{\boldsymbol{D}}$ are fed into the *differentiable colormapping module* to reconstruct the new visualization $\hat{\boldsymbol{I}}$. Following the design guidelines, we define the total loss L as the sum of four loss components: a *color fidelity loss* L_c to measure differences between the recovered and original colormap, a *color order loss* L_o to penalize disorder in the colormap, a *data loss* L_d to encourage the recovered data to faithfully represent the underlying data, and a *reconstruction loss* L_r to measure the difference between the reconstructed and original visualizations.

To achieve our goal, we formulate the training objective for the colormap recovery network as the following optimization function:

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}(L(I,\Xi,D;\hat{I},\hat{\Xi},\hat{D})), \tag{1}$$

where θ represents the parameters of the network, $\hat{\theta}$ indicates the optimal parameters. $\mathbb{E}$ denotes that the total loss L is averaged over the entire training set. During inference, $\hat{\theta}$ are fine-tuned in a self-supervised manner to recover colormaps for single visualizations without access to the original colormap Ξ and data D.

4.2 Architecture

4.2.1 Colormap Representation

In our framework, a colormap Ξ is defined as a sequence of *control points* and an *interpolating function* [64]. Control points are represented as an array of $n+1$ color values in RGB color space[1]: $\boldsymbol{C} = \{\boldsymbol{c_0},\ldots,\boldsymbol{c_n}\}$, with all color components normalized to the range $[0,1]$. To ensure visually smooth color transitions between control points, we use the cubic B-spline interpolating function [7, 41] rather than traditional piecewise linear interpolation [64]. The cubic B-spline interpolating function is described as follows:

$$\boldsymbol{S}(t) = \sum_{i=0}^{n} N_{i,3}(t)\boldsymbol{c_i}. \tag{2}$$

Here, $n+1$ gives the number of control points (in our case $n=9$), and $S(t)$ indicates the interpolated colors at parametric position t, as determined by the colors of the control points and the standard B-spline basis function $N_{i,j}(t)$ (where i and j are indices). The values t_i represent parametric positions of the knots of a cubic B-spline curve, arranged in increasing order from $i=0$ to $n+4$.

4.2.2 Decoupling Module

Our network architecture consists of a visualization encoder, a color decoder, and a data decoder. Below we describe the network details, illustrated in Fig. 3.

Visualization Encoder. The visualization encoder E aims to extract a latent code $\boldsymbol{z}$ that captures the essential features of the input visualization $\boldsymbol{I}$. It begins by applying a single convolution block (denoted *ConvBlock* in Fig. 3) to extract low-level features from the original visualization. Then, six downsampling modules, each including a residual block (denoted as *ResBlock*) and a downsampling block implemented as convolution layers (denoted as *DownSampleConv*), are sequentially applied to compress the feature map. The final component is an intermediate layer, which is composed of a single ResBlock. Note that all convolution layers in the visualization encoder have 3×3 kernels.

Color Decoder. The color decoder, implemented as a multi-layer perceptron (MLP), is designed to decode the latent code $\boldsymbol{z}$ into a visually smooth colormap. This process consists of two main steps:

- **Generating control points**: The MLP outputs a set of control points in RGB color space. This is achieved through six blocks, each containing a fully connected layer, a batch normalization layer, and an activation layer. The default number of control points is set to 10, so the MLP produces a 10×3 array of RGB values (10 control points, each with three color channels).
- **B-Spline Interpolation**: A cubic B-spline interpolation function is applied to generate the recovered colormap $\hat{\Xi}$ by smoothly interpolating between the estimated control points, ensuring seamless color transitions.

Data Decoder. The data decoder aims to recover the underlying data from the latent code $\boldsymbol{z}$. This auxiliary information serves to enhance the quality of the extracted colormap. The decoder consists of six

[1]We trained our network in both the RGB color space and the perceptually uniform CIELAB color space, but chose RGB due to its widespread use in deep neural networks and its superior performance in our task (see Sec. 5).

upsampling modules, each containing an upsampling block (denoted as *UpSampleBlock*) with a convolutional layer and a residual block with two stacked convolutional layers. The upsampling block increases the spatial dimension of the latent data representation, incorporating skip connections from the visualization encoder.

In summary, building upon the U-Net architecture [52], our network introduces three key modifications: (1) decoupling color and data into separate decoder branches, (2) incorporating residual blocks [21] for better detail preservation, and (3) replacing max pooling with convolutional layers for downsampling to improve feature extraction.

4.2.3 Differentiable Colormapping Module

Ideally, if the recovered colormap and data are consistent, the reconstructed visualization should closely resemble the input visualization. Based on this assumption, we define a *differentiable colormapping module* that reconstructs a visualization from the recovered colormap and data, following the colormapping procedure outlined in Sec. 3. Since the module is differentiable, we can iteratively fine-tune the recovered colormap and data through backpropagation [53] during training or inference, guided by the similarity between the reconstructed visualization and the ground-truth. This adaptability allows the module to effectively perform self-supervised learning, especially for out-of-distribution visualizations where no paired colormap and data are available in advance (see Sec. 4.4).

4.3 Loss Function

Motivated by the design guidelines, we introduce four loss functions to optimize our network. The color fidelity loss, data fidelity loss, and reconstruction loss ensure that the recovered colormap and data preserve high fidelity to the original visualization. Additionally, the color order loss encourages smooth and perceptually ordered color transitions in the recovered colormap.

Color Fidelity Loss. The color fidelity loss encourages similarity between the recovered colormap and the ground truth (GT) colormap. It is calculated by measuring the mean squared error (MSE) between the GT colors and the ones in the recovered colormap; the latter is generated using a cubic B-spline and the control points predicted by the color decoder (see Sec. 4.2).

$$L_c = \frac{1}{m}\sum_{i=1}^{m} ||\mathbf{\Gamma}(i) - \hat{\boldsymbol{S}}(t(i))||, \quad t(i) = \frac{i}{m}. \tag{3}$$

$\mathbf{\Gamma}(i)$ denotes the i-th color in the GT colormap. $\hat{\boldsymbol{S}}$ represents the recovered colormap. Here, $t(i)$ represents the parametric position on the cubic B-spline curve corresponding to the i-th color index. m signifies the number of colors sampled from the color curve, typically set to 256 to indicate a colormap with 256 colors. $||\cdot||$ denotes the difference between two colors, calculated using the Euclidean distance.

Color Order Loss. To guarantee a natural color order, we introduce a color order loss, inspired by the global legend-based color order [7, 38]. The core principle of the global legend-based order is to ensure that every color pair in a colormap is invertible and distinguishable. We achieve this by calculating the *order ratio* of color differences between each color pair in the recovered colormap to their spatial distance along the colormap. A higher *order ratio* signifies a better color order, where colors farther apart spatially exhibit more pronounced visual distinction. As neural networks typically minimize loss functions, we use the *negative* of the order ratio as our color order loss.

$$L_o = -\min_{i\neq j\in 1,\ldots,m} \frac{||(\hat{\boldsymbol{S}}(t(i)) - \hat{\boldsymbol{S}}(t(j))||}{(i-j)^2}, \tag{4}$$

where i and j represent the indices of two distinct colors within the recovered colormap. Fig. 4 shows the *order ratio* between every color pair in the recovered colormaps before and after fine-tuning.

Data Fidelity Loss. To encourage the recovered data to closely resemble the GT data, we introduce a data fidelity loss, which minimizes

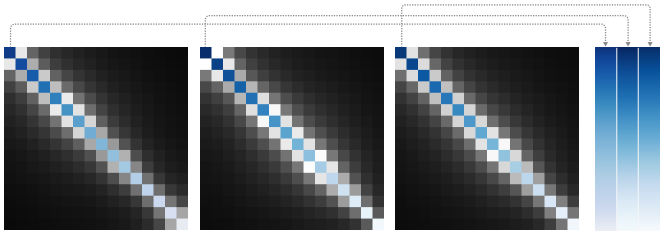


Fig. 4: **Color order**. (a–c) show the *order ratio* between every distinct pair of colors, while the corresponding recovered colormaps are shown side-by-side on the right. (a) The recovered colormap before fine-tuning; black to white indicates increasing color order ratios. (b) The recovered colormap after fine-tuning: the lower right of the matrix shows similar color order values to those in the ground truth colormap in (c).

average differences between the scalar values predicted by our model and the corresponding values in the GT data. It is defined as follows:

$$L_d = \frac{1}{WH}\sum_{h=1}^{H}\sum_{w=1}^{W} (D(h,w)) - \hat{D}(h,w))^2, \tag{5}$$

where H and W represent the height and width of the input. $D(h,w)$ and $\hat{D}(h,w)$ denote the normalized scalar value (ranging from 0 to 1) at position (h,w) in the GT data and the recovered data respectively.

Reconstruction Loss. To guarantee faithful reconstruction of the color mapping, we introduce a reconstruction loss function. This loss minimizes average differences between corresponding pixels in the input and the reconstructed visualizations, defined as follows:

$$L_r = \frac{1}{WH}\sum_{h=1}^{H}\sum_{w=1}^{W} ||\boldsymbol{I}(h,w) - \hat{\boldsymbol{I}}(h,w))||, \tag{6}$$

where $\boldsymbol{I}(h,w)$ indicates the 3D RGB color value at position (h,w).

4.4 Training and Inferencing

4.4.1 Training

During training, we combine the four proposed loss functions into a single total loss using a linear weighted sum, allowing for straightforward gradient computation in the deep neural network:

$$L = \alpha L_r + \gamma L_d + \delta L_c + \eta L_o. \tag{7}$$

Here, α, γ, δ, and η represent balancing weights. Since all loss functions are normalized to the range [0,1], we set these weights to 1 by default. The reconstruction loss L_r and color order loss L_o not only help the network infer hidden relationships between the colormap and data during training, but also serve as self-supervised constraints to fine-tune the trained parameters during inferencing. We further analyze the roles of these loss functions through ablation studies in Section 5.5.

4.4.2 Inferencing

To enhance the generality of our method, we leverage the differentiable colormapping module and the color order loss to fine-tune the trained model in a self-supervised manner, without requiring paired original colormaps and data. Specifically, we first use the trained model to generate an initial colormap and data directly from the input visualization. In each subsequent iteration, we compute the reconstruction loss between the newly recovered colormap and the directly learned colormap (which replaces the original colormap in the loss function L_c), as well as the reconstruction loss between the newly recovered data and the directly learned data (which replaces the original data in the loss function L_d). Using the newly recovered colormap and data at each iteration, we generate a corresponding reconstructed visualization and compute the reconstruction loss relative to the input visualization. Additionally, the color order loss is enforced to prevent arbitrary color arrangements. Finally, the parameters of the trained model are fine-tuned by iteratively optimizing the recovered colormap and data to minimize the color order loss and reconstruction loss.

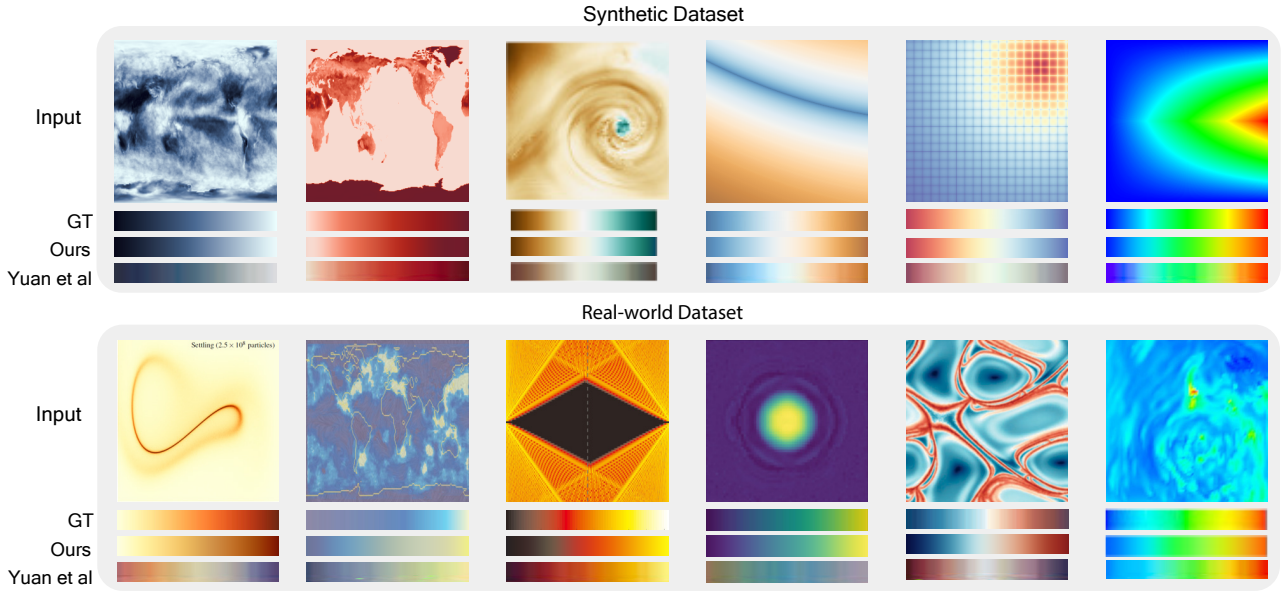


Fig. 5: **Comparison: our recovered colormaps and ones recovered by Yuan *et al.*'s method [63]**. Row 1: Colormaps recovered from synthetic data. Row 2: Colormaps recovered from a real-world data, with images screenshotted from [4, 20, 23, 30, 45, 50] in left-to-right order.

Table 1: **Quantitative comparison of recovered colormap quality** for our method and that of Yuan et al. [63], using standard MSE, PSNR, and SSIM metrics. Values reported represent averages over all tested colormaps, with the best-performing score highlighted in bold.

Dataset	Method	Recovered Colormap (*ignoring direction*)			Recovered Colormap (*considering direction*)		
		MSE↓	PSNR↑	SSIM↑	MSE↓	PSNR↑	SSIM↑
Synthetic Dataset	Ours (RGB)	**0.001**	**34.033**	**0.961**	**0.003**	**33.739**	**0.957**
	Ours (LAB)	0.015	26.456	0.907	0.026	25.527	0.887
	Yuan *et al.*	0.011	23.160	0.905	0.012	23.090	0.904
Real-world Dataset	Ours (RGB)	**0.038**	**17.205**	**0.801**	0.071	**14.910**	**0.751**
	Ours (LAB)	0.055	14.224	0.748	0.080	12.762	0.709
	Yuan *et al.*	0.047	15.034	0.759	**0.070**	13.308	0.722

5 Evaluation

We evaluate the effectiveness and utility of our method through the following experiments: (1) comparing to Yuan *et al.*'s method [63] on two datasets and three metrics, (2) measuring the perceived quality of our recovered colormaps, and (3) conducting an ablation study to analyze the importance of individual components of our network architecture. Detailed results can be found in our supplementary materials.

5.1 Implementation

We implemented our network in PyTorch and trained it on an NVIDIA 4090D GPU for 50,000 iterations (about 6.5 hours) with a batch size of 64 and a learning rate of 0.001. The Adam optimizer [26] with β_0 set to 0.5 and β_1 set to 0.999 was employed to optimize the network parameters during training. During testing, we performed fine-tuning with a learning rate of 0.0001 for each input visualization.

5.2 Datasets

5.2.1 Synthetic Dataset

We constructed a synthetic visualization dataset by mapping commonly used colormaps to existing open-sourced scalar data and various mathematical functions, using the criteria of covering a wide range of data distributions and color variations [37]. For the scalar data, we used two sources: (i) 179 randomly selected 2D scalar fields from the ECMWF dataset (`https://apps.ecmwf.int/datasets`), and (ii) 228 2D scalar fields generated using various mathematical functions (e.g., linear gradients or periodic functions) [37]. For the colormaps in the synthetic dataset, we collected a total of 179 continuous colormaps from two sources: the Python matplotlib library and academic research papers on color design [37, 46, 48, 63]. Our synthetic dataset comprises 72,853 visualization images ($179 \times 179 + 228 \times 179$), with their accompanying colormap and data. We randomly split the dataset into a *training set* (90%) and a *test set* (10%).

5.2.2 Real-world Dataset

We further compiled a dataset of 100 paired real-world visualizations and their corresponding colormaps, manually extracted from existing academic publications using the VIS30K dataset [9]. We refer to this as the real-world dataset. Specifically, we first identified 2D scalar field visualizations within the VIS30K dataset. For each visualization, we then reviewed the corresponding academic paper to check for the presence of an accompanying legend. If a legend was provided, we manually captured the corresponding colormap via screenshot to serve as the original ground-truth colormap for the visualization; otherwise, the visualization was discarded. Note that all real-world visualizations were used exclusively for evaluation purposes and not for training the network. The corresponding colormaps were used solely as ground-truth for conducting quantitative and qualitative comparisons.

5.3 Quantitative and Qualitative Comparison

We evaluate the performance of our method both qualitatively and quantitatively by comparing it to the state-of-the-art deep colormap extraction method proposed by Yuan *et al.* [63].

5.3.1 Approach

Yuan *et al.* [63] introduced a convolutional neural network for colormap extraction from single visualizations, and they have open-sourced their code at `https://github.com/yuanlinping/deep_colormap_extraction`. We consider this method the most comparable and reproducible baseline, as other inspiring prior work [43, 44] dif-

Table 2: **Quantitative comparison of different ablated networks** using MSE, PSNR, and SSIM metrics to assess recovered colormaps. Values reported represent averages over all tested colormaps, with the best-performing score highlighted in bold and the second-best in Italic.

Dataset	Method	Recovered Colormap (*ignoring direction*)			Recovered Colormap (*considering direction*)		
		MSE↓	PSNR↑	SSIM↑	MSE↓	PSNR↑	SSIM↑
Synthetic Dataset	Ours with Fine-Tuning	**0.001**	*34.033*	**0.961**	**0.003**	*33.739*	**0.957**
	Ours w/o Fine-Tuning	**0.001**	**34.054**	*0.958*	**0.003**	**33.756**	*0.954*
	w/o B-Spline	0.028	21.075	0.867	*0.036*	20.588	0.854
	w/o Recon. Loss	*0.025*	21.107	0.882	0.038	20.109	0.861
	w/o D&C Loss	0.060	13.618	0.782	0.109	11.196	0.718
	w/o Order Loss	0.028	20.432	0.874	0.041	19.502	0.852
Real-world Dataset	Ours with Fine-Tuning	**0.038**	**17.205**	**0.801**	**0.071**	**14.910**	**0.751**
	Ours w/o Fine-Tuning	*0.040*	*17.071*	*0.797*	*0.073*	*14.710*	*0.745*
	w/o B-Spline	0.054	14.476	0.755	0.078	13.179	0.720
	w/o Recon. Loss	0.061	13.667	0.751	0.076	12.815	0.725
	w/o D&C Loss	0.085	11.711	0.709	0.104	10.689	0.680
	w/o Order Loss	0.054	14.561	0.761	0.083	13.123	0.723

fers in focus (e.g., chart with legend) and lack compatible pipelines for direct comparison. To ensure a fair comparison, we trained Yuan *et al.*'s model using their publicly available code and our training data. Note that we directly compared the output of the models without further post-processing for both methods.

5.3.2 Metrics

To assess the performance of our method, we employed three metrics: *mean squared error* (MSE), *peak signal-to-noise ratio* (PSNR), and *structure similarity index measure* (SSIM). These metrics compare the recovered colormaps with the corresponding ground-truth.

- *MSE* measures the average squared difference between the corresponding pixels in the recovered colormap and the corresponding ground truth. Lower MSE indicates a better match between the recovered elements and ground truth.
- *PSNR* measures the ratio between the maximum signal (e.g., the maximum color value in the colormap) and the noise introduced during the recovery process, as quantified by the MSE. Higher PSNR indicates better reconstruction quality with lower noise.
- *SSIM* goes beyond pixel-wise differences by incorporating structural information into the evaluation. It considers luminance, contrast, and structural similarity between the recovered results and the ground truth. A value closer to one indicates a higher level of perceptual similarity.

5.3.3 Qualitative Comparison

Figure 5 compares visual results of our colormap recovery method to Yuan *et al.*'s. As demonstrated across a variety of examples, our method shows superior performance in recovering continuous colormaps. Moreover, the recovered colormaps closely resemble the ground truth, highlighting the accuracy and visual fidelity of our approach. Additional qualitative comparisons–including colormaps before fine-tuning, intermediate reconstructed data, and reconstructed visualizations–are available in the supplementary material.

5.3.4 Quantitative Comparison

Table 1 quantitatively compares our method to Yuan *et al.*'s approach [63] using the MSE, PSNR, and SSIM metrics. We tested both methods using 999 randomly selected visualizations from the test subset of the synthetic dataset, and all 100 visualizations from the real-world dataset. We evaluated the alternatives in two ways:

- *Recovered Colormap (ignoring direction)* measures the accuracy of the predicted colormap without considering the order of colors.
- *Recovered Colormap (considering direction)* measures the accuracy of the predicted colormap considering the order directions.

Our model outperforms Yuan *et al.*'s in recovering colormaps, whether the colormap direction is ignored or not. This superiority holds true for both synthetic and real-world datasets in all cases considered. However, when ordering direction is considered, the results for both models are worse than when the direction is ignored, indicating that both models may occasionally produce colormaps with reversed directionality. Our method requires five additional hours of training compared to Yuan *et al.*'s method, using the same machine and number of iterations. The colormap recovery process takes about 0.36 seconds, with an additional one minute for fine-tuning. While this is longer than Yuan *et al.*'s, it remains acceptable for interactive exploration. Detailed timing records can be found in the supplementary material.

5.4 Perceptual Study

5.4.1 Experiment Design

To assess the perceived authenticity of our recovered colormaps, we conducted a one-factor within-subject perceptual study. The single factor in the experiment was the colormap recovery method (either ours or Yuan *et al.*'s). Each participant evaluated colormaps from both methods shown with random orders. The goal was to determine how similar participants perceived the generated colormaps to the ground truth in terms of color distribution. We randomly selected 40 visualizations from the synthetic dataset and 20 from the real-world dataset. For each visualization, we recovered colormaps using both methods. For a complete list of the experimental stimuli, please refer to our supplementary material.

5.4.2 Task

Participants were presented with a set of 120 trials. Each trial displayed a ground-truth colormap on the left, and on the right a colormap reconstructed using our method or Yuan *et al.*'s method. Participants then completed a forced-choice questionnaire on a five-point Likert scale to rate the perceived color distribution similarity between the two colormaps. We asked the question "How similarly do the colors appear to be distributed in the image on the right compared to the one on the left?" with choices: (1) strongly dissimilar, (2) slightly dissimilar, (3) neutral, (4) slightly similar, and (5) strongly similar.

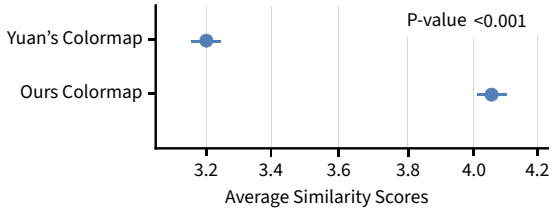
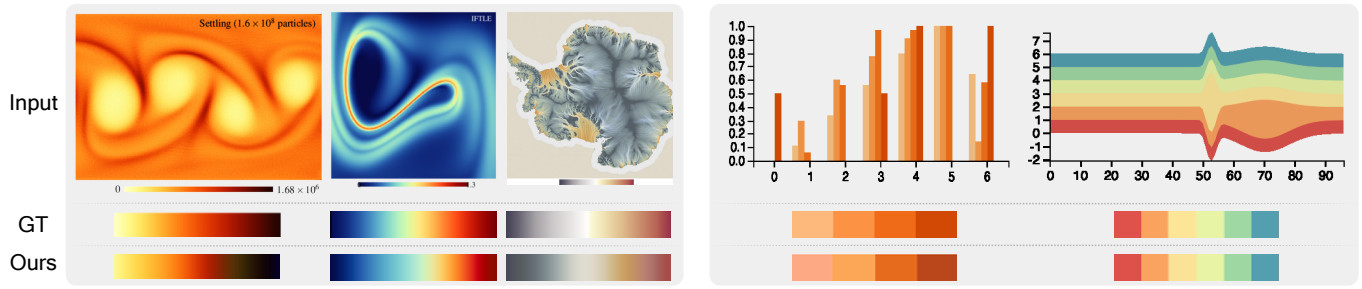


Fig. 6: **Perceptual study results**, showing average similarity scores with 95% CI and p-value between our method and the alternative.



(a) Visualizations with legend

(b) Visualizations with discrete color palettes

Fig. 7: **Adaptations of our method to general visualizations**. (a) Colormap recovery results from input visualizations with legends, taken from [2, 20]. (b) Visualizations from [63], showing the recovery of discrete color palettes.

5.4.3 Procedure

Our experiment consists of three steps: a training phase, the main experiment, and demographic information collection. The main experiment included 120 random trials for each participant, with 80 trials using colormaps from the synthetic dataset and the remaining 40 from the real-world dataset. Each trial presented two side-by-side colormaps: a ground truth colormap on the left, and on the right a colormap from the chosen method. Participants rated the similarity of each presented colormap (ours or Yuan *et al.*'s) to the ground truth. We recorded their total response time and selected similarity scores for further analysis. On average, participants took 9 minutes to complete the experiment (*minimum*: 3 minutes, *maximum*: 17 minutes).

5.4.4 Participants

We recruited 40 participants (30 males, 10 females) from the local university. To ensure data quality, participants were required to complete the experiment using the same computer screen (2048×1080 resolution) in a consistently illuminated room.

5.4.5 Results

We collected $4,800$ valid results. Average similarity scores given by participants are summarized in Fig. 6. The plot shows the raw average user similarity scores along with 95% confidence intervals (CI). Participants consistently determined our results to be more similar to the ground truth than Yuan *et al.*'s results. The differences between our results and Yuan *et al.*'s are statistically significant ($p < .001$), indicating a strong assessment of the superiority of our recovered colormaps.

5.5 Ablation Study

We conducted an ablation study to assess the importance of different components of our network by removing or replacing key components by alternative solutions. We assessed our network as follows:

- **Without fine-tuning during inference** (w/o Fine-Tuning): we directly use the trained model to generate results without fine-tuning its parameters.
- **Without using the cubic B-spline curve representation** (w/o B-Spline): we replaced the cubic B-spline curve representation with a traditional linear colormap representation [64].
- **Without using the reconstruction loss** (w/o Recon. Loss): we trained the network without calculating the reconstruction loss between the recovered visualization and the input.
- **Without data and color fidelity loss** (w/o D&C Loss): we trained our network without incorporating the data fidelity loss and color fidelity loss.
- **Without color order loss** (w/o Order Loss): we omitted color order loss in the training stage.

This ablation study dissects the significance of various network components on our model's ability to recover hidden data and accurate color information from individual visualizations. Table 2 lists results of the various metrics. Further quantitative results, including different balancing weights and control points, are available in the supplementary material. Analyzing the quantitative values and visual results, we draw the following four conclusions:

- The absence of the component responsible for calculating data and color fidelity loss (w/o D&C Loss) significantly impacts performance, suggesting their critical roles in recovering colormaps.
- Discarding B-spline representation (w/o B-Spline), reconstruction loss (w/o Recon. Loss) and color order loss (w/o Order Loss) results in a general decrease in performance.
- The real-world dataset benefitted more from the fine-tuning procedure. A possible reason is that synthetic datasets have simpler, more regular patterns, which the general model captures well. In contrast, the fine-tuned model, being more specialized, struggles with the simpler, different distribution of synthetic data. This suggests that fine-tuning improves performance on real-world visualizations but may introduce bias on simpler synthetic data.

6 Applications and Discussions

6.1 Application Prototype

Recovering hidden colormaps unlocks new possibilities for enhancing both machine and human understanding of static visualizations. This recovered information serves as a key for interactive exploration, empowering users to delve deeper into the data. We demonstrate the utility of our method with two prototype applications: **colormap adjustment** and **colormap transfer**. Fig. 8 shows the interface of our interactive tool; see our supplementary material, video, and accompanying tool for details on the interface and interactions.

Colormap Adjustment. Colormap adjustment involves either manually modifying the input colormap or applying a new colormap (e.g., a user-designed one) to the visualization. Fig. 1(c) re-renders the input visualization by (i) adjusting the colormap to allocate more colors to data ranges with higher density, and (ii) applying a carefully designed colormap to better reveal hidden spatial variations in the data.
Colormap Transfer. Colormap transfer applies the recovered colormap to new data, enabling users to leverage their preferred designs and to visually compare data. Fig. 1(d) demonstrates two examples: applying the extracted colormap to a computed tomography dataset and a synthetic dataset simulating latitude changes.

6.2 Adaptation to General Visualizations

In this section, we demonstrate two straightforward ways to adapt our method to a broader range of visualizations–specifically, visualizations that include a legend or use discrete color palettes.

6.2.1 Visualizations with legend

The simplest way to generalize our approach to visualizations with a legend is to treat them as a whole within our network, i.e., by inputting the visualization and its paired colormap as a single image. Fig. 7(a) presents three examples from existing work [2, 20], in which corresponding color legends are provided. The color legends generated

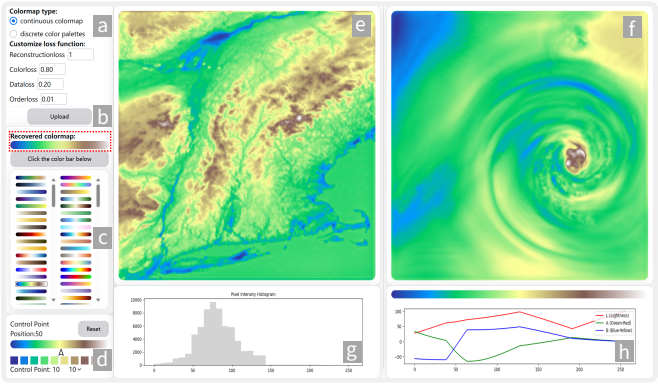


Fig. 8: **Interface of our interactive tool.** The tool consists of eight functionalities, (a) parameter control, (b) recovered colormap, (c) colormap selection, (d) colormap adjustment, (e) input visualization, (f) colormap transfer, (g) histogram distribution of the uploaded data by users, (h) color components of the recovered colormap.

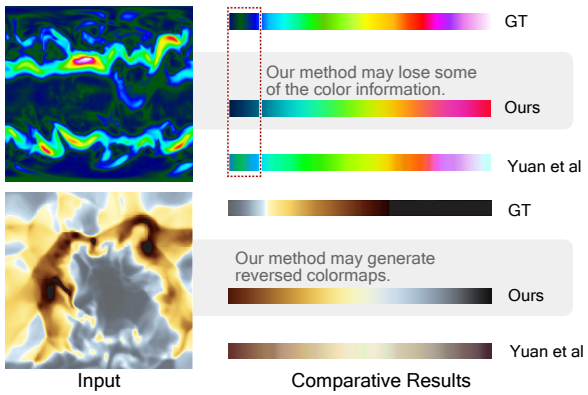


Fig. 9: **Limitations.** Above: for out-of-distribution input visualization, the colormap reconstructed by our method can at times be poor. Below: the direction of the colormap generated by our method is reversed.

by our method closely match the original ones in the visualizations, demonstrating the effectiveness of our approach.

6.2.2 Visualizations with discrete color palettes

While our approach focuses on continuous colormap recovery, it can also handle discrete color palettes by sampling colors from the recovered continuous colormap. Two important factors need to be emphasized in this process: the number of colors in the discrete color palette and their corresponding values. Our basic idea is that meaningful discrete colors can naturally emerge from the distribution of data values. Therefore, we leverage DBSCAN [15] to cluster the estimated data from our approach, using the number of clusters as the number of colors in the discrete color palette. We then determine each color in the palette by selecting a color from our recovered continuous colormaps that corresponds to the average data value within each cluster. Fig. 7(b) presents two examples of visualizations with discrete color palettes, where the original input visualizations are sourced from Yuan *et al.*'s dataset [63]. Although the results shown in Fig. 7(b) are promising, our method might struggle when dealing with discrete color palettes that violate the monotonic assumption underlying our color order loss.

6.3 Limitations

While our approach offers significant potential to unlock hidden information from visualizations without color scales, it exhibits several limitations. Our approach is built upon the fundamental assumption of linear mappings between colormaps and data; therefore, its applicability may be limited when dealing with non-linear mappings [13, 58]. Our network's performance is currently skewed towards colormaps encountered during training. Less common or user-defined colormaps may lead to less accurate recovery: see Fig. 9 (above). Visualizations can be inherently ambiguous, allowing for multiple interpretations. For instance, reversing a colormap and the corresponding data can produce an identical visual representation. In such cases, the network may struggle to differentiate between true data variations and colormap ordering, potentially leading to recovered elements that deviate from the intended meaning: see Fig. 9 (below). While the recovered colormap and data are sufficient for course-grained data analysis—such as pattern and distribution exploration—this intrinsic ambiguity makes it challenging to support accurate data analysis. Training and fine-tuning the network can be computationally expensive, requiring significant processing power and resources, which limits the accessibility for users with limited computational resources. Additionally, our network cannot distinguish legend and visualization content, which may introduce noise during the colormap recovery process.

6.4 Future Work

Colormap recovery is a highly underdetermined and challenging problem, and there remains a large body of future work in this direction. To improve the network's ability to handle non-linear mappings or user-defined colormaps, techniques like data augmentation and domain adaptation can be explored [29]. Visualizations with inherent ambiguity or multiple interpretations may remain fundamentally unresolved, even when incorporating a large amount of training data, as such data may confuse the network. Addressing these challenges requires incorporating stronger constraints, such as additional perceptual cues [7, 55], uncertainty quantification [25, 27], and interactive exploration [14, 28] into the learning framework. To improve computational efficiency, lightweight network architectures as well as meta-learning techniques, e.g., MAML [16], can be used to reduce the computational burden and inferencing-time optimization. To improve colormap and data recovery by focusing on relevant visual elements in visualizations with legends, automatic object detection methods [49] could be introduced to identify and separate legend regions. We also hope to explore applications beyond color design, including integrating the network with scientific visualization tools, data exploration platforms, accessibility solutions for visualizations lacking color legends, and question-answering for visualization interpretation. Furthermore, although colormap recovery is beneficial for generating accessible visualizations, it may also pose risks concerning data security, which intrigues us to develop visual encodings for security in the future.

7 Conclusions

We present a novel colormap recovery network that leverages a decoupling-and-reconstruction strategy to accurately predict embedded colormaps and data from visualizations without legends. Backed by a mathematical analysis and informed by design requirements, the network incorporates a cubic B-spline colormap representation and four key loss functions for high-fidelity recovery. Our reconstruction loss function enables self-supervised fine-tuning during inferencing. Extensive comparisons and an ablation study demonstrate the superiority of our approach over alternative methods. Furthermore, we show the utility of our method with a prototype application with *visual recoloring* and *data view* functionalities. These, combined with the core colormap recovery capabilities, unlock the potential of "in-the-wild" visualizations, not only facilitating color design but also potentially improving accessibility for existing visualizations.

Acknowledgments

This work is supported by the grants from the National Key R&D Program of China (No. 2021YFF0704300), the National Natural Science Foundation of China (No. 62372271, No. 62132017 and No. U2436209), the Natural Science Foundation of Shandong Province (No. ZQ2022JQ32), the Excellent Young Scientists Fund Program (Overseas) of Shandong Province (No. 2023HWYQ-034), the Taishan Scholars Program (No. tsqn202408291), the Beijing Natural Science Foundation (L247027), and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Ecmwf public datasets. https://apps.ecmwf.int/datasets/. 2

[2] G. Abram, F. Samsel, M. R. Petersen, X. Asay-Davis, D. Comeau, and S. F. Price. Antarctic water masses and ice shelves: Visualizing the physics. *IEEE Computer Graphics and Applications*, 41(1):35–41, 2021. doi: 10.1109/MCG.2020.3044228 8

[3] K. Angerbauer, N. Rodrigues, R. Cutura, S. Öney, N. Pathmanathan, C. Morariu, D. Weiskopf, and M. Sedlmair. Accessibility for color vision deficiencies: Challenges and findings of a large scale study on paper figures. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, article no. 134. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3491102.3502133 1

[4] S. Bergner, T. Moller, D. Weiskopf, and D. J. Muraki. A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1353–1360, 2006. doi: 10.1109/TVCG.2006.113 6

[5] J. Bernard, M. Steiger, S. Mittelstädt, S. Thum, D. Keim, and J. Kohlhammer. A survey and task-based quality assessment of static 2d colormaps. In *Visualization and Data Analysis*, vol. 9397, 2015. doi: 10.1117/12.2079841 2

[6] D. Borland and R. M. Taylor Ii. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2), 2007. doi: 10.1109/MCG.2007.323435 1

[7] R. Bujack, T. L. Turton, F. Samsel, C. Ware, D. H. Rogers, and J. Ahrens. The good, the bad, and the ugly: A theoretical framework for the assessment of continuous colormaps. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 2018. doi: 10.1109/TVCG.2017.2743978 2, 3, 4, 5, 9

[8] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein. Palette-based photo recoloring. *ACM Transactions on Graphics*, 34(4), 2015. doi: 10.1145/2766978 2, 3

[9] J. Chen, M. Ling, R. Li, P. Isenberg, T. Isenberg, M. Sedlmair, T. Möller, R. S. Laramee, H.-W. Shen, K. Wünsche, and Q. Wang. Vis30k: A collection of figures and tables from ieee visualization conference publications. *IEEE Transactions on Visualization and Computer Graphics*, 27(9), 2021. doi: 10.1109/TVCG.2021.3054916 1, 2, 6

[10] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. *Computer Graphics Forum*, 38(3), 2019. doi: 10.1111/cgf.13686 2

[11] F. Crameri, G. Shephard, and P. Heron. The misuse of colour in science communication. *Nature Communications*, 5444(11), 2020. doi: 10.1038/s41467-020-19160-7 1

[12] A. Dasgupta, J. Poco, B. Rogowitz, K. Han, E. Bertini, and C. T. Silva. The effect of color scales on climate scientists' objective and subjective performance in spatial data analysis tasks. *IEEE Transactions on Visualization and Computer Graphics*, 26(3), 2020. doi: 10.1109/TVCG.2018.2876539 1

[13] M. Eisemann, G. Albuquerque, and M. Magnor. Data Driven Color Mapping. In *Proceedings of EuroVA 2011: International Workshop on Visual Analytics*, 2011. doi: 10.2312/PE/EuroVAST/EuroVA11/005-008 2, 9

[14] N. Elmqvist, P. Dragicevic, and J. Fekete. Color lens: Adaptive color scale optimization for visual exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(6), 2011. doi: 10.1109/TVCG.2010.94 3, 9

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996. doi: doi/10.5555/3001460.3001507 9

[16] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *ICML'17*, p. 1126–1135, 2017. doi: doi/10.5555/3305381.3305498 9

[17] J. Fu, B. Zhu, W. Cui, S. Ge, Y. Wang, H. Zhang, H. Huang, Y. Tang, D. Zhang, and X. Ma. Chartem: Reviving chart images with data embedding. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 2021. doi: 10.1109/TVCG.2020.3030351 3

[18] C. Garth, F. Gerhardt, X. Tricoche, and H. Hans. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1464–1471, Nov. 2007. doi: 10.1109/TVCG.2007.70551 1

[19] C. C. Gramazio, D. H. Laidlaw, and K. B. Schloss. Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), 2017. doi: 10.1109/TVCG.2016.2598918 2

[20] T. Günther and H. Theisel. Backward finite-time lyapunov exponents in inertial flows. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):970–979, 2016. doi: 10.1109/TVCG.2016.2599016 6, 8

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90 5

[22] J. Heer and M. Stone. Color naming models for color selection, image editing and palette design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2207676.2208547 2

[23] H. Janicke, A. Wiebel, G. Scheuermann, and W. Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1384–1391, 2007. 6

[24] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo. Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3025453.3025957 2

[25] A. Kamal, P. Dhakal, A. Y. Javaid, V. K. Devabhaktuni, D. Kaur, J. Zaientz, and R. Marinier. Recent advances and challenges in uncertainty visualization: a survey. *Journal of Visualization*, 24(5):861–890, Oct. 2021. doi: 10.1007/s12650-021-00755-1 9

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.1155/2023/7037124 6

[27] L. Kong, H. Kamarthi, P. Chen, B. A. Prakash, and C. Zhang. Uncertainty quantification in deep learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, p. 5809–5810. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3580305.3599577 9

[28] G. Li, Y. Liu, G. Shan, S. Cheng, W. Cao, J. Wang, and K.-C. Wang. Paramsdrag: Interactive parameter space exploration via image-space dragging. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):624–634, Jan. 2025. doi: 10.1109/TVCG.2024.3456338 9

[29] J. Li, Z. Yu, Z. Du, L. Zhu, and H. T. Shen. A comprehensive survey on source-free domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. doi: 10.1109/TPAMI.2024.3370978 9

[30] S. Liu, D. Wang, D. Maljovec, R. Anirudh, J. J. Thiagarajan, S. A. Jacobs, B. C. Van Essen, D. Hysom, J.-S. Yeom, J. Gaffney, et al. Scalable topological data analysis and visualization for evaluating data-driven models in scientific applications. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):291–300, 2019. doi: 10.48550/arXiv:1907.08325 6

[31] K. Lu, M. Feng, X. Chen, M. Sedlmair, O. Deussen, D. Lischinski, Z. Cheng, and Y. Wang. Palettailor: Discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 2021. doi: 10.1109/TVCG.2020.3030406 2

[32] J. Luo, Z. Li, J. Wang, and C.-Y. Lin. Chartocr: Data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021. doi: 10.1109/WACV48630.2021.00196 2

[33] D. Masson, S. Malacria, D. Vogel, E. Lank, and G. Casiez. Chartdetective: Easy and accurate interactive data extraction from complex vector charts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, number 147 in CHI '23. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3544548.3581113 2

[34] V. A. Mateevitsi, M. E. Papka, and K. Reda. Science in a blink: Supporting ensemble perception in scalar fields. In *2024 IEEE Visualization and Visual Analytics (VIS)*, pp. 216–220, 2024. doi: 10.1109/VIS55277.2024.00051 1

[35] P. Mishra, S. Kumar, and M. K. Chaube. Evaginating scientific charts: Recovering direct and derived information encodings from chart images. *J. Vis.*, 25(2), apr 2022. doi: 10.1007/s12650-021-00800-z 2

[36] S. Mittelstädt, D. Jäckle, F. Stoffel, and D. A. Keim. ColorCAT: Guided Design of Colormaps for Combined Analysis Tasks. In E. Bertini, J. Kennedy, and E. Puppo, eds., *Eurographics Conference on Visualization (EuroVis) - Short Papers*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151135 2

[37] P. Nardini, M. Chen, R. Bujack, M. Böttinger, and G. Scheuermann. A testing environment for continuous colormaps. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/TVCG.2020.

3028955 2, 6

[38] P. Nardini, M. Chen, M. Böttinger, G. Scheuermann, and R. Bujack. Automatic improvement of continuous colormaps in euclidean colorspaces. *Computer Graphics Forum*, 40(3):361–373, 2021. doi: 10.1111/cgf.14313 2, 3, 5

[39] P. Nardini, M. Chen, F. Samsel, R. Bujack, M. Böttinger, and G. Scheuermann. The making of continuous colormaps. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3048–3063, 2021. doi: 10.1109/TVCG.2019.2961674 2, 3

[40] J. R. Nuñez, C. R. Anderton, and R. S. Renslow. Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PLoS One*, 13(7), 2018. doi: 10.1371/journal.pone.0199239 1, 2

[41] B. Pham. Spline-based color sequences for univariate, bivariate and trivariate mapping. In *Proceedings of the First IEEE Conference on Visualization*, 1990. doi: 10.1109/VISUAL.1990.146383 2, 4

[42] H. Q. Phan, H. Fu, and A. B. Chan. Color orchestra: Ordering color palettes for interpolation and prediction. *IEEE Transactions on Visualization and Computer Graphics*, 24(6), 2018. doi: 10.1109/TVCG.2017.2697948 2

[43] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Computer Graphics Forum*, 36(3), Jun 2017. doi: 10.1111/cgf.13193 2, 3, 6

[44] J. Poco, A. Mayhua, and J. Heer. Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 2018. doi: 10.1109/TVCG.2017.2744320 1, 2, 3, 6

[45] T. Rapp, C. Peters, and C. Dachsbacher. Void-and-cluster sampling of large scattered data and trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):780–789, 2019. doi: 10.1109/TVCG.2019.2934335 6

[46] K. Reda, P. Nalawade, and K. Ansah-Koi. Graphical perception of continuous quantitative maps: The effects of spatial frequency and colormap design. In *Proceedings of the 2018 ACM CHI Conference on Human Factors in Computing Systems*, 2018. doi: 10.1145/3173574.3173846 1, 2, 6

[47] K. Reda, A. A. Salvi, J. Gray, and M. E. Papka. Color nameability predicts inference accuracy in spatial visualizations. *Computer Graphics Forum*, 40(3), 2021. doi: 10.1111/cgf.14288 2

[48] K. Reda and D. A. Szafir. Rainbow revisited: Modeling effective colormap design for graphical inference. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/TVCG.2020.3030439 2, 6

[49] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. doi: 10.1109/TPAMI.2016.2577031 9

[50] B. E. Rogowitz and L. A. Treinish. Why should engineers and scientists be worried about color. *IBM Research*, 1996. 6

[51] B. E. Rogowitz, L. A. Treinish, and S. Bryson. How not to lie with visualization. *Computers in Physics*, 10(3), 1996. doi: 10.1063/1.4822401 1

[52] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015. doi: 10.48550/arXiv.1505.04597 4, 5

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, (323):533–536, 1986. doi: 10.1038/323533a0 5

[54] M. Savva, N. Kong, A. Chhajta, F.-F. Li, M. Agrawala, and J. Heer. Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11. Association for Computing Machinery, New York, NY, USA, 2011. doi: 10.1145/2047196.2047247 2, 3

[55] K. B. Schloss, C. C. Gramazio, A. T. Silverman, M. L. Parker, and A. S. Wang. Mapping color to meaning in colormap data visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1), 2019. doi: 10.1109/TVCG.2018.2865147 9

[56] S. Silva, B. S. Santos, and J. Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2), 2011. doi: 10.1016/j.cag.2010.11.015 2

[57] S. Smart, K. Wu, and D. A. Szafir. Color crafting: Automating the construction of designer quality color ramps. *IEEE Transactions on Visualization and Computer Graphics*, 26(1), 2020. doi: 10.1109/TVCG.2019.2934284 2

[58] D. Thompson, J. Bennett, C. Seshadhri, and A. Pinar. A provably-robust sampling method for generating colormaps of large data. In *IEEE Symposium on Large-Scale Data Analysis and Visualization*, 2013. doi: 10.1109/LDAV.2013.6675161 2, 9

[59] C. Tominski, G. Fuchs, and H. Schumann. Task-driven color coding. In *Proceedings of 12th International Conference Information Visualisation*, 2008. doi: 10.1109/IV.2008.24 1, 2

[60] C. Tominski and H. Schumann. *Interactive Visual Data Analysis*. CRC Press, 2020. doi: 10.1201/9781315152707 2, 3

[61] J. Woodring, M. Petersen, A. Schmeisser, J. Patchett, J. Ahrens, and H. Hagen. In situ eddy analysis in a high-resolution ocean climate model. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):857–866, Jan. 2016. doi: 10.1109/TVCG.2015.2467411 1

[62] M.-J. Yoo, I.-K. Lee, and S. Lee. Color sequence preserving decolorization. *Computer Graphics Forum*, 34(2), 2015. doi: 10.1111/cgf.12567 2

[63] L.-P. Yuan, W. Zeng, S. Fu, Z. Zeng, H. Li, C.-W. Fu, and H. Qu. Deep colormap extraction from visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 28(12), 2022. doi: 10.1109/TVCG.2021.3070876 2, 3, 6, 7, 8, 9

[64] Q. Zeng, Y. Zhao, Y. Wang, J. Zhang, Y. Cao, C. Tu, I. Viola, and Y. Wang. Data-driven colormap adjustment for exploring spatial variations in scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 28(12), 2022. doi: 10.1109/TVCG.2021.3109014 1, 2, 3, 4, 8

[65] P. Zhang, C. Li, and C. Wang. Viscode: Embedding information in visualization images using encoder-decoder network. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 2021. doi: 10.1109/TVCG.2020.3030343 3

[66] L. Zhou and C. D. Hansen. A survey of colormaps in visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(8), 2016. doi: 10.1109/TVCG.2015.2489649 2

[67] L. Zhou, R. Netzel, D. Weiskopf, and C. R. Johnson. Spectral visualization sharpening. In *ACM Symposium on Applied Perception*, article no. 18, 2019. doi: 10.1145/3343036.3343133 3

[68] L. Zhou, M. Rivinius, C. R. Johnson, and D. Weiskopf. Photographic high-dynamic-range scalar visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(6), 2020. doi: 10.1109/TVCG.2020.2970522 3

[69] L. Zhou, D. Weiskopf, and C. R. Johnson. Perceptually guided contrast enhancement based on viewing distance. *Journal of Computer Languages*, 55, 2019. doi: 10.1016/j.cola.2019.100911 3