Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2016, Article ID 2635124, 17 pages
http://dx.doi.org/10.1155/2016/2635124

Research Article

A Novel Framework for Interactive Visualization and Analysis of Hyperspectral Image Data

Johannes Jordan, Elli Angelopoulou, and Andreas Maier

Pattern Recognition Lab, University of Erlangen-Nuremberg, Erlangen, Germany

Correspondence should be addressed to Johannes Jordan; johannes.jordan@cs.fau.de

Received 24 May 2016; Revised 23 August 2016; Accepted 8 September 2016

Academic Editor: Sos Agaian

Multispectral and hyperspectral images are well established in various fields of application like remote sensing, astronomy, and microscopic spectroscopy. In recent years, the availability of new sensor designs, more powerful processors, and high-capacity storage further opened this imaging modality to a wider array of applications like medical diagnosis, agriculture, and cultural heritage. This necessitates new tools that allow general analysis of the image data and are intuitive to users who are new to hyperspectral imaging. We introduce a novel framework that bundles new interactive visualization techniques with powerful algorithms and is accessible through an efficient and intuitive graphical user interface. We visualize the spectral distribution of an image via parallel coordinates with a strong link to traditional visualization techniques, enabling new paradigms in hyperspectral image analysis that focus on interactive raw data exploration. We combine novel methods for supervised segmentation, global clustering, and nonlinear false-color coding to assist in the visual inspection. Our framework coined Gerbil is open source and highly modular, building on established methods and being easily extensible for application-specific needs. It satisfies the need for a general, consistent software framework that tightly integrates analysis algorithms with an intuitive, modern interface to the raw image data and algorithmic results. Gerbil finds its worldwide use in academia and industry alike with several thousand downloads originating from 45 countries.

1. Introduction

Multispectral imaging allows capturing of rich reflectance information that is not available with traditional RGB cameras. It has played a key role in the field of remote sensing for decades, ever since aircrafts and satellites became equipped with hyperspectral sensors systems. One such notable example is the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) [1]. Hyperspectral sensors are less widely used "on the ground," though they are becoming more popular in areas like astronomy, cultural heritage, agriculture, and medical imaging.

In a multispectral image, each pixel is a vector of intensity values, where each value corresponds to the scene radiance over a small range of wavelengths. The resulting vectors typically are of lengths ranging between 31 and 200. The high-dimensional nature of the data and its strong interband correlations pose challenges to computer vision algorithms. Adaptations are needed to expose and exploit the information contained in the data. A similar problem arises when one tries to manually process such high-dimensional data. There is a lack of intuition which demands a presentation that guides the user during data exploration. Effectively, one needs two components: simultaneous visualization of different aspects of the information and strong user interaction.

As the popularity of multispectral sensors has been increasing, so has the need for computer-aided, interactive analysis. For example, one of the first popular analysis frameworks for providing a graphical interface for inspection was the Spectral Image Processing System which was introduced in 1992 [2]. However, almost all available multispectral analysis software is tailored to specific applications. As expected, most of the available software focuses on remote sensing and is designed to foster a workflow specific to this domain.

In recent years, the availability of cheaper, easier-to-use multispectral cameras further opened this modality to

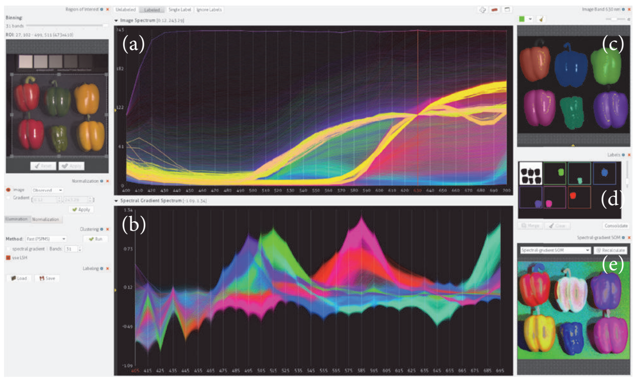


Figure 1: Gerbil user interface. (a) Spectral distribution view, (b) spectral gradient distribution view, (c) spatial view of a single image band with segment overlays, (d) label manager, and (e) false-color display.

new applications. Various devices based on electronically tunable filters [3] or imaging spectrometers [4] are commercially available from several vendors. An example of sensor targeting emerging applications is MuSIS [5], which is currently in use in several museums and libraries. However, the users of these sensors lack a software solution that enables them to explore the data in an interactive and intuitive fashion.

To address this limitation we developed the Gerbil software platform which incorporates established interactive visualization concepts to provide a new presentation of spectral images as well as a new workflow for inspecting them. Figure 1 depicts a screenshot of Gerbil in use for inspecting a multispectral image. Gerbil is developed by an open-source software project in the tradition of free software frameworks in the signal processing research community such as OpenCV [8] or Weka [9]. In this paper we provide a scientific reference for Gerbil and describe the core methodologies it encompasses. We also provide a detailed presentation of the feasibility and performance of the algorithms in Gerbil and show how our novel approach is a significant improvement compared to established practice.

The paper bundles and expands on the following contributions:

(i) Interactive visualization of spectral distributions based on efficient parallel coordinates [6]

(ii) Supervised segmentation of hyperspectral data [10]

(iii) Fast global clustering with superpixel support [11]

(iv) Fast nonlinear false-color visualization [12]

These methods were derived by adapting established algorithms for hyperspectral data and interactive time constraints. They allow us to introduce new paradigms in hyperspectral image analysis that focus on interactive raw data exploration, built on the tight incorporation of the aforementioned techniques into a comprehensive open-source software framework. The software is publicly available at http://gerbilvis.org/ under a free software license.

2. Previous Work

One of the first widely recognized software packages for multispectral analysis, LARSYS, became available in the 1960s. It was operated via a text terminal. Several frameworks for graphical, interactive multispectral, or hyperspectral data analysis that are still of broad influence today date back to the early 1990s. Earlier frameworks focus on a specific application, most prominently in the field of remote sensing. Boardman et al. [13] provided an overview of the history of established software systems for remote sensing hyperspectral data. Biehl [14] created an updated list on the web in 2007. In this section we give first a brief overview of domain-specific software packages before reviewing more widely applicable software. We focus on the interactive inspection of images.

2.1. Remote Sensing. The Spectral Image Processing System (SIPS) was presented by Kruse et al. [2] in 1993. It introduced the Spectral Angle Mapper (SAM), a spectrum-comparison tool still popular in the field of spectral matching [15]. Data was presented either as single bands or in a false-colored composite of three user-selectable bands. The user could also view a selected pixel's spectrum. Color-coded stacked spectra were provided for a selected slice (a vertical or horizontal line scan or an arbitrary path in the image). This set of visualization forms are still typical of popular software for hyperspectral capture or analysis.

The MultiSpec freeware package by Biehl and Landgrebe [16] can analyze multispectral images from various sources, using the powerful GDAL library [17] for data I/O. The focus of this software is classification. The target audience is the general Earth science community. Besides providing all common visualizations already mentioned, it can generate biplots that relate selected regions in a pair of bands and a statistics image display, which depicts the correlation between these bands.

A widely used commercial software is ENVI [18], initially developed by Boardman et al. [13]. Several innovations in hyperspectral analysis were introduced in ENVI, including, for example, the Pixel Purity Index (PPI) [19]. PPI finds the most spectrally pure pixels in an image. These pixels typically correspond to endmembers (constituent spectra of an image). A notable innovation in visualization is the *n-Dimensional Visualizer*, which uses PPI as input. It is an interactive data visualization method that allows real-time rotation of scatterplots in n-dimensions. The presentation of n-dimensional scatterplots in 2D can be somewhat unintuitive. Yet the tool is valuable to experts for identification of endmembers based on the depicted point clouds.

Two other notable pieces of software for hyperspectral remote sensing data analysis are HyperCube [23] and Opticks [24]. HyperCube is released by the US Army Corps of Engineers and contains functions to filter, warp (register two images), calibrate and undistort, photometrically project, and arithmetically manipulate the data. Opticks also includes many common tools like GIS annotations, false coloring, and histograms. Additionally, third parties can provide functionality through an external module, adding further capabilities to the software. For example, the Spectral Processing Extension [25] includes typical tools for hyperspectral data analysis.

Within the remote sensing context, many works exist on visualizing a multispectral image by employing dimensionality reduction. The goal is to present the image in false-color RGB by reducing dimensionality from spectral vectors of length D to length 3, which are then used as `r`, `g`, and `b` values. Based on the resulting pixel colors, the user should be able to discriminate regions of the image according to his specific interests. Methods to achieve this include variants of the principal component analysis (PCA), independent component analysis (ICA), and nonlinear methods. Some variants also incorporate classification results. A recent approach by Cui et al. [26] focuses on the interactive adaptation of such visualization. However, it still uses the common pixel map representation of the image.

2.2. Astronomy. A field with a long history of hyperspectral imaging is astronomy. The most notable software framework in this area is ds9 [27], available from the Smithsonian Astrophysical Observatory. This software is very powerful in the spatial representation of astronomical imagery. However, it provides a limited 3D visualization of the data cube that is rarely viable for nonastronomical data.

Recently, Li et al. [28] explicitly tackled the question of how to present multiband data. They draw image bands in 3D either as an image-stack or as a volume-rendered model, for example, horseshoe. Their volume rendering handles the obvious problem of clutter by applying transparency to individual data points based on their intensity or on a user-adjusted mask. However, one cannot generally assume that large image regions can be faded out to still unobstructedly display more relevant data.

2.3. Other Application Domains. Multispectral imaging has become increasingly popular in the preservation and analysis of artwork as well as historical documents. Colantoni et al. [29] analyzed multispectral images of paintings from the perspective of human color perception. From the image data, a representation in the CIE XYZ color space [30] is computed under controlled virtual illumination. Several tools can then be applied for visualization of trichromatic data. The original spectra are not considered in the analysis.

In 2010, Kim et al. [31] presented a solution for interactive visualization of historical documents. They provided a nicely designed self-contained analysis tool and incorporated innovations in how the data is presented. Some are specific to document analysis, for example, dealing with aging-related artifacts, while others are more general. The *spectral lens* feature, for example, presents data from two spectral bands in a single display. Similarity maps can be computed based on the L_2 norm between the mean spectral value of a selected region and all image pixels. However, similarity measures like SAM [15] would be better suited for spectra comparison. A 3D histogram plot is used to compare two spectra.

The National ICT Australia (NICTA) offers the Scyven software free of charge which features the reflectance recovery and material classification pipeline developed at NICTA [7, 32]. Its visualization includes false coloring and an adaptation of the parallel coordinates visualization introduced in Gerbil (see Section 3.2).

Labitzke et al. [33] introduced an interactive framework for linear spectral unmixing of multispectral datasets. Spectral unmixing is especially popular in remote sensing, where the spatial resolution is often low. Many algorithms exist for finding endmembers and performing spectral unmixing [34]. Labitzke et al. introduced an incremental method that can semiautomatically find endmembers. Then, visual feedback is provided by their complementary visualization that reflects the quality of the characterizing set. This set can be interactively modified in order to improve the unmixing. The authors explicitly differentiated their approach from Gerbil [33]. Spectral unmixing is integral to their interactive visualization approach, while in our methods we visualize the raw data. The algorithm and workflow proposed by Labitzke et al. are not in opposition to our approach, but instead they nicely complement the visualization capabilities of Gerbil.

3. Visual Inspection

In general, existing approaches share the same basic set of data representations and user interactions, while individual extensions typically follow a specific application scenario. However, the wider range of applications of multispectral imaging necessitates an interactive visualization framework that is both sufficiently general for a broader range of applications and more versatile than existing basic representations. In our framework, Gerbil, we follow a novel concept that enables an entirely new workflow in exploring a multispectral image. It revolves around presentation and exploration that make the image data more apparent to the user, effectively allowing a more direct interpretation of the data. The user does not rely on, but is merely guided by, automatic processing.

3.1. Main Concept. Figure 2 gives an overview of the key features that constitute Gerbil. In order to have a framework that is suitable for use in a large variety of applications, we incorporate several well-understood hyperspectral analysis tools into Gerbil, while adding concepts from the data visualization and computer vision communities. The value of our approach lies in employing these features in a fashion that enriches the user experience while investigating a multispectral image. One such example is the incorporation of parallel coordinates, a technique for display of high-dimensional data. Parallel coordinates are used in visualizing spectral distributions. The visualization is highly dynamic and interacts with the topological representations of the image (e.g., a grayscale depiction of a single image band). Another contributing factor is the emphasis on the *spectral gradient* descriptor [35]. Its distribution is displayed next to the spectral distribution. The spectral gradient is a multispectral descriptor that focuses on data aspects, which are directly related to material and reflectance properties.

Figure 1 depicts the three main components of the graphical user interface: the visualization of original spectra (Figure 1(a)) and the spectral gradient spectra (Figure 1(b)) both via parallel coordinates and the spatial image view

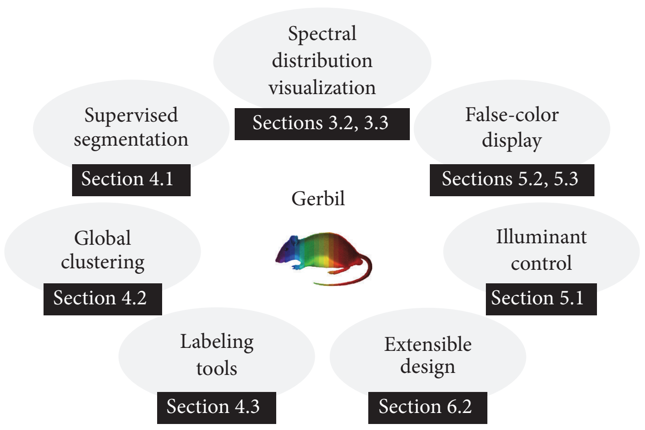


FIGURE 2: Overview of the Gerbil feature set and corresponding article sections.

(Figure 1(c)). The interaction between these elements, as is explained in detail in Section 3.3, is augmented by tools that give further guidance to the user. One such example is dimensionality reduction via PCA or nonlinear methods, as discussed in Section 5. Another tool is feature-space clustering of data, described in Section 4.2. Especially important in our workflow is supervised segmentation, where the user provides input depending on the goal of their investigation or on prior knowledge. In Section 4.1, we describe how we employ a graph-based segmentation algorithm for this task.

3.2. Spectral Distribution View. Existing approaches on depicting a multispectral image in its entirety are limited by the spatial layout of the image. The image data is viewed as a cube, with the z-axis corresponding to spectral bands, which are stacked on top of each other. Use of modern volume rendering techniques can make this representation useful in some scenarios [28]. However in most cases, where there is no sparsity in pixels of interest, a very cluttered view is obtained that reveals the shortcomings of a simple 3D arrangement. Other methods, such as scatter plots or false coloring, rely on dimensionality reduction. While the visualization of reduced data is helpful in many applications, it is hard to preserve subtle details. In contrast, a good visualization of the original data helps the observer evaluate how well a dimensionality reduction method fits a specific application.

One way of addressing this issue is to defer from the topological relations in the image, for the time-being, and concentrate instead on a graspable representation of the spectral distribution. To do this efficiently, we employ the parallel coordinates [36] method as explained below. This is a well established technique for visualizing high-dimensional geometry and analyzing multivariate data. It has been widely used, for example, in financial applications and geographic information systems. One can see the traditional spectral visualization as a specific instantiation of parallel coordinates visualization. By building on the more general concept, we can incorporate tools from the visualization community for high-dimensional data presentation.

According to the parallel coordinates concept, a D-dimensional feature space (resulting from D spectral bands) is projected onto a two-dimensional view as follows. D parallel vertical lines denote the D axes, that is, the D spectral bands. The y-coordinate on the ith axis corresponds to a spectrum's value at band i. To display the spectral vector of a pixel, a polyline is drawn with its vertices lying on the corresponding vertical axes. The resulting display follows the layout of a plot where the x-axis would denote wavelength and the y-axis denotes intensity.

3.2.1. Optimized Parallel Coordinates. Drawing a polyline for each single multispectral vector has several drawbacks: it is time consuming; and the display may easily get cluttered in which case single polylines may not separate well from the rest of the data. A solution to both clutter and speed concerns is to draw fewer polylines, where a polyline can represent a set of pixels. With this distinction, polylines that represent more pixels appear stronger.

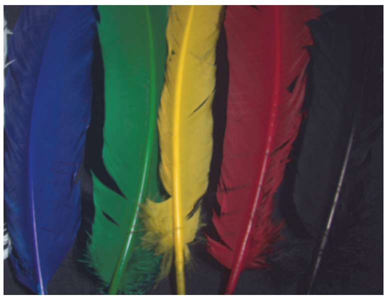
We realize this representation by introducing a histogram in the feature space with B evenly distributed bins in each dimension. B is user adjustable between 2 and the dynamic range of the captured data (typically 2^8 to 2^{16}). For example, the histogram for a 31-band multispectral image with $B = 256$ would hold $B^D = (2^8)^{31} = 2^{248}$ bins. Building a dense histogram of this size is not feasible; however a sparse histogram can be created by using an ordinary hashing algorithm. The key idea here is that the amount of occupied bins is bound by the spatial resolution of the input image. For example, a spatial resolution of 512×512 leaves only 2^{18} possible distinct values, effectively giving an upper bound to the amount of bins filled in a sparse histogram, or the amount of hash keys needed.

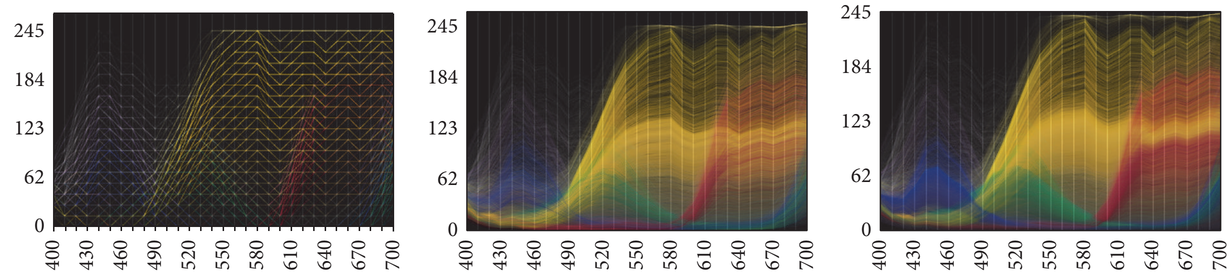
For each populated bin, a polyline is drawn in the parallel coordinates visualization. The strength of the polyline is manipulated by assigning it an opacity α. α is determined by the relationship between the number of pixels represented by the bin, n_{bin}, and the total number of processed pixels, n_{total}.

$$\alpha = c_\alpha \frac{f(n_{\text{bin}} + 1)}{f(n_{\text{total}})}, \tag{1}$$

where c_α is a user-adjustable factor and $f(\cdot)$ is a logarithmic function. The logarithm emphasises bins with fewer pixels. The idea is that even a single pixel should be perceptible. The logarithm also ensures that the resulting dynamic range of alpha values can be represented reasonably well with an 8-bit alpha channel. Nowadays, Gerbil overcomes this limitation by drawing onto a floating-point framebuffer and the user can choose $f(\cdot)$ to be linear.

3.2.2. Drawing Refinements. The newer design of Gerbil includes several measures to further enhance the visual quality and accuracy of the spectral distribution display. By dividing the feature space into a fixed number of equally spaced bins, the histogram applies a nonadaptive quantization of a spectral vector $\mathbf{x}$. A possible strategy to reduce the introduced quantization error is to employ a binning that is adapted to the observed data. A straight-forward method is to


(a) True-color display



(b) Previous work [6], $B = 20$ (c) Proposed method, $B = 20$ (d) Proposed method, $B = 256$

FIGURE 3: Spectral distribution view of cropped *feathers* image with different bin parameters. Spectra in true color.

perform a separate histogram equalization in each dimension i, which enforces a uniform PDF of the mapped vector values x_i [37]. While it works well for big clusters of similar pixels, spectra that are sparsely represented in the image will suffer from such an operation. It may increase the average accuracy at a great expense in the precision of single pixel representations, which is not desirable.

We employ an alternative strategy by adjusting how a bin is drawn. Our method achieves an improved general accuracy without a significant expense in the accuracy for any single pixel. When drawing a bin, we draw the mean vector of its contents instead of its midrange values. This can be computed on the fly while adding new vectors, with a final normalization based on the already stored number of entries. Figure 3 shows an example of the visual quality gain. A broad description of the spectral distribution is already possible with a low B.

Another hindrance in visual quality is the mutual obstruction of pixel representations. In many use-cases, pixels are color coded (see Section 3.3). This involves effectively drawing several distributions on top of each other. In highly populated intensity ranges this can lead to extensive occlusions. There exists work on clutter reduction in parallel coordinate plots that tackles this problem [38]. The key idea is to only render a randomly sampled subset of the data points. Instead, we significantly reduce clutter by drawing data in a randomly shuffled order, without dismissing any information.

3.2.3. Evaluation. The error introduced by the histogram-based quantization can be measured by the average root mean squared error (RMSE) as well as the maximum absolute deviation (MAD), between original vectors $\mathbf{x}_i$ and their quantized counterparts $\hat{\mathbf{x}}_i$. This gives us the measure

$$\text{RMSE} = \sum_{i=1}^{N} \frac{\sqrt{\sum_{j=1}^{D} \left(\hat{x}_{ij} - x_{ij}\right)^2 / D}}{N}, \tag{2}$$

for the average RMSE, and

$$\text{MAD} = \max \left(\max \left(\left| \hat{x}_{ij} - x_{ij} \right|, \ j \le D \right), \ i \le N \right), \tag{3}$$

where $\max(\cdot)$ is the sample maximum.

The number of bins per dimension B is a crucial parameter. It lets the user choose between drawing speed, viewing quality, and accuracy. Even a considerably low B should provide acceptable accuracy, and the speed-up by lowering B should be effective. We evaluate RMSE and MAD for both the previously published [6] and the newly proposed drawing methods on several datasets from different application domains. Table 1 provides statistics on the datasets. The CAVE dataset consists of objects in a laboratory settings, while Foster captured natural (outdoor) scenes. Indian Pines and D.C. Mall are widely used remote sensing images.

We use a desktop machine equipped with a quad-core Intel Core i7 CPU running at 2.80 GHz and a GeForce GTX 550 Ti consumer graphics card for testing the computational performance. We measure the time needed for drawing operations via `GL_TIMESTAMP` and draw in WUXGA resolution.

In Figure 4 we plot running times against accuracy for varying B. *Bin Center* describes the originally published drawing method, while *Mean* denotes our refined drawing method. We can observe that the average RMSE becomes

Table 1: Datasets used for evaluation.

Name	Size	Resolution	Bands	λ (nm)	Source
CAVE	32	512×512	31	400–700	[20]
Foster	8	1340×1020	33	400–720	[21]
Indian Pines	1	145×145	220	400–2500	[22]
D.C. Mall	1	307×1280	191	400–2475	[22]

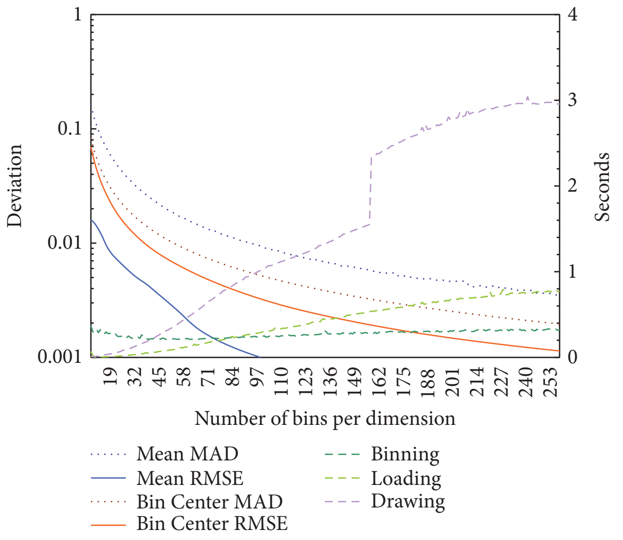


Figure 4: Experimental results on a scene from the Foster dataset. Quantization errors are plotted for both the old (Bin Center) and the refined (Mean) drawing method.

negligible with higher B for both methods. However, our method achieves low RMSE values for considerably lower settings of B. Due to outliers present in some of the histogram bins, the MAD for the refined method is somewhat higher than the original MAD.

The time needed to build the histogram is denoted as *binning* and is not determined by B. The time needed for preparing the geometry and loading it on the GPU (*Loading*) slowly grows with B. In contrast, B plays an important role for the time needed for the drawing operation (*Drawing*). For higher B values, the time needed for drawing grows to multiples of the time needed for preparation. Hence the histogramming plays an important role in achieving interactivity.

In Table 2 the running times and accuracy measures for the refined drawing method are listed on all datasets except Indian Pines, where, due to the very low resolution of the image, running times are negligible. The times shown for preparation are the combined histogram building and geometry loading times. As in Figure 4, it is observed that a low B can already achieve a small quantization error. Setting $B = 64$ is a reasonable compromise between speed and accuracy on the tested datasets. It provides an effective speed-up in comparison to a high histogram resolution without a perceivable loss in drawing accuracy.

It can be seen in Table 2 that on our test machine, even with a moderate setting, drawing the spectra from a large

Table 2: Average drawing times and accuracy.

Dataset	# bins	Prep. (s)	Draw. (s)	RMSE	MAD
CAVE	32^{31}	0.089	0.161	0.002	0.027
	64^{31}	0.115	0.240	0.001	0.014
	256^{31}	0.177	0.390	0.000	0.003
Foster	32^{33}	0.274	0.146	0.005	0.031
	64^{33}	0.358	0.554	0.002	0.014
	256^{33}	1.085	2.955	0.000	0.004
D.C. Mall	32^{191}	0.585	1.789	0.001	0.026
	64^{191}	0.881	3.171	0.000	0.013
	256^{191}	1.020	3.554	0.000	0.003

image may still take several seconds. Typically it is impractical to work on a high-resolution image without a region of interest. Yet, we alleviate longer drawing times by incrementally drawing the data (disabled in the benchmarking) in order to provide direct visual feedback in the form of a rough approximation of the full data (as pixels are drawn in a random order).

3.3. Interactive Inspection. An important aspect of today's visualization approaches for multivariate data is interactive manipulation of the presentation. A single view most often cannot provide the full understanding that may be gained by a series of user-controlled depictions. User input is vital to parallel coordinates in particular. We provide several mechanisms for both transient (cursor highlights) and persistent (color labels) interactive viewing.

In the parallel coordinate plots, the user can dynamically highlight specific data points, that is, spectral vectors that represent pixels. These are displayed in yellow and with full opacity as an overlay over nonhighlighted spectra. We realize this in OpenGL with layered frame buffers. Updates to the highlight only need to redraw highlighted spectra. While the highlight constantly follows the keyboard and/or mouse input; the corresponding pixels are instantly highlighted in the spatial view. While scrolling through the spectra, the topological view always reveals which pixels contribute to the highlighted spectra.

Two modes of operation exist for highlighting in the spectral distribution: single-band limited and multiband limited. The single-band limited highlight is formed by all spectral vectors falling into bins that share a specific intensity range in one band (see Figure 5(a)). The coarseness of this selection is therefore directly related to the binning parameter. The user selects the band and intensity range via mouse click or cursor keys. The multiband limited highlight provides a higher level of control in exchange for more detailed user input. Here, in each band, separate lower and upper intensity bounds can be set (see Figure 5(b)).

Another method of highlighting exists in the topological view. Here, the user can direct a cursor over individual pixels in the spatial view. In the corresponding viewports, the spectrum, and spectral gradient data of the pixel under the cursor are presented as a yellow overlay.

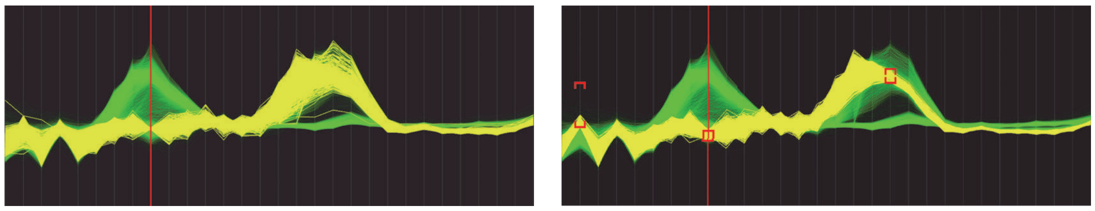

(a) Single-band limited (b) Multiband limited

Figure 5: Highlighting modes used in the spectral gradient distribution view of a scene with two objects. In (a), one objects is highlighted. In (b), further discrimination is performed by adding intensity bounds in two further bands.

While highlights give instant feedback, they constantly change as the user investigates the data. It is often desired to keep a selection of pixels distinguished from the rest of the data, for example, for comparison purposes. We call this pixel set a label; each pixel can be part of at most one label. A label can be seen as a permanent highlight. For each label, a distinct sparse histogram is created as described in Section 3.2.1. It is then drawn in the label color. When a histogram bin is part of the current transient highlight, the color is significantly shifted towards yellow.

When the user has selected pixels within the transient highlight, they may add this set of pixels to a label or delete their label association. By doing so, they can iteratively refine the labeling of the data to concentrate on specific details. Another way to alter the labeling is to use a "label" brush in the topological view or to use automated segmentation methods as discussed in Section 4. Labelings can also be stored for later use.

Labels are important because they serve as a memory in the connection between different representations of the multispectral image such as between a single band and the spectral distribution. A selection, or temporary highlight in one representation, is instantly propagated to the others. By labeling this highlight, it becomes permanent. The user can then continue his investigation within another representation, for example, the spectral gradient view, that may reveal new insights within this labeling. For example, a user may start by hand-labeling parts of a scene in the topological view. Then, they may restrict the spectral distribution view to this label. The parallel coordinates visualization could reveal a certain variance within the labeled pixels. By using selection tools inside this viewport, the user could separate parts of the pixels and assign them to a second label. The topological view instantly reveals which parts of the object contribute to which portion of the distinguished spectra. Next, the user may initialize a multiband limited highlight based on this second label. This reveals other regions in the scene that share spectral characteristics with the labeled pixels.

As a result, we facilitate a workflow of inspecting an image that is not possible with a traditional hyperspectral analysis framework. It is based on concurrent, concerted work with both topological and spectral views and allows a smooth and instantaneous switch in attention between them. Such a step-by-step exploration enables the user to quickly discover and grasp underlying information. In the visualization domain this procedure is considered a valuable tool for understanding complex data [39]. A narrated video demonstration of this workflow can be accessed at http://video.gerbilvis.org/.

4. Segmentation and Labeling

An interactive interface to the multispectral data is no replacement for automatic processing. In fact, the two approaches together form a powerful combination. Within our framework, it is easy to interpret and assess the results of algorithms used in automated analysis. These results can be a good starting point for further interactive analysis. Gerbil is equipped with two powerful methods that segment the data either according to spectral characteristics on a global level or based on topological relation and local similarity. In the latter case we bring supervised segmentation to the multispectral domain especially for the purpose of interactive inspection.

4.1. User-Guided Segmentation. Supervised segmentation incorporates user-provided prior knowledge. A user specifies a set of background and foreground pixels. All other pixels are then determined as belonging to either the background or the foreground. We make this concept a powerful tool within our interactive workflow.

The early version of Gerbil [6] included a rudimentary version of supervised segmentation. We have now adapted an existing algorithm family specifically to the multispectral domain [10].

4.1.1. Graph-Based Supervised Segmentation. In recent years, graph-based algorithms have dominated supervised segmentation on both grayscale and RGB images. In 2011, Couprie et al. introduced a framework for supervised segmentation that incorporates several key methods based on graph theory [40]. Their *power watersheds* integrate graph cuts, random walker, and watersheds algorithms in a single mathematical framework.

For this algorithm family, the input consists of two sets of pixels, the foreground seeds $\mathcal{F}$ and background seeds $\mathcal{B}$, as well as the pixel values $\mathbf{x}_i$, $1 \leq i \leq N$, which are strictly used in a differential manner. The topological relation of pixels is reflected in a graph structure. Each pixel i corresponds to a vertex v_i. A neighboring pixel j with corresponding v_j is connected to v_i via an edge e_{ij}. The edge weight w_{ij} of an edge e_{ij} is a function of the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. We compute an N-element vector $\mathbf{y}$, where $y_i \in [0,1]$ indicates that a pixel i belongs to foreground or background via

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}} \sum_{e_{ij} \in E} w_{ij}^p |y_i - y_j|^q + \sum_{v_i} w_{Fi}^p |y_i|^q + \sum_{v_i} w_{Bi}^p |y_i - 1|^q, \tag{4}$$
$$\text{s.t.} \quad \forall i \in \mathcal{F} : y_i = 1, \ \forall i \in \mathcal{B} : y_i = 0,$$

where w_{Fi} and w_{Bi} denote unary weights penalizing foreground and background affinity. Simple thresholding leads to a binary segmentation s with $s_i = 1$ (foreground pixel) if $y_i \geq 1/2$, and 0 (background pixel) otherwise. Based on the selection of parameters p and q, this minimization matches the graph cuts, random walker, or shortest paths algorithms. With $p = \infty$, $q \geq 1$, the power watershed algorithm is obtained. See Couprie et al. [40] for details.

Couprie et al. define the edge weights for grayscale and RGB images;

$$w_{ij} = \exp\left(-\beta\left(d\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)^2\right), \tag{5}$$

where β is a constant and $d(\cdot)$ is the L_2 norm or the L_∞ norm [40]. For hyperspectral data, these choices are reasonable if a single band or the PCA of the image is used as input. However when operating on the full spectra of the image, a more appropriate distance function should be employed.

4.1.2. Adaptation to Hyperspectral Data. In our previous work, we evaluated a range of distance functions on multispectral data, including established similarity measures for spectral mapping [41, 42] and a data-driven measure [10]. In spectral mapping, captured spectra are compared to a database of known spectra to determine a material. Hence, spectral mapping similarity measures are well suited for hyperspectral distance functions.

The best performing measurement in our experiments is the spectral angle from the Spectral Angle Mapper [15]. The spectral angle has the property that it disregards pure intensity changes. It is defined for two spectra $\mathbf{u}$ and $\mathbf{v}$ as

$$\operatorname{SA}(\mathbf{u}, \mathbf{v}) = \cos^{-1}\left(\frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|_2 \cdot \|\mathbf{v}\|_2}\right), \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $\| \cdot \|_2$ the L_2 norm.

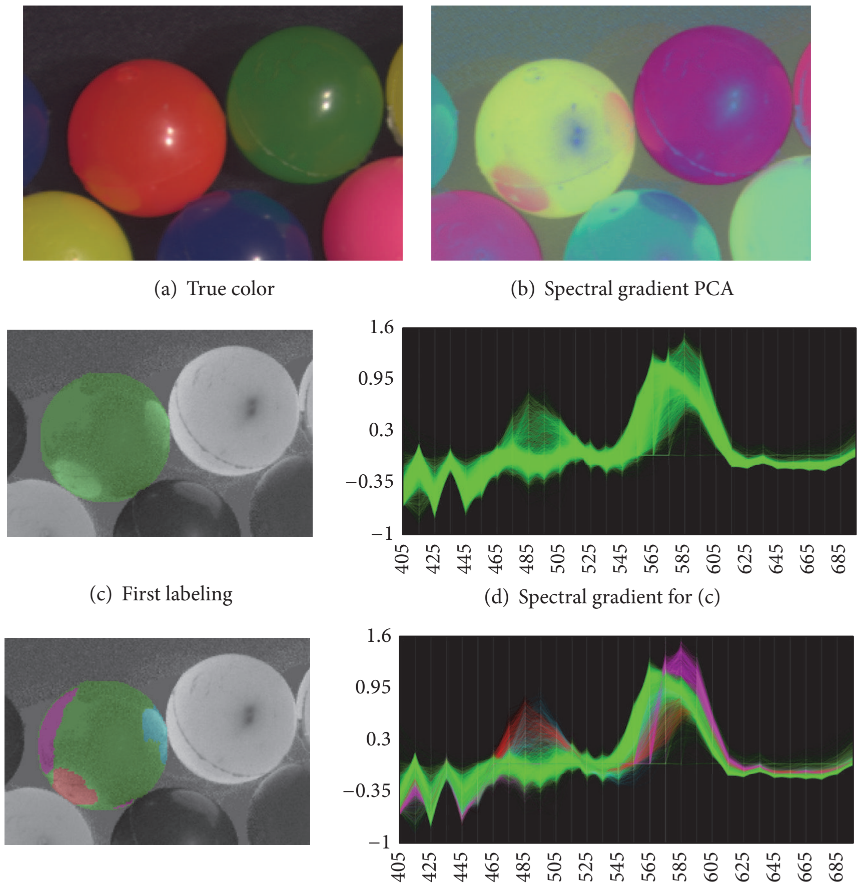
4.1.3. Example Use-Case. We demonstrate the use of supervised segmentation on the *superballs* image. It shows plastic balls of various color that highly reflect on each other. In Figure 6, the user investigates a red ball specifically. With an RGB camera we could only spot the reflection from the yellow ball (see Figure 6(a)). Calculating the PCA on the spectral gradient (as discussed in Section 5.2) also reveals the reflection from the green ball (see Figure 6(b)). However, the blue balls also reflect on the red ball, which is not revealed by these depictions. The first labeling, Figure 6(c), is obtained by setting one foreground seed in the middle of the ball and drawing a circle around the ball as background seeds. When looking at the spectral gradient distribution of that segment (see Figure 6(d)), we see the deviations from the distribution caused by these interreflections. Browsing through these parts of the spectral gradient distribution plot with the interactive highlight reveals their spatial locations. By setting corresponding seed points again, we can find three additional segments within the ball (see Figure 6(e)). The spectral gradient distribution plot (see Figure 6(f)) reveals how the pixels from these three segments contribute to the deviations in the distribution. To further understand them, the user might continue by also segmenting the surrounding balls, effectively adding them to the plot.

Our user-guided segmentation method is especially useful in interactive analysis. Segmentation results are helpful, quick to obtain, and easy to refine within the provided user interface.

4.2. Unsupervised Segmentation. While graph-based segmentation is valuable within an interactive session, it is object-driven in a sense that spatially connected pixels are grouped. However, often it is desired to explore the image as a whole, which means connecting pixels that share no spatial relation in the image, for example, to form a segment consisting of all scene objects that share the same albedo. Note that while the multiband thresholding detailed in Section 3.3 and illustrated in Figure 5(b) often allows finding such spatially disconnected material clusters in the spectral gradient manually, it can be a tedious task, especially when several material clusters in the scene are to be revealed. A global clustering method method however can reveal all material clusters in the scene and be used as a starting point for further analysis.

For this task we incorporate the *mean shift* algorithm into Gerbil. Mean shift has been used for global clustering of multispectral data for several tasks [43, 44]. Mean shift is a good choice due to its general purpose design. For example, while the well-studied and fast *k-means* algorithm works best on clusters with circular shape and the number of clusters needs to be defined a priori, mean shift is a density gradient estimator and it neither requires prior knowledge of the number of clusters nor constrains the shape of the clusters [45], which is desired property for analysis of arbitrary image data. A drawback of the original mean shift is computational complexity. The Fast Adaptive Mean Shift (FAMS) algorithm by Georgescu et al. [46] provides an important improvement to computation time by introducing an approximate nearest-neighbor search. However, the method still takes minutes to hours to segment a hyperspectral image. To facilitate the use of mean shift in interactive inspection, we introduced a new variant that combines mean shift with a superpixel presegmentation [11].

4.2.1. Superpixel Mean Shift Clustering. The mean shift algorithm works in feature space alone to obtain a global clustering rather than spatially constrained segments. It, thus, disregards any topological context. We reintroduce the use of spatial information by incorporating superpixel detection.



(a) True color (b) Spectral gradient PCA

(c) First labeling (d) Spectral gradient for (c)

(e) Second labeling (f) Spectral gradient for (e)

Figure 6: Inspection of cropped *superballs* image. Labelings are shown as overlay on the spectral gradient at 490–500 nm.

Both spatial context and superpixels gained attention recently as means to improve the performance of hyperspectral image processing tasks [47, 48].

In our algorithm, we first obtain a superpixel segmentation by adapting the method of Felzenszwalb and Huttenlocher [49] to hyperspectral data. We obtain a set of small, homogeneous regions. Then, FAMS is performed using these regions as data points. This significantly reduces the complexity of the mean shift operation. Overall computation times are in the range of five to ten seconds for images from the CAVE database [20]. The reliance on superpixels impacts the spatial resolution of the segmentation, in a way that often improves the spatial coherence of segments. A detailed presentation of this work can be found in [11].

4.2.2. Clustering Results and Refinement. In Figure 7, we show an example result on an image from the CAVE database. Clusters are assigned random colors with an equal spacing in the hue range. Our superpixel mean shift algorithm, denoted as MSSP, uses an established spectral matching measure [41] to form superpixels in the original feature space and then performs mean shift for each superpixel in the spectral gradient feature space. It is compared to FAMS operating in both the original feature space as well as the spectral gradient feature space (denoted as SG-FAMS). The results reveal that the spectral gradient distribution is more helpful for material clustering as compared to the original spectral distribution affected by geometry and that we can obtain a very good approximation of the SG-FAMS result with MSSP.

In Figure 8, we show another example result on an image from the Foster database, which other than the previous image captures a natural scene. Due to its high spatial resolution, clustering is a time-consuming task on this image. We compare our approach to the method by Huynh and Robles-Kelly [7], denoted HRK. Due to memory constraints, HRK was applied on a down-sampled (by a factor of two) version of the image and took 6 minutes. Note that, for HRK, the number of desired clusters needs to be set in advance. Our method took 154 seconds on the original image and 15 seconds on the down-sampled version.

Within Gerbil it is easy to further refine clustering results. The spectral distribution visualization reveals the compactness of a cluster. It helps the user to spot under-segmentation, where a single cluster contains several spectral profiles (as seen in Figure 6(d)). Oversegmentation, as is the case in Figure 8(c), can also be easily spotted by comparing the spectral distribution of two clusters. Gerbil provides an intuitive interface to merge such segments. To obtain

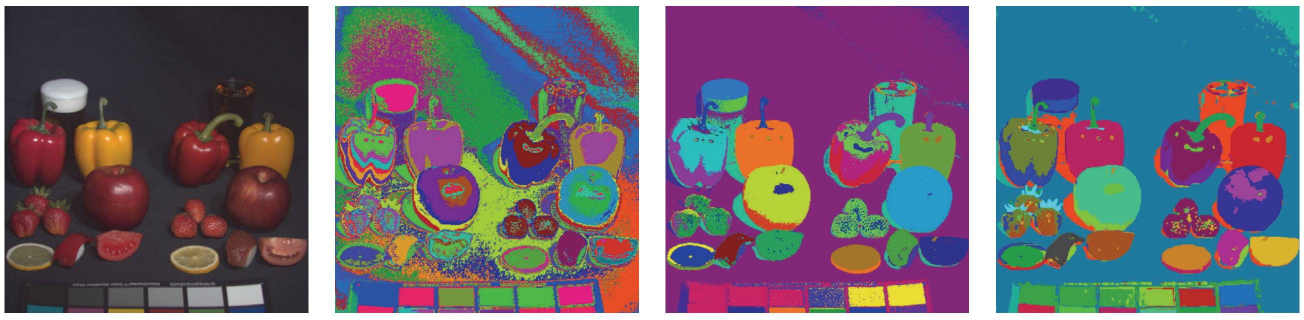
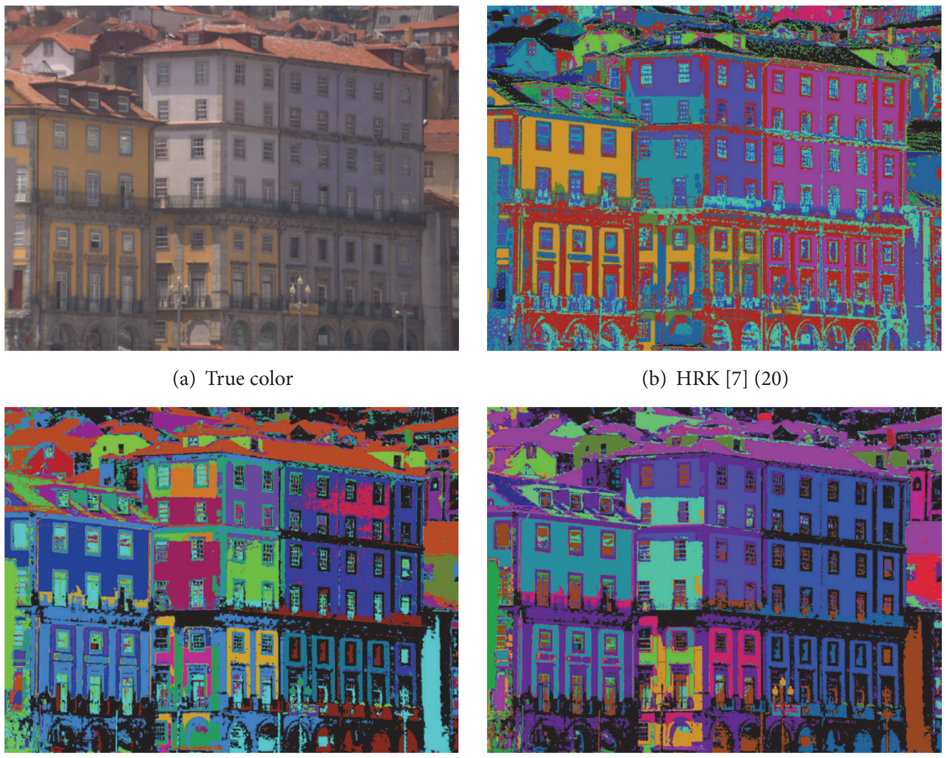

(a) True color (b) FAMS (41) (c) SG-FAMS (24) (d) MSSP (30)

FIGURE 7: Clustering results on *fake and real food* image. The number of clusters is shown in parentheses in the respective caption.


(a) True color (b) HRK [7] (20)

(c) MSSP (27) (d) MSSP refined (20)

FIGURE 8: Clustering results on *Ribeira* image. The number of clusters is shown in parentheses in the respective caption.

the result in Figure 8(d), the user performed four simple merging operations. The user may also run a supervised segmentation to refine the clustering, especially in the case of an undersegmented area.

4.3. Label Inspection and Manipulation. The segmentation methods in Gerbil provide beneficial input in a typical usage scenario. However, a good presentation of the results is equally important. Depending on the underlying data, mean shift clustering may produce a high number of meaningful segments. In the supervised scenario, often follow-up segmentations are used to locally refine obtained segments. Several measures aid in discerning and manipulating segmentations.

The first measure is the automatic determination of practical label colors. A preset of label colors consists of the primary and secondary palette, excluding yellow, which is reserved for temporary highlights. For more than five labels, colors are determined by an equal spacing in the hue range excluding yellow. Figures 1, 6, 8, and 10 show examples of automatically labeled segmentations. In cases where a temporary highlight falls onto labeled pixels, the corresponding pixels are drawn in their label colors significantly shifted towards yellow.

The second measure is a specific mode of operation where existing single labels can be selected through mouse hover in the topological view. In this mode, the user can better examine which pixels and spectra are part of a specific label.

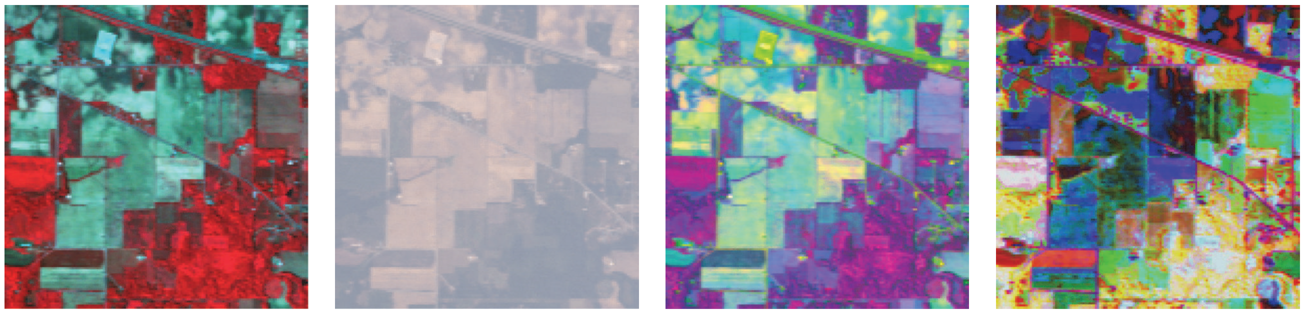

(a) Three-band composite (b) True color (c) PCA false color (d) SOM false color

FIGURE 9: Color visualizations of *Indian Pines* image. SOM visualization is more feature-rich when compared to PCA.

Several labels can be selected at once and then be either deleted or merged using the label manager (Figure 1(e)).

Finally, a label can be used to initialize a multiband limited highlight. The limits on each band are set to include all pixels contained in the label. This helps both in finding similar spectra that are not yet included in the label and in separating several clusters within the label by adjusting the limits. The remaining selection can then be added to another label or form a new label.

Gerbil is a very versatile tool for the creation of labelings for further use. A common use-case is the creation of ground truth data for evaluation of algorithms. For this tedious task, the user can start from a high-quality clustering or segmentation and quickly incorporate the information contained in all bands.

5. Color Coding and Color Display

A popular and useful method of visualizing multispectral and hyperspectral data is the generation of a color image that preserves the spatial relations in the image. While a grayscale image can present a single piece of information per pixel, a color image displays three data values within the `r`, `g`, `b` color triplet. There are four common categories of color displays for hyperspectral image data. (a) *Band composite*: three image bands are selected and then mapped to `r`, `g`, `b` color channels. (b) *Data fusion*: a subset of bands are fused according to a specific criterion. Most often, this is the human color perception, modeled by the CIE XYZ color-matching functions [30]. (c) *Dimensionality reduction*: the entire spectral information is reduced to three dimensions, which are mapped to `r`, `g`, `b`. (d) *Feature extraction*: a combination of application-specific indicators (e.g., contributions of three user-selected endmembers to each spectrum) forms a false-color image. For example, in [50], false-color images can depict the occurrence of forest types to assess forest structure.

In line with other analysis frameworks, Gerbil provides color displays based on human color perception (true color) and on linear dimensionality reduction via principal component analysis (PCA). Additionally, a fast nonlinear dimensionality reduction method based on a self-organizing map (SOM) was developed. Figure 9 depicts the results of the three methods on an example image.

5.1. Human Color Perception. For images whose spectrum lies within the range of visible light, an intuitive natural representation can be formed by mimicking human color perception. This representation is often referred to as *true color*. The CIE 1931 colorimetric system defines three color-matching functions $\overline{x}(\lambda), \overline{y}(\lambda), \overline{z}(\lambda)$ [30]. These functions describe the spectral sensitivity of a human observer, called the CIE 1931 standard colorimetric observer. The tristimulus values X, Y, and Z are then obtained by integrating the product of observed intensities and the respective color-matching function over the wavelength λ. In practice, the integrations are replaced by summations over bands of equal width $\Delta\lambda$ centered at wavelength λ [30]. Popular filter-based multispectral sensors capture the incoming light with $\Delta\lambda = 10$. The X, Y, and Z values are directly used in the CIE XYZ color space. They can be transformed to RGB given a reference white and a gamma correction value. We display true-color images in sRGB space as it comes with default values for both and is specifically designed for computer displays [51].

Scene illumination plays an important role in true-color visualization. As hyperspectral images provide a complete spectral response of each pixel and homogeneous lighting conditions are present in most data capture scenarios, illumination is easy to manipulate. Gerbil offers removal and exchange of illuminants [6] and includes a set of reference illuminants modeled as black-body radiators [30]. Apart from a true-color image with applied reillumination, the effect of the illuminant is also illustrated within the spectral distribution plot.

5.2. Principal Component Analysis. Objects of varied reflectance properties can yield the same color sensation, rendering true-color representations and their variants unsuitable for several applications. As an alternative, a *false-color* image can be created that puts emphasis on different characteristics of the data. A popular approach uses principal component analysis (PCA) [52], where the three components with maximum variance become the `r`, `g`, `b` values of a color image. Due to the nature of PCA, the per-component variance differs significantly. Therefore, an automatic white balancing is performed for display purposes [52].

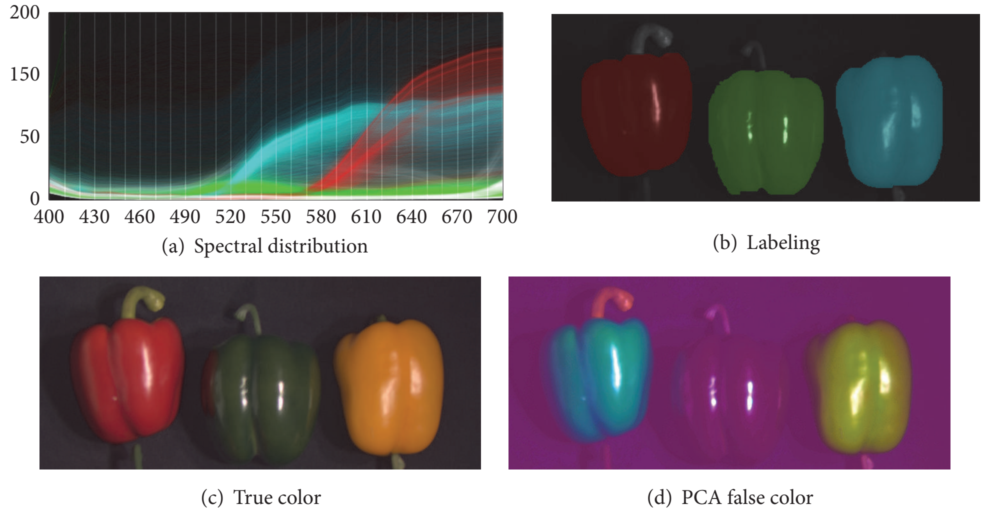


Figure 10: Displays of cropped *fake and real peppers* image. The spectral distribution (a) is label colored. (b) shows the respective labeling as an overlay on the image band at 530 nm.

Using three bases that explain most of the variance in the data is not always the most helpful representation for discerning relevant spectra. Figure 10 shows a counter example. The contrast of the green peppers against the background in Figure 10(d) is very low. It is a direct effect of the low object reflectance when compared to the red and yellow peppers in the image, as can be seen in Figure 10(a). This is unintuitive to human observers (Figure 10(c)) who have a high sensitivity for green. We would expect the false-color display to better distinguish the green pepper from the background.

The simultaneous availability of the parallel coordinates representation and the PCA-based false-color visualization allows one to easily judge the quality of the dimensionality reduction in comparison to the original image data.

5.3. Self-Organizing Map. A self-organizing map (SOM) converts the nonlinear statistical relationship between high-dimensional data into simpler geometric relationships [53]. In other words, a SOM provides a topological representation of the spectral vector distribution of a multispectral image. A typical SOM consists of a 1D array or 2D grid of *model vectors* (also referred to as neurons) that represent vectors in the original data space. The 3D SOM variant extends a 4-connected 2D lattice to a 6-connected 3D lattice [54]. The third dimension often enables the SOM to learn a more accurate topology on hyperspectral data. We exploit this topology for false coloring by mapping the 3D coordinates to the RGB cube [12, 55, 56].

5.3.1. Training and Traditional Approach. A SOM consists of n model vectors $\mathbf{m}_j \in \mathbb{R}^D$, with D corresponding to the number of bands in the input image. The 3D SOM is a cube of side length $n' = \sqrt[3]{n}$. For any input vector $\mathbf{x}$ (a pixel's captured spectrum), we can find the *best-matching unit* (BMU) $\mathbf{m}_c$ with index

$$c^{(\mathbf{x})} = \underset{j}{\operatorname{argmin}} \quad d\left(\mathbf{x}, \mathbf{m}_j\right), \tag{7}$$

where $d(\cdot)$ is the Euclidean distance. In our experiments we did not observe any advantage by changing the distance function, for example, to the spectral angle. The *location* $\mathbf{r}^{(j)} \in \mathbb{Z}^3$ of a neuron $\mathbf{m}_j$ describes its position in the 3D lattice topology and is a bijective mapping of j.

Our SOM training is unsupervised and uses s unlabeled pixels from the input image. Model vectors are initialized with random values. In each iteration $1 \le t \le s$, we randomly draw a spectrum $\mathbf{x}$. We first determine the BMU $\mathbf{m}_c$ of input $\mathbf{x}$. Then, all model vectors are updated as $\mathbf{m}_j^* = h_{c,j}(t) \cdot (\mathbf{x} - \mathbf{m}_j)$, where $h_{c,j}(t)$ defines the influence of $\mathbf{x}$ on the model vectors $\mathbf{m}_j$ based on the SOM topology. As suggested by Kohonen [53],

$$h_{c,j}(t) = \alpha(t) \cdot \exp\left(-\frac{\left\|\mathbf{r}^{(c)} - \mathbf{r}^{(j)}\right\|^2}{2\sigma^2(t)}\right). \tag{8}$$

The learning-rate factor $\alpha(t)$ is a user-adjustable parameter that is monotonically decreasing. The kernel width $\sigma(t)$ describes the sphere of influence of a sample vector in the SOM topology and is also monotonically decreasing. During the early training phase a SOM seeks a rough global ordering. In later iterations, local regions are smoothed out. We train the SOM on the input image itself on the fly and do not depend on any prior knowledge.

The trained SOM is then used as input for false coloring. In previous work [55], a false-color image has been created by finding BMU $c^{(\mathbf{x})}$ for each image pixel $\mathbf{x}$ (7). Then, an $\mathtt{r}$, $\mathtt{g}$, $\mathtt{b}$ triplet is created by scaling $\mathbf{r}^{(c)}$ by $255/n'$. However, this method leads to suboptimal visual quality and achieves even lower entropy values than PCA false coloring. The reason is the limited amount of model vectors, resulting in a strongly quantized output. In the configuration of [55], only 64 different color values could be produced. While we use SOMs of size $n = 10^3$, we are still far from the capabilities of an 8-bit color display. Training a SOM of larger sizes, however, would become infeasible both due to longer learning times and due to the limited amount of available training samples.

5.3.2. Improved Look-Up for False Coloring. To avoid the aforementioned quantized output, we introduce a novel look-up method [12]. Firstly, instead of using a single best-matching unit, we develop a set of best-matching units (BMUs). Furthermore, we order the BMUs according to the L_2 distance and assign a set of predetermined weights to the ordered set. These rank-based weights are crucial. While a simple unweighed combination would only smooth the result, using the L_2 distances directly as weights is not reliable [57]. In our high-dimensional space, distances would appear very close to each other and the weights would not discern well. We define rank weights that ensure both a majority contribution by the first BMU and significant contributions by the additional BMUs.

Consider, as in [58], a vector of BMU indices

$$\mathbf{c}^{(\mathbf{x})} = \underset{\mathbf{j}}{\operatorname{argmin}} \sum_{k=1}^{C} d\left(\mathbf{x}, \mathbf{m}_{j_k}\right), \tag{9}$$

where C is the number of desired BMUs. For each pixel value $\mathbf{x}$, we calculate a *representative location* $\mathbf{r}'$ as

$$\mathbf{r}' = \sum_{k=1}^{C} w_k \cdot \mathbf{r}^{(c_k^{(\mathbf{x})})}, \tag{10}$$

given weights w_k with the properties

$$\begin{gathered} w_k = 2w_{k+1}, \\ \sum_{k=1}^{C} w_k = 1, \\ \forall w_k,\ k < C, \\ d\left(\mathbf{x}, \mathbf{m}_{c_k}\right) < d\left(\mathbf{x}, \mathbf{m}_{c_{k+1}}\right), \\ \forall \mathbf{m}_{c_k},\ k < C, \end{gathered} \tag{11}$$

which expresses that the BMUs are sorted according to distance to the query vector and weighed by their rank, where the weight for rank k is always twice as high as the weight for subsequent rank $k + 1$. The representative location $\mathbf{r}' = \left(r_1' \ r_2' \ r_3'\right)$ describes the position in the learned topology that best represents $\mathbf{x}$. We finally obtain

$$\begin{aligned} \mathtt{r} &= \frac{r_1'}{n'}, \\ \mathtt{g} &= \frac{r_2'}{n'}, \\ \mathtt{b} &= \frac{r_3'}{n'}. \end{aligned} \tag{12}$$

Our method gives a high-quality, nonlinear dimensionality reduction. The advantage over previous false-color work based on SOMs is that we do not suffer from quantization effects. The advantage to other nonlinear methods, for example, ISOMAP [59], is speed. While ISOMAP takes minutes to hours to compute, we train a SOM with $n = 10^3$ in less than 20 seconds on a typical hyperspectral remote sensing image. As most computation is spent in training, execution time is not significantly affected by image size. Rather, it mostly depends on the SOM size and the dimensionality of the data.

The SOM itself can be visualized within Gerbil during, as well as after, its training. The topological views correspond to the traditional SOM-based data visualization and reveal the SOM topology. The parallel coordinate plots depict the distribution of the model vectors, and make it visually comparable to the distribution of the training data. These visualizations help in both assessing the training and in fine-tuning its parameters. Figure 11 depicts a remote sensing image and its spectral distribution view as compared to the spectral distribution of a SOM trained on the image, as well as the ten principal components computed by PCA.

The true-color and false-color display capabilities of Gerbil provide and extend the state of the art in hyperspectral visualization. They are especially helpful for selecting regions of interest (ROIs) that should be examined. In large images, due to clutter and computational expense, it is beneficial to select a ROI before performing further visualization tasks. Additionally, the false-color displays can give a good first impression of the data, for example, to spot the same reflectance in several regions of the image or to find inhomogeneities within a depicted object. The user is directed to specific regions for starting an investigation. An illustrating example is the *fake and real peppers* image, as depicted in Figure 12. In the SOM false-color display pixels with the same material or reflectance properties are consistently colored. One such example are the specular highlights. Another instance is the stems of the plastic and real peppers as annotated in Figure 12.

6. Software Framework

Gerbil is built around the idea of being able to form a base for other research projects. We ensure this by working with code libraries that are already popular in image processing and by a modular and easily expandable design.

6.1. Software Foundation. Gerbil is written in C++. It relies on OpenCV [8] for all image processing tasks and for the

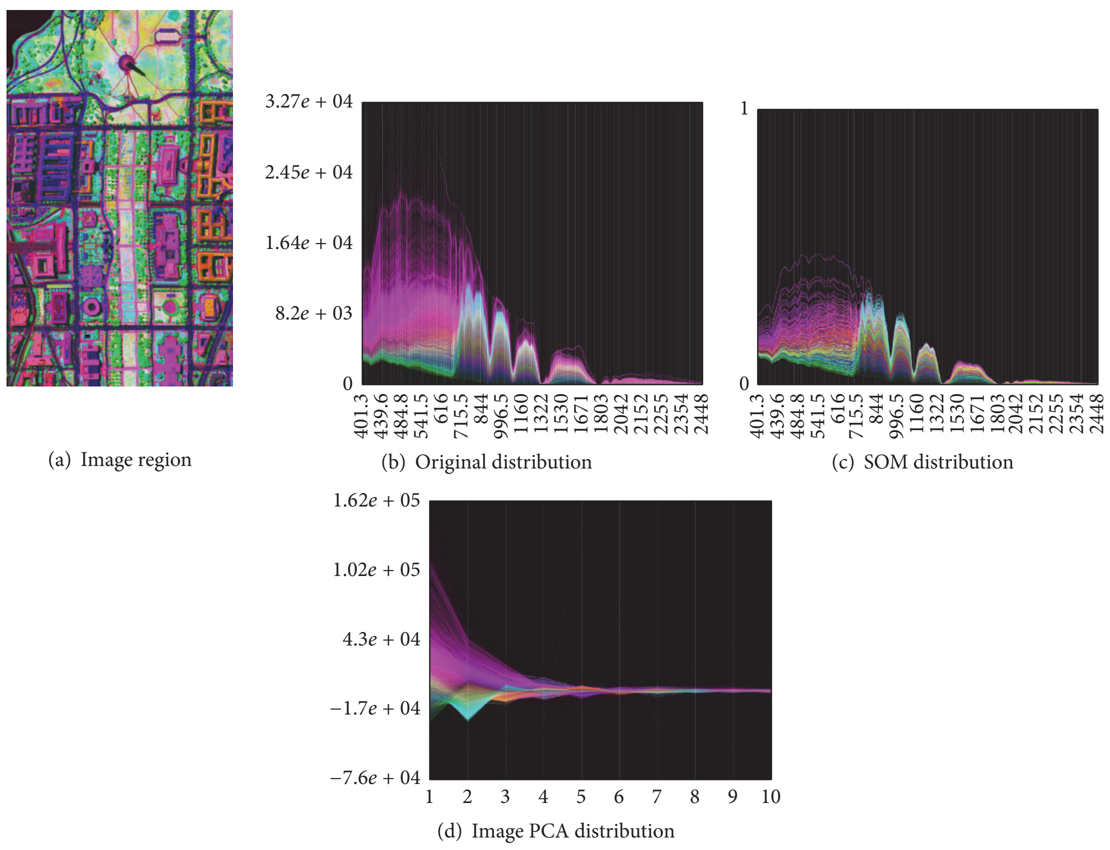


Figure 11: Center part of the *D.C. Mall* image and parallel coordinates visualizations computed from the data. Pixels and spectral vectors are colored using the SOM visualized in (c).

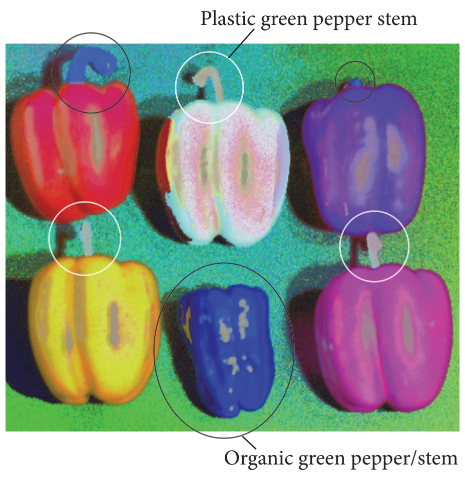


Figure 12: Figure 1(e), spectral gradient SOM display. The occurrences of two materials in their respective color coding (blue and light gray) are annotated with circles.

internal representation of multispectral data. Due to the tight integration of Gerbil and OpenCV it is very easy to apply OpenCV's extensive functionality on multispectral data within Gerbil. Furthermore, when available, the software utilizes CUDA through OpenCV for some of its common calculations. For image I/O we incorporate the Geospatial Data Abstraction Library (GDAL) [17]. It reads a large variety of image formats from various sources, including FITS, LAN, GIS, and the ENVI.hdr header formats.

For all operations related to the graphical user interface, the Qt framework [60] is used. Together with a CMake-based build system we maintain a truly cross-platform software that runs on a variety of operating systems, including all recent versions of Windows, MacOS X, and GNU/Linux.

6.2. *Internal Design.*

Our software framework consists of both a powerful core and a modular system of extensions. In its core, Gerbil provides a flexible, easy to use programming interface to the image data. It is built on an internal representation of multispectral images that combines both per-band and per-pixel data structures [6].

6.2.1. *Modularity.*

Gerbil is organized in modules that work independently of each other and of the user interface. A module can (optionally) depend on external libraries and other modules. Both the graphical user interface of Gerbil and a sophisticated command-line interface are stand-alone units that expose the functionality of the other modules to the user. The command-line interface enables batch processing of image analysis tasks.

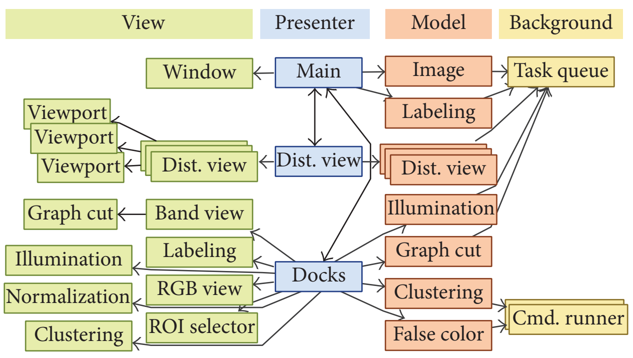


Figure 13: GUI software design. Presenter components control both view and model components.

6.2.2. Parallelization and Backgrounding. Performance and responsiveness are both crucial for interactive analysis. OpenCV utilizes vector instruction sets in CPU computation. In order to employ all CPU cores, we parallelize computationally demanding tasks via Intel Thread Building Blocks [61]. We also heavily recycle previously computed data. For example, image representations and distribution histograms are incrementally updated when the current region of interest is changed.

Computations need to run in the background to keep the interface responsive to the user and enable the user to cancel a running task. We introduce a custom shared datatype with locking facilities and an atomic swap operation. Background tasks calculate new data on their own shadow copy and lock the data for a single swap when finished. By using a sequential background queue and defined data swaps at the end of each task, we guarantee that at most two copies of the data are held in memory at once. An exception to this are tasks that can run for a longer time on their own and are not part of the regular user interaction. Examples are global clustering or computation of false-color representations. These tasks run independently on their own data copy.

6.2.3. GUI Software Design. A significant level of sophistication is required in creating Gerbil's GUI in order to expose the framework's functionality to the user while efficiently reacting to user input. We model the GUI module after the model–view–presenter [62] design pattern. Figure 13 gives an overview of the GUI components. *View* components display data and receive user input. Image data, as well as other representations, for example, distribution histograms, is handled by the respective *model* components. These components have access to both the background queue to enqueue tasks and may spawn command runners which can run in the background. The *presenter* components handle user requests and trigger calculations. They are in control of both view and model components, which do not see each other. For data synchronization, Qt's signal/slot messaging [60] is employed.

With this design we ensure a strict separation between models and views and reduce unnecessary overhead in data synchronization code where signals can be passed through.

6.2.4. Extensibility. With our software design it is easy to extend the functionality of Gerbil. New methods can be integrated as modules in the framework. They benefit from a defined interface for parameter setting and command execution through the command-line interface. They may also (optionally) depend on other modules which provide a standardized method call. To add a new method in the GUI, a new dock widget can be created as a view component, while interfacing data calculation with the corresponding module in a model component. The high level of abstraction on several levels keeps interfaces simple and prevents side effects of changes in other components.

7. Conclusions

Gerbil contains several key components that are novel and essential to interactive visual analysis of multispectral and hyperspectral image data. We introduce interactive visualization via parallel coordinates, supervised segmentation via a graph-cut algorithm family, fast global clustering using a combination of superpixel segmentation and mean shift density estimation, and fast nonlinear false coloring based on a 3D self-organizing map. In Gerbil, we tightly integrate these algorithms into a consistent framework that allows the user to explore the data in a novel way. Existing research and commercial software are often comprised of a rich toolbox of useful algorithms for specific applications. However, the goal of a general software framework that also provides intuitive access to the raw image data is unmet. Gerbil fills this gap within a modern architecture and GUI.

In recent years, multispectral capture has gained wider adoption across a range of applications, in particular in cultural heritage. Domain experts who did not work with multispectral data before need assistance in understanding and utilizing the modality. The new intuition in working with multispectral images that Gerbil strives to provide may therefore broaden the scope of multispectral analysis and foster adoption.

Gerbil is an active open-source project and accepts contributions on Github [63]. As of May 2016, it includes 25 902 physical source lines of code (SLOC) in 326 source code files with a total of 37 273 lines. In the past 12 months, 3622 downloads of the Gerbil software package were counted, originating from 45 countries.

For future work, more methods that are established in remote sensing need to be incorporated and adapted to interactive analysis, as done in [33]. Drawing techniques for the spectral distribution views should be further investigated, for example, for clutter reduction. Gerbil is an evolving project that constantly seeks more input from diverse application domains.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors gratefully acknowledge funding of this work by the German Research Foundation (GRK 1773 Heterogeneous Image Systems and Open Access Publishing Programme) and funding of the Gerbil open-source project by the European Space Agency (ESA SOCIS 2012–2016).

References

[1] R. O. Green, M. L. Eastwood, C. M. Sarture et al., "Imaging spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)," *Remote Sensing of Environment*, vol. 65, no. 3, pp. 227–248, 1998.

[2] F. A. Kruse, A. B. Lefkoff, J. W. Boardman et al., "The Spectral Image Processing System (SIPS)—interactive visualization and analysis of imaging spectrometer data," *Remote Sensing of Environment*, vol. 44, no. 2-3, pp. 145–163, 1993.

[3] N. Gat, "Imaging spectroscopy using tunable filters: a review," in *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 4056 of *SPIE Society of Photo-Optical Instrumentation Engineers*, pp. 50–64, Bellingham, Wash, USA, 2000.

[4] J. Fisher, M. M. Baumback, J. H. Bowles, J. M. Grossmann, and J. A. Antoniades, "Comparison of low-cost hyperspectral sensors," in *Proceedings of the SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, vol. 3438, pp. 23–30, San Diego, Calif, USA, July 1998.

[5] Forth Photonics, "MuSIS," July 2012, http://archive.is/musis.forth-photonics.com.

[6] J. Jordan and E. Angelopoulou, "Gerbil—a novel software framework for visualization and analysis in the multispectral domain," in *Vision, Modeling and Visualization*, pp. 259–266, 2010.

[7] C. P. Huynh and A. Robles-Kelly, "A probabilistic approach to spectral unmixing," in *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 344–353, Springer, Berlin, Germany, 2010.

[8] G. Bradski, "Open Source Computer Vision Library," Dr. Dobb's Journal of Software Tools, 2016, http://opencv.org/.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[10] J. Jordan and E. Angelopoulou, "Supervised multispectral image segmentation with power watersheds," in *Proceedings of the 19th IEEE International Conference on Image Processing (ICIP '12)*, pp. 1585–1588, IEEE, Orlando, Fla, USA, September-October 2012.

[11] J. Jordan and E. Angelopoulou, "Mean-shift clustering for interactive multispectral image analysis," in *Proceedings of the 20th IEEE International Conference on Image Processing (ICIP '13)*, pp. 3790–3794, IEEE, Melbourne, Australia, September 2013.

[12] J. Jordan and E. Angelopoulou, "Hyperspectral image visualization with a 3-D self-organizing map," in *Proceedings of the Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS '13)*, pp. 1–4, IEEE, 2013.

[13] J. W. Boardman, L. L. Biehl, R. N. Clark et al., "Development and implementation of software systems for imaging spectroscopy," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '06)*, pp. 1969–1973, IEEE, July-August 2006.

[14] L. Biehl, "Eval-ware: hyperspectral imaging [best of the web]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 125–126, 2007.

[15] P. E. Dennison, K. Q. Halligan, and D. A. Roberts, "A comparison of error metrics and constraints for multiple endmember spectral mixture analysis and spectral angle mapper," *Remote Sensing of Environment*, vol. 93, no. 3, pp. 359–367, 2004.

[16] L. Biehl and D. Landgrebe, "MultiSpec—a tool for multispectral-hyperspectral image data analysis," *Computers & Geosciences*, vol. 28, no. 10, pp. 1153–1159, 2002.

[17] F. Warmerdam, "The geospatial data abstraction library," in *Open Source Approaches in Spatial Data Handling*, G. B. Hall, M. G. Leahy, S. Balram, and S. Dragicevic, Eds., vol. 2, pp. 87–104, Springer, Berlin, Germany, 2008.

[18] Harris Geospatial Solutions, "ENVI," 2016, http://www.harrisgeospatial.com/ProductsandSolutions/GeospatialProducts/ENVI.aspx.

[19] J. Boardman, F. Kruse, and R. Green, "Mapping target signatures via partial unmixing of AVIRIS data," in *Proceedings of the Summaries, 5th JPL Airborne Earth Science Workshop*, vol. 1, pp. 23–26, 1995.

[20] F. Yasuma, T. Mitsunaga, D. Iso, and S. Nayar, "Generalized assorted pixel camera: post-capture control of resolution, dynamic range and spectrum," Tech. Rep., Columbia University, 2008.

[21] D. H. Foster, K. Amano, S. M. C. Nascimento, and M. J. Foster, "Frequency of metamerism in natural scenes," *Journal of the Optical Society of America A*, vol. 23, no. 10, pp. 2359–2372, 2006.

[22] Purdue Research Foundation, "Hyperspectral images," May 2016, https://engineering.purdue.edu/_biehl/MultiSpec/hyperspectral.html.

[23] United States Army Corpse of Engineers, "Hypercube," May 2016, http://www.erdc.usace.army.mil/Media/FactSheets/FactSheetArticleView/tabid/9254/Article/610433/hypercube.aspx.

[24] Ball Aerospace & Technologies Corp, "Opticks," May 2016, https://opticks.org/.

[25] "Opticks-Spectral Processing Extension," July 2012, https://opticks.org/display/opticksExt.

[26] M. Cui, A. Razdan, J. Hu, and P. Wonka, "Interactive hyperspectral image visualization using convex optimization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 6, pp. 1673–1684, 2009.

[27] W. A. Joye and E. Mandel, "New features of SAOImage DS9," in *Astronomical Data Analysis Software and Systems XII*, vol. 295, pp. 489–492, 2003.

[28] H. Li, C.-W. Fu, and A. J. Hanson, "Visualizing multiwavelength astrophysical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1555–1562, 2008.

[29] P. Colantoni, R. Pillay, C. Lahanier, and D. Pitzalis, "Analysis of multispectral images of paintings," in *Proceedings of the European Signal Processing Conference*, Florence, Italy, 2006.

[30] G. Wyszecki and W. S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, Wiley-Interscience, New York, NY, USA, 2nd edition, 2000.

[31] S. J. Kim, S. Zhuo, F. Deng, C.-W. Fu, and M. Brown, "Interactive visualization of hyperspectral images of historical documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1441–1448, 2010.

[32] National ICT Australia Limited, “Scyven,” May 2016, http://www.scyven.com.

[33] B. Labitzke, S. Bayraktar, and A. Kolb, “Generic visual analysis for multi- and hyperspectral image data,” *Data Mining and Knowledge Discovery*, vol. 27, no. 1, pp. 117–145, 2013.

[34] N. Keshava and J. F. Mustard, “Spectral unmixing,” *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 44–57, 2002.

[35] E. Angelopoulou, S. W. Lee, and R. Bajcsy, “Spectral gradient: a material descriptor invariant to geometry and incident illumination,” in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV ’99)*, vol. 2, pp. 861–867, Kerkyra, Greece, September 1999.

[36] A. Inselberg and B. Dimsdale, “Parallel coordinates: a tool for visualizing multi-dimensional geometry,” in *in IEEE Conference on Visualization*, pp. 361–378, 1990.

[37] R. Gonzalez, R. Woods, and S. Eddins, *Digital Image Processing*, Prentice Hall Press, New York, NY, USA, 3rd edition, 2008.

[38] G. Ellis and A. Dix, “Enabling automatic clutter reduction in parallel coordinate plots,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 717–724, 2006.

[39] R. Fuchs and H. Hauser, “Visualization of multi-variate scientific data,” *Computer Graphics Forum*, vol. 28, no. 6, pp. 1670–1690, 2009.

[40] C. Couprie, L. Grady, L. Najman, and H. Talbot, “Power watershed: a unifying graph-based optimization framework,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011.

[41] Y. Du, C.-I. Chang, H. Ren, C.-C. Chang, J. O. Jensen, and F. M. D’Amico, “New hyperspectral discrimination measure for spectral characterization,” *Optical Engineering*, vol. 43, no. 8, pp. 1777–1786, 2004.

[42] A. E. Gutiérrez-Rodríguez, M. A. Medina-Pérez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, “dissimilarity measures for ultraviolet spectra identification,” in *Advances in Pattern Recognition: Second Mexican Conference on Pattern Recognition, MCPR 2010, Puebla, Mexico, September 27–29, 2010. Proceedings*, vol. 6256 of *Lecture Notes in Computer Science*, pp. 220–229, Springer, New York, NY, USA, 2010.

[43] S. Bo, L. Ding, H. Li, F. Di, and C. Zhu, “Mean shift-based clustering analysis of multispectral remote sensing imagery,” *International Journal of Remote Sensing*, vol. 30, no. 4, pp. 817–827, 2009.

[44] X. Huang and L. Zhang, “An adaptive mean-shift analysis approach for object extraction and classification from urban hyperspectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 12, pp. 4173–4185, 2008.

[45] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[46] B. Georgescu, I. Shimshoni, and P. Meer, “Mean shift based clustering in high dimensions: a texture classification example,” in *Proceedings of the 9th IEEE International Conference on Computer Vision*, pp. 456–463, October 2003.

[47] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, “Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 11, pp. 3804–3814, 2008.

[48] D. R. Thompson, L. Mandrake, M. S. Gilmore, and R. Castaño, “Superpixel endmember detection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4023–4033, 2010.

[49] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[50] S. L. Ustin and A. Trabucco, “Using hyperspectral data to assess forest structure,” *Journal of Forestry*, vol. 98, no. 6, pp. 47–49, 2000.

[51] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta, “A standard default color space for the internet-sRGB,” Tech. Rep., Microsoft and Hewlett-Packard, 1996, http://www.color.org/sRGB.xalter.

[52] N. P. Jacobson and M. R. Gupta, “Design goals and solutions for display of hyperspectral images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 11, pp. 2684–2692, 2005.

[53] T. Kohonen, *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*, Springer, Berline, Germany, 3rd edition, 2001.

[54] J. Jordan and E. Angelopoulou, “Edge detection in multispectral images using the n-dimensional self-organizing map,” in *Proceedings of the 18th IEEE International Conference on Image Processing (ICIP ’11)*, pp. 3181–3184, IEEE, Brussels, Belgium, September 2011.

[55] J. Gorricha and V. Lobo, “Improvements on the visualization of clusters in geo-referenced data using self-organizing maps,” *Computers & Geosciences*, vol. 43, pp. 177–186, 2012.

[56] J. M. Fonville, C. L. Carter, L. Pizarro et al., “Hyperspectral visualization of mass spectrometry imaging data,” *Analytical Chemistry*, vol. 85, no. 3, pp. 1415–1423, 2013.

[57] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ‘nearest neighbor’ meaningful?” in *Database Theory—ICDT’99*, C. Beeri and P. Buneman, Eds., vol. 1540 of *Lecture Notes in Computer Science*, pp. 217–235, 1999.

[58] M. Sjöberg and J. Laaksonen, “Optimal combination of SOM search in best-matching units and map neighborhood,” in *Advances in Self-Organizing Maps*, vol. 5629, pp. 281–289, Springer, Berlin, Germany, 2009.

[59] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina, “Improved manifold coordinate representations of large-scale hyperspectral scenes,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2786–2803, 2006.

[60] The Qt Company, “Qt-cross-platform application and UI framework,” 2016, http://qt-project.org/.

[61] J. Reinders, *Intel Threading Building Blocks: Outfitting C++ for Multi-Core Processor Parallelism*, O’Reilly Media, 2010.

[62] M. Potel, *MVP: Model-View-Presenter the Taligent Programming Model for c++ and Java*, Taligent, 1996.

[63] J. Jordan, *Gerbil Hyperspectral Visualization and Analysis Framework*, 2016, http://github.com/gerbilvis/gerbil.