Histomages: Fully Synchronized Views for Image Editing

Fanny Chevalier
Department of Computer Science
University of Toronto, Canada
fanny@dgp.toronto.edu

Pierre Dragicevic
INRIA
F-91405 Orsay, France
dragice@lri.fr

Christophe Hurter
University of Toulouse
ENAC, France
christophe.hurter@enac.fr

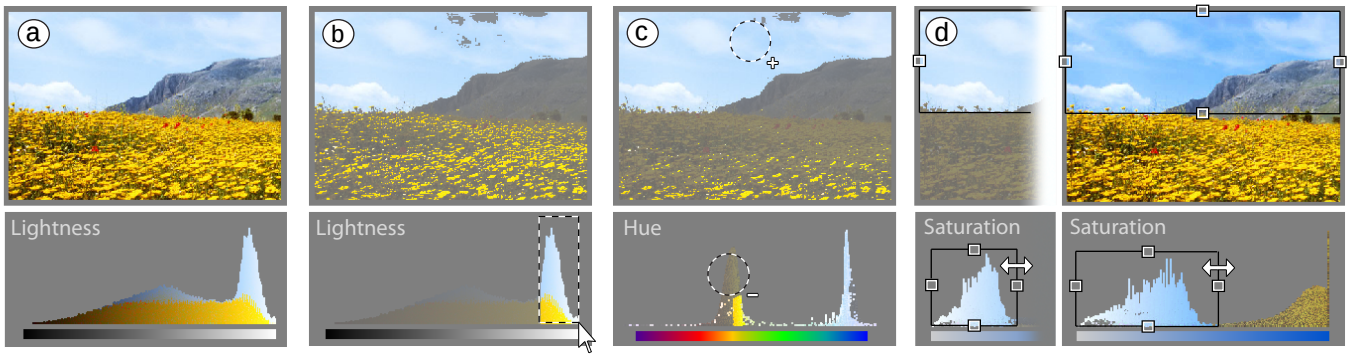

Figure 1: Sky enhancement with Histomages: (a) the image is duplicated and its pixels rearranged into a lightness histogram; (b) bright pixels are selected with the rubber-band selection tool; (c) all pixels are rearranged into a hue histogram and yellow pixels are filtered out with the "subtract" selection brush (bottom). Missing pixels are added with the "add" selection brush on the image (top); (d) the sky is enhanced by resizing the selection on the saturation histogram.

ABSTRACT

We present Histomages, a new interaction model for image editing that considers color histograms as spatial rearrangements of image pixels. Users can select pixels on image histograms as they would select image regions and directly manipulate them to adjust their colors. Histomages are also affected by other image tools such as paintbrushes. We explore some possibilities offered by this interaction model, and discuss the four key principles behind it as well as their implications for the design of feature-rich software in general.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors.

Keywords: Image editing, synchronized views, histograms.

INTRODUCTION

Bitmap image editing software like Photoshop or Gimp is extremely powerful in the hands of experts but is hard to master, because of the wealth of functionality and tools that must be learned. Taken individually, most of these tools are relatively easy to learn and to use (e.g., levels, hue shift, color range selection or selection transforms). However, they each follow their own logics and have a different user interface. In order to make image manipulations more flexible and fluent, image editing software needs more unified, coherent interaction models.

UIST'12, October 7-10, 2012, Cambridge, MA, USA.

We present Histomages, a new interaction model based on synchronized views that supports rich image selection and manipulation operations through simple tools. Although synchronized views have been used in various domains, their support in current image editing software is still rudimentary. In addition to illustrating the benefits of full support for synchronized views in image editing, Histomages more generally demonstrate the benefits that can be gained from focusing on an *interaction model* rather than a *swiss army knife* approach to the design of feature-rich applications.

BACKGROUND

We first briefly review previous work in interaction models, image editing tools and synchronized views.

Interaction Models

An interaction model is "a set of principles, rules and properties that guide the design of an interface" [4]. Interaction models can range from general principles (e.g., direct manipulation) to more specific guidelines and mechanisms. They help achieve logical soundness and consistency in user interfaces, with benefits to both developers and end users of complex software.

Like many authoring and productivity applications, image editors are primarily based on the *swiss army knife* principle (i.e., a large set of specialized, independent and complementary tools), along with transversal mechanisms such as selection, undo and redo, copy and paste, scripts and layers.

Software companies mainly focus on improving existing tools and adding more powerful ones (often through plug-in mechanisms), while the underlying interaction models are considered immutable. Since these models only weakly enforce consistency between tools – each new tool such as an image filter typically comes with specific controls – image editing

applications are increasingly difficult to master and more and more intimidating to beginners as new versions appear. In this article we explore an alternative model for two types of image editing tasks: *selection* and *color manipulation*.

Image Selection

Selecting image regions is a key task in image editing and several tools are generally provided. They include *image-space* tools (e.g., lasso or rubber-band selection) and *color-space* tools (e.g., color range selection). Recently, a method based on color naming has been proposed [17]. *Mixed-space* tools combine both approaches. They include the magic wand and other "intelligent" cutout tools whose algorithms are still an active topic of research [22, 28]. Some of these tools dynamically select image regions while the image is being painted [25]. Advanced selection tools can dramatically facilitate common image editing tasks, but the more elaborate they are, the more difficult it is for users to build a clear mental model of how they work and to predict their results.

Since no selection tool alone can address all selection tasks no matter how sophisticated it is, it is also essential to be able to combine different types of selection techniques. This is achieved in image editing software through the support of *boolean operations* on selections, a powerful feature that acts as a bridge between selection methods and allows users to, e.g., select a region using a color picker and tolerance slider, then refine it by lassooing on the image. However, since different types of selections involve different interaction techniques, switching between them can still be tedious.

Color Manipulation

Color manipulation is another key task in image editing, particularly in photo editing. The state-of-the-art interaction model consists in filters whose parameters are set in control panels, with immediate preview on the image. The tools are numerous and include luminosity and contrast correction, color balance, and color replacement. Adobe Photoshop CS5 provides 21 color manipulation tools, 3 of which are automatic and 18 have parameters that are exposed in specialized dialog boxes [1]. Techniques have also been proposed for controlling parameters directly on the image [24, 14].

Popular tools among photographers are image histograms, which give a rapid overview of how color components (e.g., brightness) are distributed in an image and what corrections need to be applied. Most image editors include an interactive histogram that can be stretched by moving control points, which in turn adjusts colors on the image. Although useful, current histogram tools are often difficult to relate to the image and require trial and error, or thorough practice.

A few professional applications provide more advanced interactive histograms, such as Adobe Lightroom 4 where histograms can be locally stretched by direct manipulation, although only the vertical boundaries of a few predefined histogram regions can be moved, which effectively amounts to moving control points [2]. Capture NX 2 provides histograms where value ranges can be selected and the corresponding pixels are shown on the image with a blinking animation [24]. However, this technique cannot be used to select regions of the image for subsequent manipulation.

Synchronized Views

Early uses of the principle of synchronized views can be found in the visual exploration system by Becker and Cleveland [6] and in the MVC design pattern from the Smalltalk programming language [21], later implemented in a number of GUI toolkits. Today, examples of applications with elaborate support for synchronized views include document editing environments that combine a markup editor with a WYSIWYG window [11], and a wealth of visual exploration tools [27, 26]. These tools support *synchronized highlighting*, i.e., hovering over an element in one view highlights the same element in the other, and sometimes *synchronized editing*, i.e., edits in either view are reflected in the other.

Synchronized views have been extensively employed and studied in the field of information visualization under the name of *coordinated views* (see [27] for a survey). The use of multiple data representations is thought to promote effective visual exploration, especially if tools are provided to help analysts relate different views. A popular approach in the domain is synchronized highlighting, termed *brushing and linking* [6, 7]. Other solutions involve drawing links between views [10] or using animated transitions [16]. Synchronized view configuration mechanisms have also been proposed [26], but synchronized editing has comparatively received little attention in this field, with a few exceptions [3].

Image editing applications also involve alternative image representations (mostly histograms) but with very primitive support for synchronized views: interaction with histograms is poor and requires tools that are completely distinct from other tools. Yet histograms are just another way of laying out image pixels and could be therefore treated (at least partly) as images. The similarity between histograms and images has been evoked in art [15] but to the best of our knowledge, it has never inspired any actual tool. In the following we use simple scenarios to illustrate some of the possibilities offered by such an interaction model for image editing.

HISTOMAGES WALKTHROUGH

Histomages employ a classic multi-window image editor interface and a toolbar that contains a small but fully-functional set of traditional image editing tools (Figure 2). The only visible difference is an extra toolbar at the top of each window. One button is for cloning the window (similar to the Clone Window menu command in Photoshop) while other buttons switch between the image view and several types of histogram views (e.g., red, green and blue channels, LAB lightness or hue) and a 2-D lightness/hue scatterplot.

In Histomages, a histogram is a *spatial rearrangement* of the image's pixels (Figure 1a): all pixels are displayed with their initial color but instead of being laid out on a grid, they are grouped into bins depending on the value of the color channel that is visualized. Pixels overlap vertically in case the histogram does not fit the window, and their stacking order reflects their y-ordering in the original image. Thus, pixels remain coherently grouped, which produces a visual effect similar to stacked graphs [8] and makes it easier to identify image regions. In addition, view switches are smoothly animated by having each pixel rapidly move to its new location.

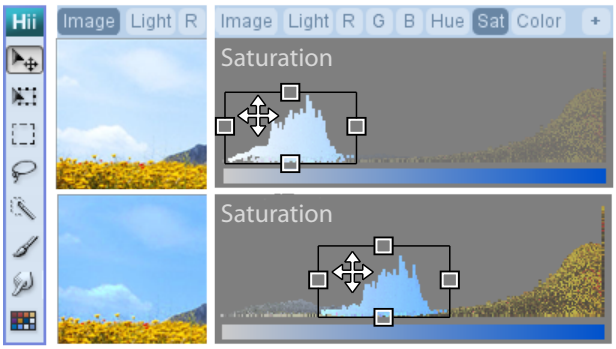

Figure 2: Increasing the saturation of the sky.

Since histograms are merely images with alternative pixel layouts, all operations that can be done on images can also be done on histograms. We illustrate this with a simple scenario: Pamela was asked to illustrate the cover of a new novel called "The Promised Land". She finds a stock photography (Figure 1a top) and feels it would fit perfectly with a few adjustements. She considers making the sky more blue and the hill more verdant. She also finds that the color of the carpet of yellow flowers is too aggressive and needs to be softened.

Adding and Switching Views

After loading the image in Histomages, Pamela duplicates the window so that one view will be kept to display the image while the other view will show histograms. She first selects the lightness histogram. As the image transitions to the histogram, she sees the sky region quickly move to the right while the mountain region piles up at the center (Figure 1a). On the final histogram, she sees that the two regions exhibit a clean separation in terms of lightness, as opposed to the yellow flowers that clearly span the whole luminosity range.

Selecting Regions of Histograms

Pamela first focuses on the sky. She picks the rubber band selection tool and selects the rightmost region of the lightness histogram (Figure 1b). This effectively selects the brightest pixels on both the histogram and the image view. Unselected pixels fade out to better show the selection.

Although unwanted pixels have been included in the selection, Pamela is aware that the sky and the flowers differ enough in hue. She switches to the hue histogram and indeed sees the flower region and the sky region split out into two separate clusters. Using the brush selection tool and holding the key modifier for the "subtract" mode, she quickly removes all yellow pixels as well as other isolated pixels outside the blue range (see Figure 1c, bottom). As some pixels from the sky are still missing, she switches to the "add" mode and brushes over the missing regions directly on the image (see Figure 1c, top). Now the sky is properly selected.

Dragging and Stretching Histogram Regions

Since the sky appears washed out, Pamela switches to the saturation histogram and indeed notices that the selected pixels are all squeezed to the left, in the low saturation range (see Figures 1d and 2). Therefore, she picks the transform tool and drags the selected region to the right, towards higher saturation values on the histogram. This effectively changes the color of all selected pixels so that their saturation match their

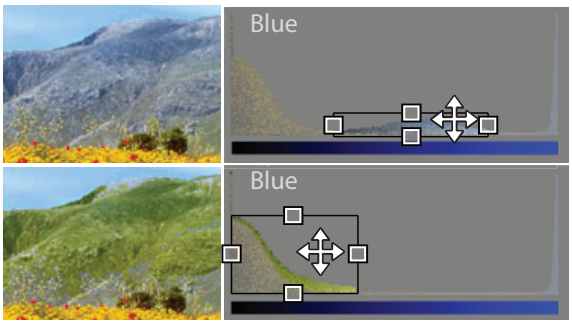

Figure 3: Reducing the blue component of the hill.

new x-position on the histogram (see Figure 2). While she drags the selection, Pamela sees the sky change on the image, and she stops once she is satisfied with the color.

Although increasing saturation made the sky more blue overall, Pamela does not find the effect very natural. Since clouds are generally supposed to be white, their saturation should probably be left unchanged. Therefore, Pamela drags the selection back to the left, then grabs the right handle and stretches the selection towards more saturated values (Figure 1d). Her idea is to increase the dynamic range of the selected pixels on the saturation channel, and while she is dragging the handle, she does notice that this type of color transformation works much better visually.

More Selecting and Dragging

The various histograms provide many different ways of selecting and recoloring image regions. As another example, when Pamela switches to the hill, she browses through different histograms and finds that it forms a clear bump in the blue channel (Figure 3 top). She brushes the bump then cleans up the selection on the hue and the lightness channels. She then adjusts the hill to a greener tone by dragging the selection to the left of the blue channel. Again, the image is dynamically updated, which in turn updates the histogram in a way that preserves the y-ordering of pixels and makes them appear to "crawl" over the yellow region (Figure 3 bottom).

Two-Axis Scatterplots

To make yellow colors less aggressive and poppies stand out, Pamela decides to retouch the carpet of yellow flowers. She brings up the hue histogram but realizes that the hue channel does not do a good job at separating yellow and green pixels, which in addition to the stems now also include the hill.

Therefore, Pamela switches to a 2-D scatterplot visualization of the color space, where the vertical axis is mapped to hue and the horizontal one to lightness (Figure 4 left). Although the scatterplot does not show pixel counts like histograms, it is appropriate for selecting complex ranges of colors. In that case, Pamela needs to select dark to light yellows and orange tones while avoiding red, green and dark pixels. She picks the brush selection tool, applies it starting from a bright yellow color, then uses the immediate selection feedback on the image to help her adjust the trajectory of the brush (Figure 4 middle). She occasionally uses the subtract mode to clean the selection or zooms in to gain more precision.

Once the yellow flowers are properly selected, their hue and lightness can be simultaneously adjusted by freely dragging

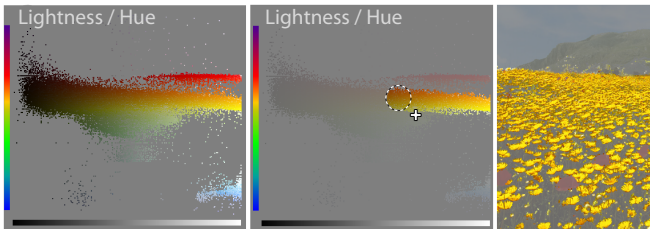

Figure 4: Selection in the Lightness/Hue space.

Figure 5: Applying the paintbrush on a view temporarily freezes all histograms (middle). At mouse release, painted pixels jump to their new location (right).

the selection on the scatterplot. However, since Pamela only needs to wash out the colors, she switches back to the saturation channel and drags the selection to the left. By dragging the selection too far she accidentally discovers that a carpet of white flowers produce a soothing and enchanting effect and thus decides to stick to white flowers.

Painting Histograms

Histomages are also compatible with other traditional image editing tools, which opens up a range of other possibilities in terms of color manipulation. Right after she finished her novel cover, Pamela is asked to produce urgently a cover for a special issue of the Garlic Magazine on a personality.

Pamela downloads a photograph of the personality and decides to produce a Warhol-style posterized cover. She makes four copies of the photo then loads the produced image in Histomages. She then picks the paintbrush tool, and on the red channel histogram, she paints a stripe in a uniform orange (Figure 5). She uses the image preview as a guide, while the histogram is temporarily frozen to prevent the brush from accidentally affecting neighboring pixels. At mouse release, all pixels jump to the corresponding bin of the histogram.

Pamela then paints other stripes of the histogram. Since she wants to preserve the lightness component of the hair, she uses the transform tool and shrinks the selection to zero width instead. Finally, she selects one of the faces on the image view and drags it on the lightness/hue scatterplot until she is satisfied with the variant produced. Through the same method, she produces variants for the two remaining faces.

Quick Comparison with Photoshop

The principal strength of Histomages does not lie in its expressive power (the image manipulations shown can be done with any professional photo editing application) but in the elegance and the concision of its interactions. For example, as and far as our expertise goes, carrying out the task illustrated in Figure 1 with Photoshop CS3 would require a user to:

(1) Select the *magic wand* or bring up the *color range selection* dialog (*Selection* menu) ; click on a pixel of the sky and adjust the *tolerance* value (text field or slider) until the sky is properly selected ; possibly change the seed pixel and repeat ; if no combination of seed pixel and tolerance is found to work (which is the case in our example), break down the task into several steps using boolean selections through *key modifiers* ; Close the dialog, pick a tool like *polygon selection* and clean up the selection on the image. (2) Bring up the *Hue/Saturation* dialog (*Image* $\rightarrow$ *Adjustment* menu or Ctrl+U) and push the *saturation* slider to the maximum. This happens to yield a result similar to the one on Figure 1c, but if a result closer to Figure 2 was desired instead, it is not clear how this could be done. Supposing the *Levels* tool supports saturation, the *Hue/Saturation* dialog box would have to be closed (all dialogs are modal), and the *Levels* dialog box would have to be brought up (*Image* $\rightarrow$ *Adjustment* or Ctrl+L) in order to see the histogram and its control handles.

DISCUSSION

We illustrated some of the possibilities of Histomages, a prototype application that is based on a new interaction model for image editing, and rethinks the way selection and color manipulation tasks are carried out. The goal of Histomages is to demonstrate new concepts, more so than to provide a finished product. Therefore, our prototype has limitations and possible extensions that we discuss. We then generalize these concepts and discuss their possible implications for the design of feature-rich applications.

Use by Novices

Retouching images on histograms requires a good understanding of image histograms, i.e., the meaning of different color axes like hue or saturation, and the principle of counting and vertically accumulating pixels of similar value. Although Histomages is initially not meant to be a learning tool, we believe that features like pixel color preservation and animated transitions can help novices understand histograms. We also think that letting users freely explore and manipulate various types of histograms in a consistent manner may have an educational value too. These, however, are only conjectures that need to be further tested in user studies.

Limitations of Selections in Histograms

Selection in histograms is arguably more difficult when there is little contrast between regions. This problem is however common to all color-space selection tools, which are meant to be used when the region of interest differs enough in color. Histomages supports the same tasks but in a more elegant and flexible manner, since it is based on a clear interaction model rather than on black-box color selection algorithms.

Again, no selection tool can solve all selection problems. Intelligent mixed-space selection tools are also limited, as selection is more than automatically detecting contours or color clusters. Ultimately, only the user can tell what is her region of interest so the solution likely lies in familiar, simple and powerful interactions. Boolean operations on selections brought lots of power by allowing selection tools to be combined, while Histomages pushes simplicity and flexibility further by adding logical consistency to these tools.

Currently, Histomages only supports binary selection. A useful extension would be support for fuzzy selections, with support for antialiasing and feathering operations both in the image and in the histogram spaces. Partially selected pixels would travel slower while dragged on a histogram or on a scatterplot, resulting in less prominent edges after color transformation and more natural-looking images. In addition, selected histogram regions are currently manipulated independently from unselected regions, but it could also useful to have a "rubber sheet" feature that preserves boundaries in a similar way to Capture NX 2's direct manipulation tool [24].

Non-linear Color Transfer Functions

Our current prototype allows histogram manipulation with the *selection transform* tool exclusively (move & resize) and therefore only supports linear color transfer functions. Traditional image histograms usually include a third control point and therefore allow for more flexibility in transfer functions, while the Curve tool from Photoshop CS5 supports up to 14 control points. However, in practice 3 to 4 control points are enough to specify transfer functions as complex as S-curves. This is the number of control points supported by interactive geometrical deformation tools such as Photoshop's *warp transform*, so supporting these tools on histograms would allow specifying non-linear color transfer functions.

Generalization to Other Image Editing Tools

We demonstrated two ways of extending traditional image editing tools so that they can be used on histograms: *(a)* having the tool affect histogram pixels the same way as image pixels (selection and paintbrush) and *(b)* interpreting changes in pixel positions within the histogram's coordinate system (the selection transform tool). We experimented with only a few tools, but other common editing tools could be extended to work on histograms following the same two approaches.

A major class of image editing tools are tools that affect pixel colors locally. Beside the paintbrush we implemented, these include the airbrush, eraser, pencil, and brushes that affect pixel lightness or hue. All could be easily modified to work like our paintbrush. We expect these to be useful for retouching non-contiguous close-to-uniform regions. For example, for red eyes removal a user could roughly select the eyes area on the image then paint the hue histogram with a hue brush. This procedure contrasts with current tools, which first require to explicitely select the exact area to be painted.

Another class of tools are pixel displacement tools, of which our selection transform tool is an example. Other geometrical transform tools like the skew, distort, perspective and warp tools could be extended the same way, as well as local displacement tools like the smudge tool. Applied to histograms, these tools could enrich the repertoire of color manipulations. We already mentioned possibilities warp tools could offer in terms of non-linear color transformations.

Other categories of tools would require more experimentation before one can tell whether or not they could be useful on histograms and how to extend their semantics. These include spatial filters (e.g., blur and sharpen) and pixel duplication tools (e.g., copy-paste and the clone stamp). Finally, some tools such as writing text might not make sense at all.

Alternative Pixel Layouts

We illustrated our interaction model on histograms and scatterplots, but other types of pixel layouts can be explored. Visual exploration and complex selections might be facilitated with multidimensional visualizations techniques such as parallel coordinates [18] or scatterplot matrices [13], where data points would be pixels and the dimensions would be color components and pixel coordinates. Some layouts could also be user-defined, in order to let users create meaningful pixel groups and easily reuse them for subsequent manipulations. Especially promising are techniques that combine both approaches such as star plots [20], where users directly manipulate axes to create and explore linear combinations of dimensions, or dynamic filtering lenses [19], where users can switch between different dimension mappings locally.

An Interaction Model for Rich Multi-View Applications

The interaction model behind Histomages reuses and refines the paradigm of synchronized views and is based on the following four general principles:

1. Consistent View Structures. All views are made of similar graphical primitives (pixels in our case) which try to retain their graphical attributes across views. This gives a sense of "conservation of matter" and can make it easier for users to build a mental model of how views are constructed (e.g., what an image histogram means), and how regions of different views relate to each other once they are constructed.

2. Consistent View Interactions. Any tool used to interact with a view can also be used on other views, with identical or similar effects. This is especially relevant for *selection* tools, because if both multi-view selections and boolean operators are supported, powerful "selection sculpting" tasks can be carried out [13]. Furthermore, tool consistency can be beneficial to facilitate learning, since what has been learned for one view can be transferred to other views with little effort (e.g., our paintbrush). Finally, the semantics of some tools can be adapted to the specificities of different views in order to enrich their expressive power (e.g., our transform tool). Overall, applying this principle requires breaking traditional master/slave relationships between views [27] in favor of multidirectional dependencies [26, 12], and removing traditional couplings between tools and views in favor of tool polymorphism [4, 5].

3. Constantly Synchronized Views. The visual consistency between views and the underlying data is maintained at all times, except during modifications that require a view to be temporarily frozen (e.g., using a paintbrush on a histogram). This principle, together with principles 1 and 2, ensures that the system fully supports the traditional synchronized views paradigm: synchronized highlighting (or brushing and linking) is supported through selection tools, while synchronized editing is supported through regular editing tools.

4. Smooth View Transitions. Transitions between views within the same window are smoothly animated, which can further help explain how views are built and clarify how they are mapped [16, 9, 11]. The design and implementation of animated transitions are greatly facilitated by the principle 1. We believe this interaction model can make it easier for both

experts and beginners to use multi-view interfaces, and can also make interaction less tedious and more rewarding.

However, this interaction model alone cannot support the large variety of tasks supported by feature-rich commercial applications and is not meant to be used as a replacement. It is rather a complement to existing tools and mechanisms and a way to "polish" their logics. Although integrating this model in existing software might require significant refactoring, it does not necessarily involve many changes from the user's perspective. Histomages is fully integrated with the traditional GUI of image editors: it uses the same multi-window environment and similar visual representations (with some visual enhancements to histograms), and the selection and editing tools it supports are essentially the same.

CONCLUSION

There has been much focus on the problem of "software bloat", often with the assumption that not all features of large applications are useful, or at least not for everyone and not all the time [23]. However, complex activities such as image editing cannot be properly carried out without mastering a wealth of tools that are all complementary and all relevant to many tasks. We believe that the major issue with most feature-rich applications is not an excess of functionalities but a lack of logics and consistency in the way they are exposed at the user interface. Often, most development efforts are spent designing new tools, without much concern for the interaction models that allow these tools to work together.

With Histomages, we show that with a proper interaction model, a small number of tools can provide a large set of functionalities (with sometimes unexpected features), while ensuring some logical coherence. Histomages also suggests that new interaction models do not necessarily have to disorient expert users and can be sometimes integrated into existing GUIs without major changes.

ACKNOWLEDGMENTS

We would like to thank Dustin Freeman and Jean-Daniel Fekete for their comments on this paper.

REFERENCES

1. Adobe. Photoshop CS5. `www.photoshop.com`, 2011.
2. Adobe. Photoshop Lightroom 4. `www.adobe.com/products/photoshop-lightroom`, 2012.
3. T. Baudel. From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets. In *Proc. ACM User interface software and technology*, UIST '06, 67–76, 2006.
4. M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Proc. ACM Human factors in computing systems*, CHI '00, 446–453, 2000.
5. M. Beaudouin-Lafon and W. E. Mackay. Reification, polymorphism and reuse: three principles for designing visual interfaces. In *Proc. ACM Advanced visual interfaces*, AVI '00, 102–109, 2000.
6. R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, May 1987.
7. A. Buja, D. Cook, and D. F. Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5:78–99, 1996.
8. L. Byron and M. Wattenberg. Stacked graphs & geometry & aesthetics. *IEEE Trans. Vis. & Comp. Graphics*, 14:1245–1252, 2008.
9. F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete. Using text animated transitions to support navigation in document histories. In *Proc. ACM Human factors in computing systems*, CHI '10, 683–692, 2010.
10. C. Collins and S. Carpendale. Vislink: Revealing relationships amongst visualizations. *IEEE Trans. Vis. & Comp. Graphics*, 13(6):1192–1199, 2007.
11. P. Dragicevic, S. Huot, and F. Chevalier. Gliimpse: Animating from markup code to rendered documents and vice versa. In *Proc. ACM User interface software and technology*, UIST '11, 257–262, 2011.
12. P. Dragicevic, G. Ramos, J. Bibliowitcz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh. Video browsing by direct manipulation. In *Proc. ACM Human factors in computing systems*, CHI '08, 237–246, 2008.
13. N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Trans. Vis. & Comp. Graphics (Proc. InfoVis)*, 14(6):1141–1148, 2008.
14. N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Color lens: Adaptive color scale optimization for visual exploration. *IEEE Trans. Vis. & Comp. Graphics*, 17(6):795–807, 2011.
15. D. Friedman. Idea: The histogram as the image. Blog post `http://goo.gl/2PfcV`, 2007.
16. J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Trans. Vis. & Comp. Graphics (Proc. InfoVis)*, 13:1240–1247, 2007.
17. J. Heer and M. Stone. Color naming models for color selection, image editing and palette design. In *Proc. ACM Human factors in computing systems*, CHI '12, 237–246, 2012.
18. P. Hoffman, G. Grinstein, and D. Pinkney. Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In *Proc. 1999 workshop on new paradigms in information visualization and manipulation*, NPIVM '99, 9–16, 1999.
19. C. Hurter, A. Telea, and O. Ersoy. Moleview: An attribute and structure-based semantic lens for large element-based plots. *IEEE Trans. Vis. & Comp. Graphics (Proc. InfoVis)*, 17(12):2600–2609, Dec. 2011.
20. E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proc. IEEE Information Visualization Symposium, Late Breaking Hot Topics*, 9–12, 2000.
21. G. E. Krasner and S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Object-Oriented Programming*, 26–49, 1988.
22. Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graphics*, 23(3):303–308, Aug. 2004.
23. J. McGrenere and G. Moore. Are we all in the same bloat ? In *Proc. of the Graphics Interface Conference*, 187–196, 2000.
24. Nikon. Capture NX 2 v2.3.0. `www.capturenx.com`, 2012.
25. D. R. Olsen, Jr. and M. K. Harris. Edge-respecting brushes. In *Proc. ACM User interface software and technology*, UIST '08, 171–180, 2008.
26. N. H. Riche, B. Lee, and C. Plaisant. Understanding interactive legends: a comparative evaluation with standard widgets. *Comput. Graph. Forum*, 29(3):1193–1202, 2010.
27. J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. IEEE Coordinated and Multiple Views in Exploratory Visualization*, 61–71, 2007.
28. J. Wang, M. Agrawala, and M. F. Cohen. Soft scissors: an interactive tool for realtime high quality matting. In *ACM SIGGRAPH*, 2007.