Customized Slider Bars for Adjusting Multi-dimension Parameter Sets

Shigeru Owada[1], Makoto Okabe[2], Takeo Igarashi[3],
Frank Nielsen[4], Norimichi Tsumura[5]

[1] Sony CSL `sowd@acm.org`
[2] The University of Tokyo `makoto21@ui.is.s.u-tokyo.ac.jp`
[3] The University of Tokyo / Sony CSL `takeo@acm.org`
[4] Sony CSL `frank.nielsen@acm.org`
[5] Chiba University `tsumura@faculty.chiba-u.jp`

Abstract. We propose a meta user interface to design the user interface to tune vector-valued parameters of a given parametrized model. In our framework, the user defines a desired number of slider bars to control the parameters. The number of slider bars is independent from the dimension of the original parametrized model, which is important to design perceptually plausible user interface. Those slider bars correspond to straight axes in the parameter space. If the axes are not perpendicular each other, change of the value of one slider bar affects other slider bars, which significantly degrades the usability of the system. Our system can minimize this effect by simply enabling the lock attribute of each slider bar, which orthogonalizes the corresponding axes in the parameter domain. We show an image color adjustment software as our application example. However, the underlying idea of our system can be applied to any parametrized models.

1 Introduction

User Interface (UI) is a critical part of software systems, especially for contemporary computerized systems where the amount of computational resources such as program execution speed or storage size are huge enough to perform quite difficult tasks in real-time but the limitation of human ability constrains significant portions of the efficiency of the whole process.

Although many UI concepts and concrete techniques have been proposed since the birth of interactive systems [1, 2], the importance of the user interface design have only recently been widely recognized. Actually, UI is one of the most essential parts of software systems, because it is the element that the user directly touches, manipulates and observes. Therefore, software designers should carefully design UI for their products.

Application softwares usually support a single set of user interaction tools that are supposed to be optimal. It is satisfactory if the UI is really optimal for everyone. However, since the user's intent and preference vary one by one, such fixed toolkits may cause inefficiency of the UI. Therefore, instead of designing a fixed set of UI toolkits, we are more interested in customizing UI for independent users.

The idea of customizing UI is not new. One of the traditional ways is to define a macro which gathers a sequence of operations into a very simple one-shot operation. Macros are not only a standard tool to simplify the user's interactions, but also a method to redefine the original operations into a new set of operations, defined by the user. In this sense, it is a customization of UI. Macros are useful for discrete set of operations, but do not well capsule the user's intent for changing smoothly various variables, usually controlled by a specific type of UI toolkit such as slider bars. Our method customizes such user interface toolkit. The user can define arbitrary number of customized slider bars that match their subjective preference.

One issue here is that the axes that correspond to the slider bars may not be orthogonal each other in the parameter space because the user's preference does not necessarily produce independent components, and also because the number of sliders may exceed the dimension of the parameter space. It means that a change in one slider bar value can also change other slider bar values. We try to minimize this effect by introducing the "Lock" property to each slider. Locking the slider bar orthogonalizes the axes in the parameter space, which fixes motions of undesired slider bars.

Through this work, we try to show that the customizable UI is efficient, and also orthogonalization of tools is essential to improve the efficiency of the UI.

2 Previous work

Customization of user interface is an important domain of UI research [3] since recent computer systems are multi-purposed while the user's preferences and purposes are diverse.

Macro is a standard way to pack a sequence of operations into a compact form that is easy to reuse. This is a common way to improve the usability of a software in the user-desired way. For example, Microsoft Office (R) has a macro definition mode where the user's operations are recorded. The recorded macros are easily evoked by pressing the corresponding icon on the tool bar or pressing a key on the keyboard. Autodesk Maya (R) also has the similar function to pack a sequence of commands into a special button on the tool bar. Some programming languages such as C or Lisp also have macros, which are useful to simplify and abstract the source code.

User's burden can also be reduced by compressing the space of possible operations. Principal component analysis (PCA), nonnegative matrix factorization (NMF), locally linear embeddings (LLE), isomap, etc. are all used in this context. Especially, PCA and NMF are very widely used in computer graphics domain [4, 5], and some use the reduced space for modeling new data [6, 7]. There are other trials to reduce the operation space by adding constraints on examples to be mixed [8] or creating a completely new, psychophysically-based space by using multidimensional scaling (MDS) technique [9]. Design galleries is another interesting general framework to select a parameter set, based on semi-random presentation of candidates and human selection [10]. However, their focus is on how to explore the parameter space, instead of subjectively designing a customized UI toolkit.

3 User interface

In our system, the user explicitly defines the desired number of slider bars one by one. It is performed by giving examples for each slider bar.

The user first creates a new slider bar by pressing the “Add New Axis” button (Fig.1, the top of the right window). Then the system adds one slider bar under the examples area (Fig.1, the bottom of the right window). The user drops several desired examples onto the newly created bar (Fig. 2). The system internally computes the tendency of the dropped examples and defines the slider bar. By moving the knob on the slider bar, the user can smoothly modify parameter values and the result is displayed on the target image window (Fig. 1 left).

The user can optionally click the “Lock” box on the left of the slider bar to suppress the interference between slider bars (Fig. 3).

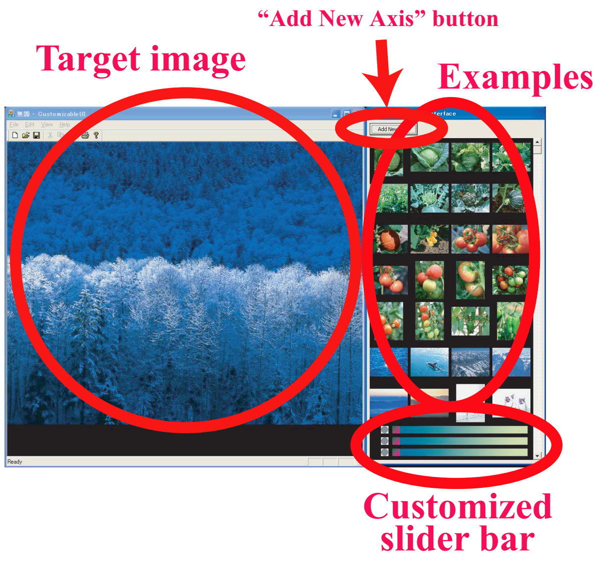

Fig. 1. UI design

4 Implementation

This section describes how to compute locked and unlocked axes and ranges of the slider bars from the user-specified examples.

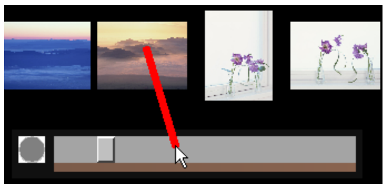
Fig. 2. Dragging an image onto the slider bar

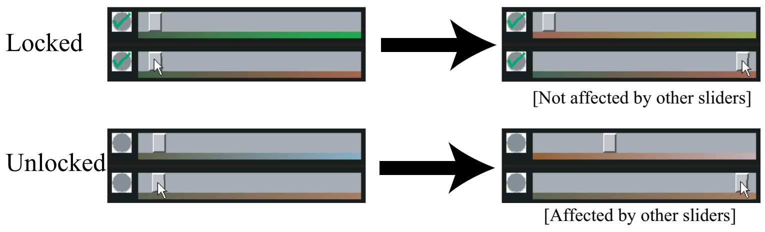

Fig. 3. Locked and unlocked slider bars (the slider bars are constructed from the same set of examples)

The user-supplied examples are given as a set of valid parameters. Let d as the dimension of the parameters. Then the examples are represented as a set of points in d-dimensional space.

Axis definition with orthogonalization For each slider bar, we apply the principal component analysis (PCA) to the corresponding examples to find the best approximating unit axis vector $\mathbf{A}_i$ of the dataset, where i is the index of the axis. If two or more axes are "Locked", those axis orientations should be orthogonal. We compute such new orientations by iteratively minimizing the following quadratic function:

$$\sum_i ||\mathbf{A}_i - \mathbf{A}_i^{new}||^2 + \sum_i \sum_{j \neq i} (\mathbf{A}_j \cdot \mathbf{A}_i^{new})^2,$$

where $||\cdot||$ is the L2-norm of a vector, i, j are indices of the locked axes, and $\mathbf{A}_i^{new}$ is the new axis to be computed, which is normalized to be unit length and replaces $\mathbf{A}_i$ after each iteration. This function consists of two terms. The first term tries to retain the previous vector, while the second term tries to orthogonalize each pair of axes. The number of iteration is 3000 in our experiment. Fig. 4 shows the graph of the convergence of this algorithm. Two 3D vectors $(1, 0, 0), (cos(\theta), sin(\theta), 0)$ are input and the dot product of those vectors after each step is plot. θ is varied from $\pi/10$ to $9\pi/10$.

Axis range We also define a line $\{t\mathbf{A}_i : t \in \Re\}$ and orthogonally project all example points (regardless of which slider it is corresponding to) onto the line to find the maximum and the minimum values of t, which corresponds to the range of the slider bar.

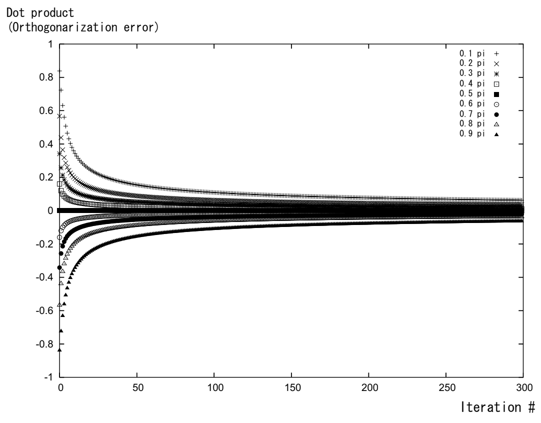

Fig. 4. Plot of convergence status

5 Result

We applied our technique for adjusting average color of images. The input of the system is a set of images and we take the average of each in (R, G, B) triplets. The user explores the RGB space by their customized slider bars and finds the desired color, which is then applied to the given image (Fig.5). This process is detailed in the accompanied video.

We observed that the “Lock” function is quite useful. Although locking the axes does not completely remove the mutual dependence (because we stop the optimization loop before convergence-See Fig.4.), it strongly suppresses the simultaneous knob motions, which significantly improves the user-friendliness of the system (Fig. 3).

6 Conclusion

We observed that customized tool bar efficiently summarizes the user’s interest, and locking the axes significantly improves the usability. However, we have not tested the possibility of our system completely. Especially, our idea can be applied to other kinds of parametrized models such as textures, bi-directional reflectance function (BRDF) and so on. It is also our future work to perform user test to verify the performance of our system.

References

1. Sutherland, I.E.: Sketchpad: A Man-Machine Graphical Communication System. Ph.D. Thesis. MIT (1963)
2. Kay, A., Goldberg, A.: Personal dynamic media. IEEE Computer **10** (1977) 31–41

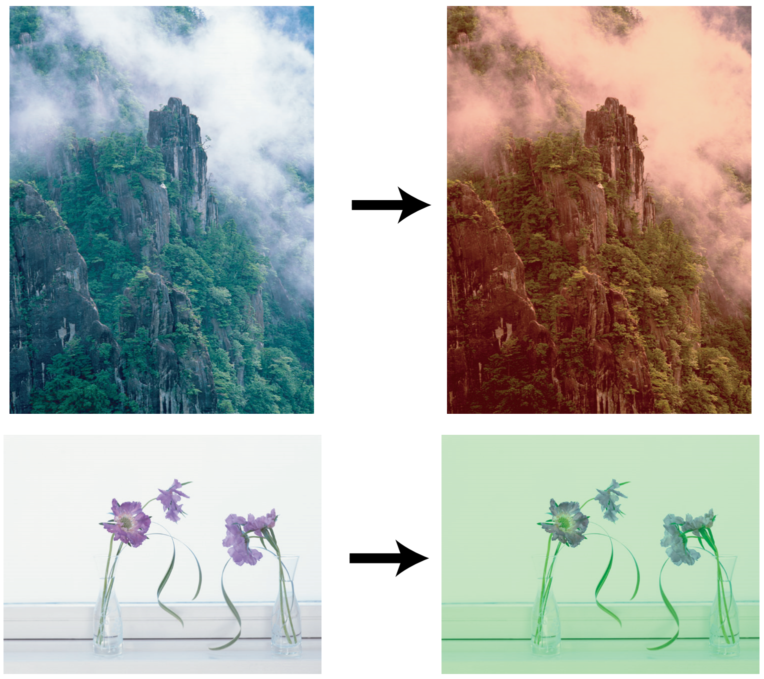

Fig. 5. Change of color specified by our system

3. Bergman, L., Lau, T.: Workshop on behavior-based user interface customization. In: IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces, New York, NY, USA, ACM Press (2004) 372–373
4. Sloan, P.P., Hall, J., Hart, J., Snyder, J.: Clustered principal components for precomputed radiance transfer. ACM Trans. Graph. **22**(3) (2003) 382–391
5. Lawrence, J., Rusinkiewicz, S., Ramamoorthi, R.: Efficient brdf importance sampling using a factored representation. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, New York, NY, USA, ACM Press (2004) 496–505
6. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (1999) 187–194
7. Matusik, W., Pfister, H., Brand, M., McMillan, L.: A data-driven reflectance model. ACM Trans. Graph. **22**(3) (2003) 759–769
8. Kovar, L., Gleicher, M.: Simplicial families of drawings. In: UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology, New York, NY, USA, ACM Press (2001) 163–172
9. Pellacini, F., Ferwerda, J.A., Greenberg, D.P.: Toward a psychophysically-based light reflection model for image synthesis. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 55–64
10. Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., Shieber, S.:

Design galleries: a general approach to setting parameters for computer graphics and animation. In: SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (1997) 389–400