Generic interactive pixel-level image editing

Y. Liang[1] & Y. Gan[1] & M. Chen[1] & D. Gutierrez[2] & A. Muñoz[2]
[1] South China Agricultural University, College of Mathematics and Informatics.
[2] Universidad de Zaragoza, I3A

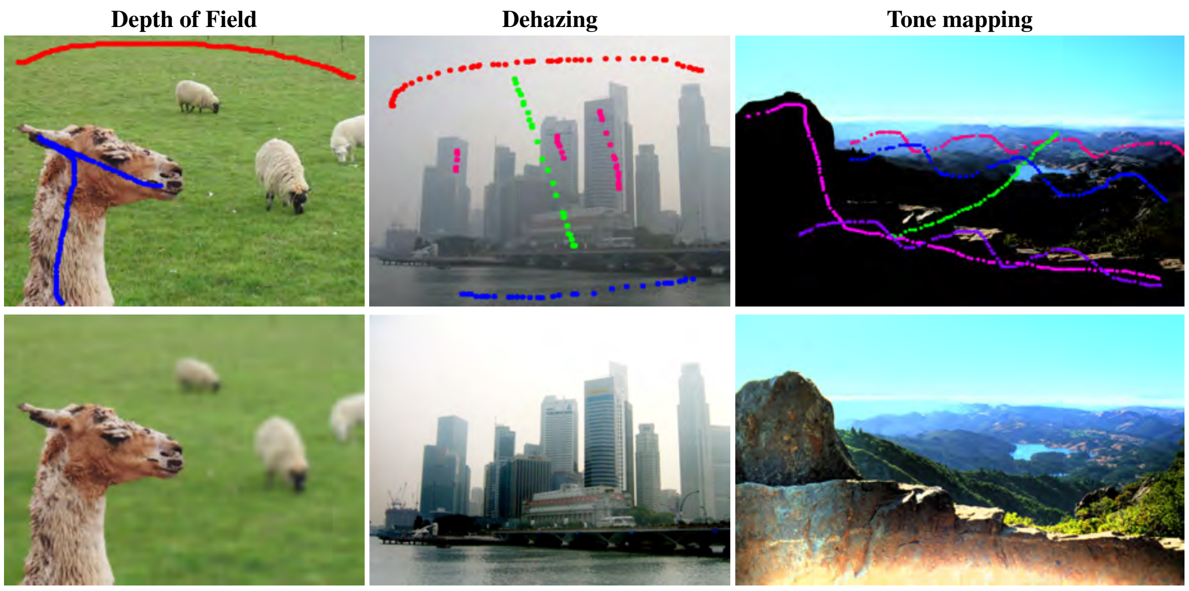

Figure 1: Generic image editing applications generated by our proposed paradigm. **Top:** The input images and users' scribbles (color lines on images). **Bottom:** The output results. **From left to right:** depth of field, dehazing, and tone mapping results. Our paradigm generates both the needed additional per-pixel values and the resulting edit at interactive rates, with minimal user input that can be iteratively refined.

Abstract
Several image editing methods have been proposed in the past decades, achieving brilliant results. The most sophisticated of them, however, require additional information per-pixel. For instance, dehazing requires a specific transmittance value per pixel, or depth of field blurring requires depth or disparity values per pixel. This additional per-pixel value is obtained either through elaborated heuristics or through additional control over the capture hardware, which is very often tailored for the specific editing application. In contrast, however, we propose a generic editing paradigm that can become the base of several different applications. This paradigm generates both the needed per-pixel values and the resulting edit at interactive rates, with minimal user input that can be iteratively refined. Our key insight for getting per-pixel values at such speed is to cluster them into superpixels, but, instead of a constant value per superpixel (which yields accuracy problems), we have a mathematical expression for pixel values at each superpixel: in our case, an order two multinomial per superpixel. This leads to a linear least-squares system, effectively enabling specific per-pixel values at fast speeds. We illustrate this approach in three applications: depth of field blurring (from depth values), dehazing (from transmittance values) and tone mapping (from brightness and contrast local values), and our approach proves both favorably interactive and accurate in all three. Our technique is also evaluated with a common dataset and compared favorably.

1. Introduction

While basic image editing applications, such as white balance, equalization or global tone mapping, require just pixel position and RGB color values as input, more sophisticated applications require additional per-pixel values. Examples of such additional per-pixel information are depth or geometry values for depth of field simulation [YG14], transmittance for dehazing [Fat14], or shading and reflectance intrinsic per-pixel values for relighting or retexturing [GMnLMG12]. Such additional per-pixel values describe properties of the scene and provide key cues for specific edits, leading to beautiful and interesting image transformations.

The generation of such additional values has drawn much attention in recent years. A trend of approaches are based on the control of the capture, either by capturing additional data during the capture process, such as gathering per-pixel depth values from sensor motion [YG14] or by using specific hardware, such as a plenoptic camera for estimating intrinsic shading-reflectance decompositions [GEZ*17]. Such techniques, however, are not general and applicable to arbitrary off-the-shelf images. Another way to obtain such values is elaborated heuristics and priors over the RGB color distribution in an image, such as relating color discontinuities with depth or reflectance discontinuities [LSH14, GMnLMG12], or particular color distributions for transmittance estimation in participating media [Fat14]. While results might be most of the times accurate, all these approaches fail when the heuristics and prior assumptions are not met.

This paper aims to obtain these per-pixel values from priors and user interaction. Previous approaches in this matter, however, are tailored for specific applications, such as intrinsic image decomposition [BPD09], intrinsic video decomposition [BST*14] or material editing [DRCP14]. As a previous approach [CMM18], our work is generic and agnostic to the underlying application. We illustrate our interaction paradigm (that involves heuristic priors and user brushes) with three applications as shown in Figure 1: depth-of-field, dehazing and tone mapping. However, the interactive image editing based on per-pixel value has to find a compromise among efficiency, expressiveness and accuracy. Present techniques that shine on one of the axes often need to sacrifice the other, such as efficiency [BPD09], expresiveness [IEK*14] or accuracy [CMM18]. Our technique overcomes such Pareto frontier by achieving a new improved compromise between the three of them.

Our key finding is that efficiency can be preserved by still using superpixels, but enabling different per-pixel values inside each superpixel. Thanks to this idea, our approach preserves efficiency independently of image resolution, while achieving greater accuracy by having a different value per pixel. Over this representation, we build an efficient linear least-squares system in which the restrictions come either from heuristics and priors (associating color continuity to label continuity, for instance) or from user scribbles (that set values or boundaries). We choose an order two multivariate polynomial distribution of per-pixel values inside each superpixel, because it is both simple (therefore most restrictions are easy to translate into linear equations) and differentiable, so restrictions not only involve values but also gradients (such as forcing planar surfaces through user scribbles).

This greatly improves over previous work [CMM18], in which we not only obtain per-pixel value smoothness and therefore fewer artifacts (see Figure 8 and Figure 9), but enable a gradient-domain global restriction and a gradient-based user brush (e.g. for planar surfaces). On top of that, we devise a new boundary brush that breaks global boundary restrictions and forces noticeable boundaries when applied. These improvements come without any substantial performance penalty (see Section 5.5 for details).

In summary, our main contributions are as follows:

1. A general pixel-level image editing paradigm is proposed based on four types of semantic brushes which do not require dense and accurate drawings. Our paradigm is not sensitive to user inputs and the editing result can be produced in real-time after user generically input some intuitive scribbles.
2. A new additional per-pixel value propagation scheme is constructed as a linear optimization system by defining an order two multinomial for each superpixel. It produces continuous real value on per-pixel very fast which makes our paradigm to achieve on-line interactive editing with accurate additional per-pixel value.
3. Three editing applications (additional per-pixel values) are provided including depth simulation (depth), dehazing (transmittance), and tone mapping (brightness, contrast). By the quantitative and qualitative evaluations on common database NYU V2 and present methods, our paradigm produces more favorable results.

2. Related work

Interactive image editing has drawn much attention for its outstanding and favorable outputs by introducing the meaningful guidance from users. Besides the direct usages in interactive cut-and-paste editing with cloning tools [ADA*04, PGB03] and compositing of multiple images [GJY*18], recently it has been widely used in many sophisticated image editing applications such as measured materials [AP08, DTPG11], intrinsic decomposition [BHY15, BPD09], depth of field [LSE17, KHC*18], color or face editing [MCY*13, PHS*18] or light field editing [JMB*14].

Some of the most sophiticated image editing appolications require additional per-pixel information (depth, reflectance, illumination). Different from previous work, which focuses on controlling the capture process [YG14, GEZ*17], this paper focuses on generating such additional per-pixel values from generic user input. Some methods have been proposed for propagating per-pixel values. Bousseau et al. [BPD09] propagate scribble constraints based on standard least-square to achieve intrinsic decomposition. Liao et al. [LSE17] use sparse user annotations as hard constraints to generate per-pixel depth by a constrained linear optimization. General propagation approaches include Markov Random Fields (MRF) [SZS*08] or Random Walker algorithms (RW) [VC17]. Such approaches, however, are computationally expensive, and therefore do not provide interactive feedback to the user.

To reduce computation time, some techniques optimize algorithms by using GPU [CXGS12], parallel processing [VN09] or a linear system [IEK*14, LFUS06]. However, the methods based on GPU and parallel process require parallel programming and graphics processing units. Some others improve efficiency by clustering the input, either based on a selection of columns of a matrix representation [AP08], a KD-Tree [XLJ*09] or superpixels [SYJ17, CMM18]. In fact, superpixels are widely used in applications such as depth estimation [JL15, LSH14], intrinsic image decomposition [GMnLMG12] and semantic or geometric labeling [WSL*18]. Other approaches propagate values combine MRF with a superpixel segmentation [MK10, CK14] and even introduce GPU [VN09] in RW to reduce time cost. Each form of clustering, however, presents some form accuracy limitations, being the most obvious having a single constant value per superpixel, and can lead to unnatural editing results. Our work overcomes this limitation while maintaining efficency by enabling different per-pixel values inside each superpixel.

Such as our approach, previous work obtain information from meaningful user scribbles with different semantics. For example, Levin et al. [DLW04] use user scribles for image and video colorization, while Bousseau et al. [BPD09] employ dense and careful scribbles for shading and reflectance restrictions in intrinsic image decomposition. Both approaches, however, are computationally expensive. Other approaches are more efficient, even interactive: slow motion photographs by providing motion direction [TPI10], recoloring by color brushes [CZZT12], editing the geometry of specific objects by scribbling structural lines [HZW*13] and inpaiting or retargeting by sketching geometrical restrictions [BSFG09].

The most efficient of scribble-based approaches (as our approach) are often based on solving a linear system. For example, Lischinski et al. [LFUS06] introduce sparse scribbles into the linear and edge-preserving energy system to adjust tonal value, Iizuka et al. [IEK*14] define a linear propagating method to compute per-superpixel depth values, Cambra et al. [CMM18] construct a linear system based on few scribbles that stablish restrictions to produce per-superpixel additional per-pixel values. Although these methods improve efficiency, they usually sacrifice accuracy because they can only represent a single value per-superpixel. On constrast, our approach enables multiple values per-superpixel interactively by distributing per-pixel values inside a superpixel according to an order two multivariate polynomial. This order two multinomial, additionaly, enables new semantic brushes that generate restrictions on the derivative (such as forcing planar surfaces when editing depth).

3. Overview

3.1. Problem statement

The goal of our interaction paradigm is to obtain a continuous additional value $L(x,y)$ for each pixel (x,y) of an image (continuous as a real number, discrete as it has a single value per pixel). The only inputs for that additional per-pixel value are the RGB color values at each pixel and scribbles from the user (several user strokes). The feedback should be interactive: after each user stroke, the algorithm should calculate a single value per pixel describing the specific property in a few seconds or less.

There is a compromise between accuracy and efficiency that must be overcome:

- We aim at getting the additional value per pixel but the time for calculating and obtaining such value is heavily dependant on image resolution: as image resolution increases, any per-pixel technique becomes non-interactive [LDCH17, EPF14, LFUS06].
- Clustering the image into superpixels makes the calculation time independent from image resolution, that now just depends on the number of superpixels. However, as each superpixel often has a single value, therefore hindering fine-grain accuracy and generating artificial discontinuities in propagating the additional per-pixel values [IEK*14, LSH14, CMM18].

Furthermore, the representation of the continuous additional value (per-pixel or per-superpixel) and the requirement of being interactive also influences (or even limits) the semantics of the brushes provided for the user strokes. For instance, a complex brush that requires a non-linear global optimization would require a much longer time than a simple brush that requires only a local optimization. On the other hand, we would like to empower the user to have brushes with high-level semantics for which each user stroke is intuitive and predictable.

Summarizing: obtaining a continuous additional per-pixel value $L(x,y)$ with user brushes that have powerful semantics forces time consuming global per-pixel optimizations. Our goal is to obtain such behavior at interactive rates without hindering the per-pixel effect nor the semantics of the user strokes.

3.2. Outline of the solution

Key idea. Our technique is based on the key insight that some forms of clustering (such as superpixels) are a must to keep time at interactive rates, so we focus on *making superpixels more meaningful*. We do that by extending the usual representation that each superpixel has a single additional value that applies to all pixels in the superpixel. We instead define a *mathematical expression* for each superpixel that gives specific per-pixel values. This enables us to keep value optimization time under control (by the number of superpixels) while having a specific value per-pixel.

Figure 2 utilizes the refocus application as an example of our paradigm to achieve generic interactive per-pixel image editing. By clustering the input into superpixels, our paradigm initializes the additional values for specific image property. Then, the user inputs some meaningful scribbles via simple brushes. By the mathematical expression of each superpixel, the additional values are propagated pixel by pixel and the image operator produces result quickly. Because the feedback result of scribbles happens at an interactive rate, the user repeatedly draws some additional scribbles based on feedback to optimize the editing as shown in the rectangle of Figure 2 until getting the ideal result.

Superpixel representation. Previous work that deals with propagating additional values of superpixels often gives a constant value for all the pixels inside the superpixel. Given a superpixel P they define the additional value for each pixel in it as:

$$L_P(x_p, y_p) = c_P \qquad \forall (x_p, y_p) \in P, \tag{1}$$

where c_P is the additional value to be optimized for superpixel P. This representation has some consequences such as artificial discontinuities at superpixel boundaries, and lack of accuracy and control. Our technique has represented each superpixel with a mathematical expression that eliminates artificial discontinuities, enhances accuracy and fine-grain controls of pixel values. Although our key idea could include any mathematical representation, we find that a good trade-off between efficiency and control is obtained

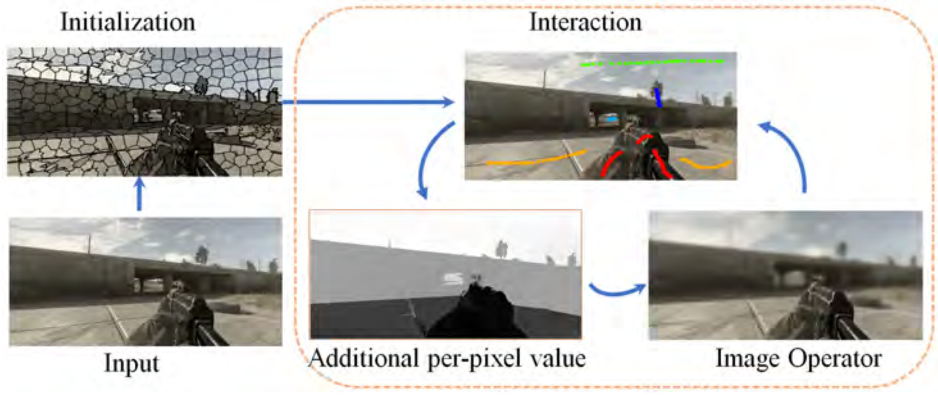

Figure 2: An example of our interactive process via Depth of Field. First, the paradigm is initialized by superpixel segmentation. Second, some meaningful scribbles are easily drawn. Third, additional per-pixel values (depth) are propagated among the image. Forth, an image operator produces a real-time result as the feedback of the user's inputs. The user repeats to draw scribbles and obtain feedback until getting an ideal result.

with order two multinomial:

$$L_P(x_p,y_p) = c_{P20}x_p^2 + c_{P11}x_p y_p + c_{P02}y_p^2 + c_{P10}x_p + c_{P01}y_p + c_{P00} \quad \forall (x_p,y_p) \in P \tag{2}$$

where the factors to optimize the per-pixel values for any superpixel P are the coefficients c_{P20}, c_{P11}, c_{P02}, c_{P10}, c_{P01} and c_{P00}.

Our multinomial representation is simple enough for an efficient additional per-pixel value optimization, while enables a variety of pixel values per superpixel. It furthermore enables finer control: for instance, there is the possibility of adding restrictions on the *gradient* inside each superpixel, from Equation (2):

$$\frac{\partial L_P}{\partial x_p}(x_p,y_p) = 2c_{P20}x_p + c_{P11}y_p + c_{P10} \tag{3}$$

$$\frac{\partial L_P}{\partial y_p}(x_p,y_p) = 2c_{P02}y_p + c_{P11}x_p + c_{P01} \tag{4}$$

This is not possible with constant superpixel values, as in Equation (1).

Order two multivariate polynomials are the optimal choice: they are convertible to linear equations at specific pixel positions and having analytical partial derivatives that are also linear (and not constant, as it would happen with order one multinomials). Higher-order multinomials would increase the number of unknowns of the solver which would affect both efficiency and numerical stability. Other mathematical formulations have been considered but rejected because they did not enable linear equations with linear derivatives and/or involved a greater number of unknowns.

Optimization. For the user interaction to be as fluid as possible, per-pixel value optimization must be as efficient as possible. As in the previous work [Gra06,CMM18], our solver is a linear system of equations. However, as opposed to previous work, our restrictions come from non-constant superpixel values and therefore can influence each pixel's value (or pixel-to-pixel value relations) coming from Equation (2) or affect each pixel's gradient (or pixel-to-pixel gradient relations) coming from Equations (3) and (4).

Both heuristic restrictions and the constraints coming from user strokes are expressed as a linear equation, then generating an over-constrained linear system $\mathbf{Ax} = \mathbf{B}$ where the unknowns in $\mathbf{x}$ are all the superpixel coefficients c, as in Equation (2), for all the superpixels. As it is (generally speaking) over-constrained, it becomes a linear least-squares minimization:

$$\min_{x \in R^m} ||\mathbf{B} - \mathbf{Ax}||_2 \tag{5}$$

which is done through the equivalent normal linear system:

$$\mathbf{A^T A x} = \mathbf{A^T B} \tag{6}$$

which being symmetric positive semi-definite is solved utilizing a LDLT (Cholesky-based) optimization, which is rather efficient and reasonably accurate.

We optimize the stability of the solver by working with normalized pixel positions: $x_p \in [0..1]$ and $y_p \in [0..1]$ in Equations (2), (3), and (4). Following this way, the structure of the linear system does not depend on image resolution.

Preconditioning. The solver of the above optimization is modeled as a linear least squares system as in Equation (5). Therefore, each equation j can be *weighted* by a preconditioning factor w_j that sets it's *importance* inside the whole system. For instance, the following standard equation:

$$\sum_{i=1}^{n} a_{ij}x_i = b_j \tag{7}$$

can be preconditioned as:

$$w_j \sum_{i=1}^{n} a_{ij}x_i = w_j b_j \tag{8}$$

In a standard linear system (not a linear least squares system), Equations (7) and (8) would be equivalent (except for $w_j = 0$) but in a linear least squares system w_j states the importance of the equation in the whole system. This preconditioning factor is applied locally for specific equations while there are global preconditioning values for parts of the system (see Section 4 for details).

Precomputation and interactive stages. For the whole process, the calculations are split into two stages, namely the precomputation and interactive stages.

In the *precomputation* stage, our technique:

- Generate the equations that set up global constraints in the image and form the initial least-square system.
- The precomputation stage is only done once per-image and takes around 3-4 seconds.

In the *interactive* stage, our technique:

- Using a specific brush with a specific meaning which can affect several pixels, the user draws one stroke in the image.
- Several equations (that correspond to the brush's meaning) are added into the linear least square system.
- The system is solved by the LDLT algorithm and the additional per-pixel values based on this stroke are generated.
- The additional per-pixel values are interpreted and the corresponding image operator is applied (depends on the specific editing application).

The interactive stage is based on user repeated interaction until the user gets the desired result. It takes less than one second, independently on image resolution. The tradeoff between efficiency and accuracy of this stage are determined by the number of superpixels. Figure 2 illustrates all the stages of this technique. The *precomputation* stage happens at the initialization step, which forms the initial least-square system based on the global constraints constructed among superpixels. The *interactive* stage forms a cycle process as shown in the rectangle of Figure 2. It firstly introduces meaningful scribbles, then quickly computes the additional per-pixel value, and finally employs image operator to produce a result. As this cycle process happens at interactive rate, user can adaptively introduce more meaningful scribbles to optimize the result based on the feedback.

Section 4 describes all the specific global constraints (for the precomputation stage) and all the specific brushes (for the interactive stage) and describes how each of them is translated into equations for the linear least square system.

4. Constraints

The constraints of our paradigm system must be expressed in the form of a linear equation (as discussed in the previous section). They come from two main sources: global precomputed constraints and user strokes made with different brushes.

4.1. Global Precomputed Constraints

The global precomputed constraints are based on reasonable heuristics that come from the RGB values in the image and the distribution of pixels among superpixels. The intuition for these is that the areas belonging to the same object are more likely to share both similar RGB values and similar per-pixel values. This is partially addressed by the superpixel segmentation, but our technique includes some constraints that link color similarity to per-pixel value similarity at the boundaries between superpixels to ensure the continuity between superpixels. In this paper, we define the global precomputed constraints by defining the following value continuity and gradient continuity of our additional per-pixel value.

Value continuity. For each pair of 8-neighbor pixels belonging to different superpixels $(x_p, y_p) \in P$ and $(x_q, y_q) \in Q$ we add the following two equations:

$$\begin{aligned} c_{PQ}L_P(x_p,y_p) &= c_{PQ}L_Q(x_p,y_p) \\ c_{PQ}L_P(x_q,y_q) &= c_{PQ}L_Q(x_q,y_q) \end{aligned} \tag{9}$$

which basically enforces continuity on propagating additional per-pixel values across superpixel boundaries. The value c_{PQ} *preconditions* both equations (see Section 3.2), and gives more importance in the overall system to those equations in which colors are more similar. It is calculated as follows:

$$c_{PQ} = e^{-k||I(x_p,y_p)-I(x_q,y_q)||} \tag{10}$$

where $I(x,y)$ represents the color of the image at pixel (x,y) in CIE-Lab color space (L-2 norm CIE-Lab difference) and k is a user-defined positive parameter which is experimentally set as $k = 1$. Note that Equation (10) establishes the importance of the per-pixel value continuity for Equation (9). The value of Equation (10) is much higher when the color is very similar between superpixels than when the color becomes too different (even though it theoretically never reaches 0 importance).

Gradient continuity. In a similar spirit, for each pair of 8-neighbor pixels belonging to different superpixels $(x_p, y_p) \in P$ and $(x_q, y_q) \in Q$, we add the following four equations that connect not only value continuity of additional per-pixel value but its gradient continuity too:

$$\begin{aligned} c_{PQ}\frac{\partial L_P}{\partial x_p}(x_p,y_p) &= c_{PQ}\frac{\partial L_Q}{\partial x_q}(x_p,y_p) \\ c_{PQ}\frac{\partial L_P}{\partial y_p}(x_p,y_p) &= c_{PQ}\frac{\partial L_Q}{\partial y_q}(x_p,y_p) \\ c_{PQ}\frac{\partial L_P}{\partial x_p}(x_q,y_q) &= c_{PQ}\frac{\partial L_Q}{\partial x_q}(x_q,y_q) \\ c_{PQ}\frac{\partial L_P}{\partial y_p}(x_q,y_q) &= c_{PQ}\frac{\partial L_Q}{\partial y_q}(x_q,y_q) \end{aligned} \tag{11}$$

where c_{PQ} also follows Equation (10) but we experimentally set $k = 0.02$. Different strategies with varying preconditioning factors were tested, but none of them proved significantly better than this simpler option. Generally speaking, the importance value c_{PQ} becomes relevant when including other restrictions coming from user strokes (see below in the text) where the solver will first break equations with low c_{PQ} values to satisfy other constraints coming from different sources.

Initialization. All global constraints establish relations between superpixels that lead to a linear system that is unsolvable (it has a trivial solution of setting all superpixel parameters to 0). In order to avoid a system crash, we set an arbitrary initialization per-pixel value of 0.5 in the upper left pixel that is removed after the first user interaction. This ensures an initial system convergence.

4.2. User brushes

The above global constraints set up a basis from which each user stroke can be efficiently propagated throughout the whole image. We provide several intuitive brushes that are also mathematically

Table 1: Color code used for the applied user strokes that indicates the kind of brush and its specific set properties.

Stroke color	Brush
🟦🟪🟥	**Seed value brush.** Value from 0 (blue) to 1 (red)
🟨🟨🟧	**Value similarity brush.** Different colors indicate different pixel connections.
🟩🟩🟨	**Gradient similarity brush.** Different colors indicate different pixel connections.
🟦🟦🟦	**Boundary brush.** Distance from 0 (no edge) to 1 (strongest edge).

expressed as linear equations and be seamlessly integrated with the rest of the linear least-squares system.

For the sake of clarity, we illustrate the user strokes by means of a coherent color code, described in Table 1, so the user (and the reader) is able to intuitively understand the user interaction. All figures (such as Figure 3) that illustrate our user interaction in this paper strictly follow such color code.

According to the meanings of the strokes, all the brushes for user strokes are classified into four kinds including the seed value brush, value similarity brush, gradient similarity brush and boundary brush. Each of the user strokes generates a set of equations that depend on the specifically chosen brush, as follows:

Seed value brush. Each pixel $(x_p, y_p) \in P$ (where P is a superpixel) affected by a user stroke with this seed value brush will be set to a specific constant value v_b:

$$L_P(x_p, y_p) = v_b \tag{12}$$

The user interface provides several seed value brushes, each of them with a different specific value v_b. Instead of specific values, the user only sees intuitive descriptions of such values (for instance "near", "far", "background" at propagating the per-pixel depth value). The effect of several seed value brushes is illustrated in the top row of Figure 3. Several brushes might be affecting the same pixel, theoretically contradicting each other. However, being a linear least-squares system the solver will average the value that mostly fits all the involved equations.

Value similarity brush. This brush does not set specific per-pixel values. Instead, it enforces that each pair of pixels $(x_p, y_p) \in P$ and $(x_q, y_q) \in Q$ affected by a user stroke has the same per-pixel value (P and Q are two superpixels, either different or the same). Specifically:

$$L_P(x_p, y_p) = L_Q(x_q, y_q) \tag{13}$$

There are several value similarity brushes, each of them establishing similarity restrictions to all pixels affected by the brush. This can be used either for a single user stroke (connecting the additional per-pixel values inside that user stroke, as in the second row of Figure 3) or with multiple user strokes, in which all the pixels affected by all strokes are connected (see Figure 3, third row). Only a few previous works enable the user to establish such per-pixel value connections at different image locations [CMM18], which performs very efficiently in producing continuous and accurate additional per-pixel values.

Gradient similarity brush. Each pair of pixels $(x_p, y_p) \in P$ and $(x_q, y_q) \in Q$ affected by a user stroke using this brush will tend to have the same gradient, which is modeled by the following equations:

$$\frac{\partial L_P}{\partial x_p}(x_p, y_p) = \frac{\partial L_Q}{\partial x_q}(x_q, y_q); \frac{\partial L_P}{\partial y_p}(x_p, y_p) = \frac{\partial L_Q}{\partial y_q}(x_q, y_q) \tag{14}$$

This brush works in the same way with the value similarity brush. Again, there are several gradient similarity brushes available to the user, each of them adding a different set of restrictions. Intuitively, this kind of brush helps to identify planar surfaces for propagating additional per-pixel values (see Figure 3, fourth row). This brush provides the gradient-based constraints which have never been possible in superpixel-based techniques before.

Boundary brush. In some occasions, the effect of the value / gradient continuity constraints (described in Section 4.1) does not prioritize specific boundaries (especially, when the color at both borders of the boundary is too similar). Therefore, we define the boundary brush to deal with the additional per-pixel values on boundary locations. This brush enforces the existence of a boundary in two ways: first, it eliminates all the equations that stablish some connectivity in the boundary by adding the same Equations (9) and (11) but with negative $-c_{PQ}$ to all the border pixels affected by the boundary brush. Additionally, for each pair of superpixel border pixels $(x_p, y_p) \in P$ and $(x_q, y_q) \in Q$ ($P \neq Q$) affected by the boundary brush we add the equation

$$L_P(x_p, y_p) = L_Q(x_q, y_q) + d_b \tag{15}$$

where d_b is a per-brush specific parameter (there can be several boundary brushes with different parameters) which establishes the *strength* of the boundary. The P and Q superpixels are selected according to the direction of the stroke: if done counterclockwise, the inner superpixel is Q and the outer superpixel is P. In practice, this equation (and the elimination of the global constraints affecting both pixels) enhances the boundary (see Figure 3, bottom row). This brush just affects the corresponding equations at superpixel boundaries. As future work, it would be interesting that this brush would also create superpixel boundaries dynamically.

5. Experiments

This section describes the evaluations and comparisons of our generic pixel-level image editing paradigm. We first verify the accuracy of our depth estimation on the famous **NYU V_2 dataset** [NSF12] in Section 5.1. We then explore the sensitivity of our approach in Section 5.2. Then, we describe the applications of our paradigm in Section 5.3. Specially, we illuminate our outstanding contribution about smoothing the additional per-pixel values on Section 5.4. Finally, we analyze the efficiency of our paradigm on Section 5.5. Unless otherwise stated, all the results are generated by a PC with i7-6700 CPU at interactive rates (383 superpixels, <1s for each interaction), without GPU.

5.1. Accuracy

We test the accuracy of our approach with the **NYU V_2 dataset**, which provides RGB images with ground truth depth (captured us-

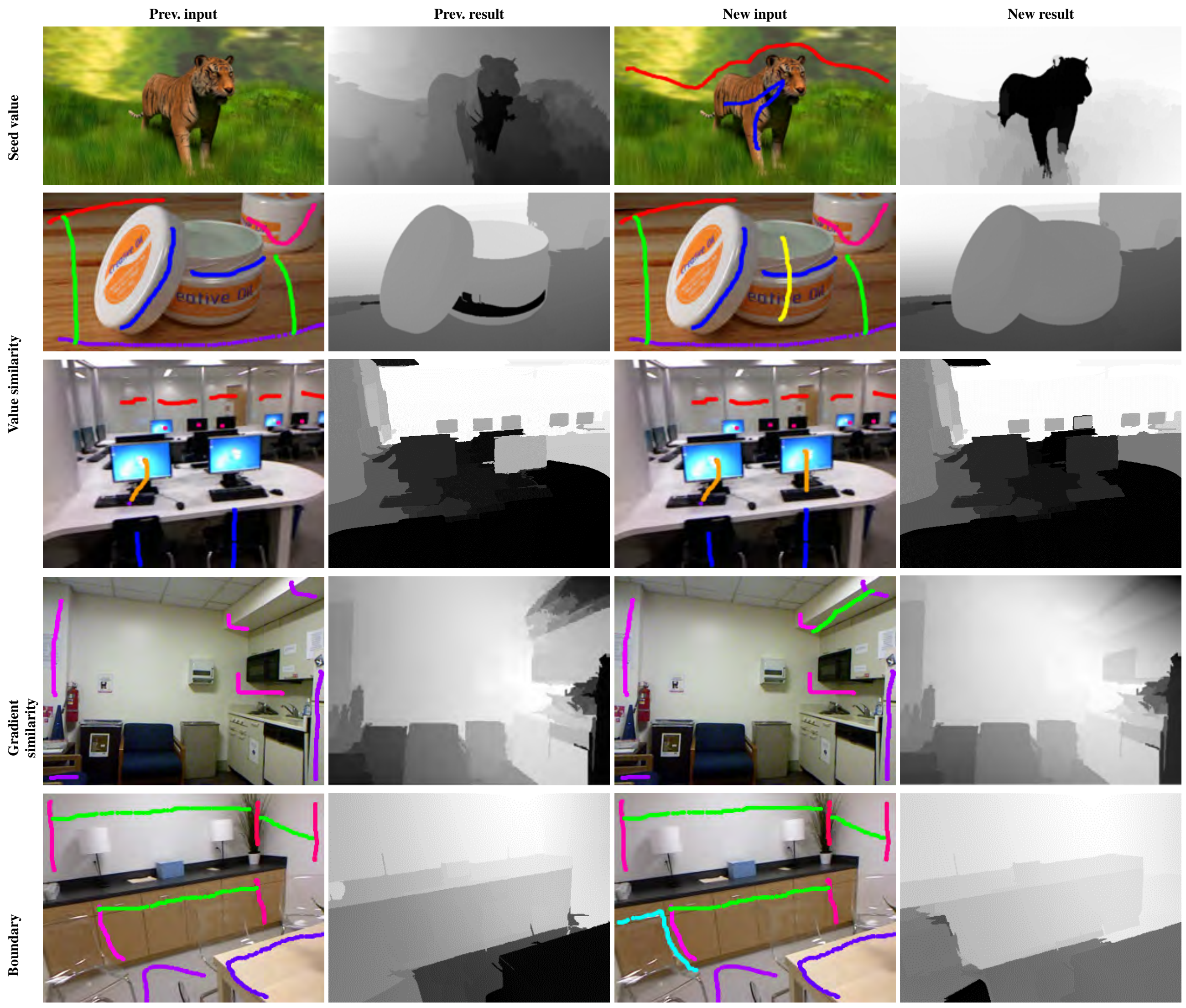

Figure 3: Individual effect of each brush with the example of depth editing. **From left to right:** input before the edit, depth map before the edit, new user stroke with the specific brushes, depth map after the edit. **From top to bottom:** seed value brushes (two strokes with different values), value similarity brush (single user stroke), value similarity brush (two separate but connected user strokes), gradient similarity brush (user stroke forces a planar surface), boundary brush (user stroke forces depth difference around the chair). Note that on each case (row) the effect of each user stroke is both meaningful and practicable.

ing a Microsoft Kinect camera) and has been often used for validating depth estimation [LSH14, EPF14, LDCH17]. Some of the images of the dataset do not have depth values for every single pixel, due to regions out of the sensor's capturing range, and therefore are not suitable for our approach. We ignore those and select a subset of 30 representative images. With few user strokes of our interaction paradigm, we generate a depth map for an RGB image of the dataset and we compare it with the ground truth (numerically and visually).

Figure 4 illustrates some of our results compared to previous automatic work [LSH14] and the ground truth from the NYU V_2 dataset (the rest of the results from this dataset are available as supplementary material). As our seed value brushes (that set specific depth values in this case) are subjectively specified ("near", "far") for this experiment, we configure them to yield specific fixed values (objective as opposed to subjective) in order to be able to test its accuracy numerically. Still, the number of user strokes for each of the results is rather low (between four and ten for all the results, see Figure 4, rightmost column). We obtain an average root mean square error of 0.439 for all the 30 images with an average of 10 user scribbles per image, which is a reasonably low number given the accuracy of the results, and a good metric of the expressiveness of our approach. To put the error metric in perspective, the most ac-

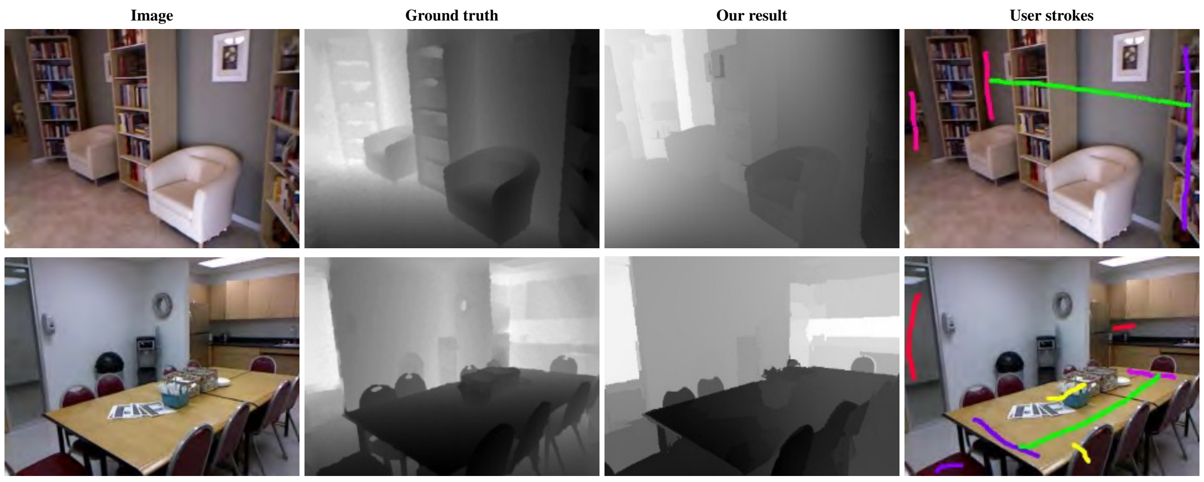

Figure 4: Each row presents different our depth map estimation results compared to the NYU V_2 ground truth. The color at user stroke (rightmost column) is described in Table 1.

curate automatic previous work reports a 0.641 average root mean square error for this dataset [EF15]. The whole accuracy analysis with several error metrics can be found as supplementary material.

Furthermore, we analyze the progression of the error with respect to the number of user strokes (see Figure 5). Note that, as expected, the error decreases as the user refines the input (number of user strokes).

5.2. Sensitivity

One of the main concerns regarding user interaction is its sensitivity, the variation of output with respect to user input. Each interaction paradigm must find the appropriate compromise between predictability (the technique is sensitive enough to the intentions on the user) and stability (the technique does not yield extreme variations on the output from minimal user input). In the case of our proposed interaction paradigm, this is partially explored in Figure 3, in which each individual brush is explored independently.

Furthermore, Figure 6 illustrates such sensitivity by applying different sets of user strokes to the same input, and compare the outputs. It can be seen that similar input yields similar output and that our approach is stable to small variations of the input user strokes. This stability comes from the balance between the global constraints (Section 4.1) and the local user scribbles (Section 4.2).

5.3. Applications

As previous work [CMM18], we illustrate the versatility of our paradigm through three different applications (illustrated in Figure 7):

Depth of field. A *depth map* is generated using our interaction paradigm. Following the previous work in [LSP14], the depth of field effect is obtained by applying a set of convolutions (blurs) of different radius according to the difference between depth at each pixel and focal distance. The user can configure both focal distance (by clicking on the specific focus point on the scene) and the maximum radius (related to the emulated aperture).

Dehazing. A *transmittance map* is generated using our interaction paradigm. The color of maximum pixels of the transmittance map indicates the color of the "haze", and from that transmittance map, the dehazed image is obtained using the standard dehazing formulation as proposed in [Fat14].

Tone mapping. We start from a global tone mapping curve [MS08] that transforms High Dynamic Range images into Low Dynamic Range images through two global parameters: brightness and contrast. Our application modulates both global parameters through our user interaction paradigm into a per-pixel *brightness map* and *contrast map*.

Figure 7 compares each of these applications with the automatic (as in automatically generated maps) state of the art, plus with previous superpixel-based editing work [CMM18]. It shows how our work, with few user strokes, enables results that are visually in comparison with the state of the art.

5.4. Superpixels

One of the key contributions of our work is to achieve per-pixel quality maps with superpixel efficiency. In previous work, each superpixel represents a single constant additional value for all the pixels within the superpixel [LSH14, CMM18] as described in Equation (1), while our work has a multinomial superpixel representation (as described in Equation (2)). This difference makes our method overcome the discontinuity of additional per-pixel value among superpixels that are common for the other present methods.

Figure 8 demonstrates the depth map obtained with two kinds of superpixel techniques: one of them(shown in the fourth column) uses learning and a discrete optimization [LSH14], the other (shown in the second column) is based on user editing [CMM18] as our work. Both of the resulting depth maps present heavy discontinuities at superpixel boundaries, while our technique is able to

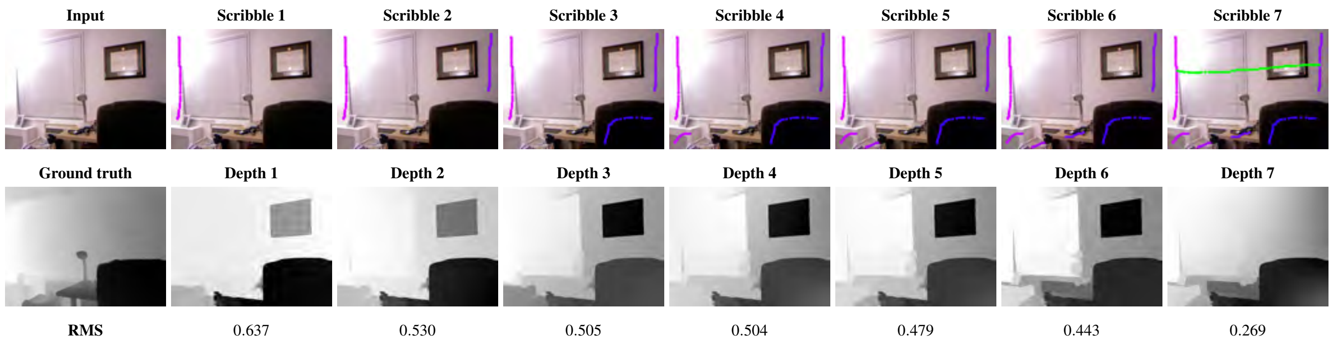

Figure 5: Accuracy progression with respect to the number of user strokes. **Left column :** Input and ground truth. **Rest of columns, from left to right :** Accuracy progression. Top row: user strokes. Middle row: output depth map. Bottom row: root mean squared error of the output with respect to the ground truth. Note that each user stroke improves accuracy.

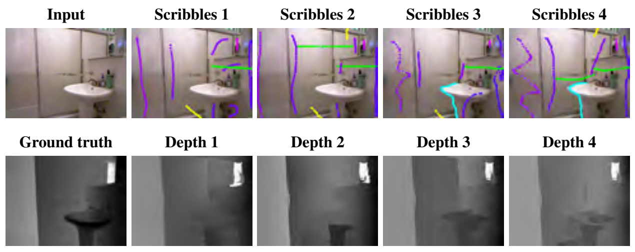

Figure 6: Analysis of scribble sensitivity. For the same scene, different scribble inputs are performed and different (but similar) depth maps are obtained. **Left column, top:** Input image. **Left column, bottom:** Ground truth depth. **Rest of the columns, top row:** Different sets of user scribbles as input. **Rest of the columns, bottom row:** Obtained depth maps corresponding to each set of user scribbles.

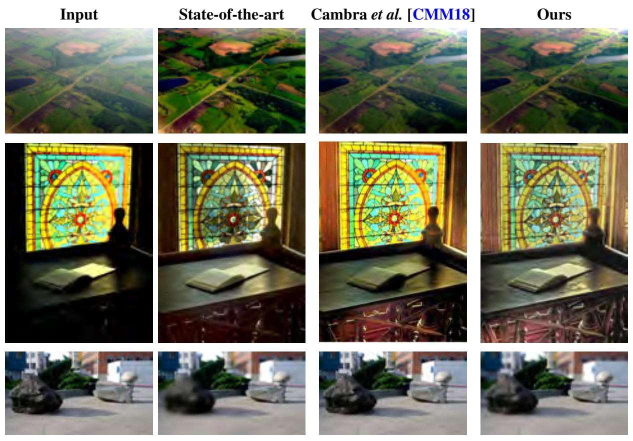

Figure 7: Comparison between our results and the state-of-the-art algorithm for different applications. **From left to right:** Input image; results obtained from different state-of-the-art algorithm; results from previous editing technique [CMM18]; our results; **From top to bottom:** Applications with different state-of-the-art algorithms: dehazing [Fat14], tone mapping [LFUS06]; depth of field [YG14].

overcome them thanks to the multinomial superpixel representation and the value and gradient continuity constraints (see Section 4.1).

Such discontinuities affect the outcome (additional per-pixel value) of the editing and be noticeable in the final result. This is illustrated in Figure 9, where the user applies similar user strokes using the method of Cambra et al. [CMM18] and ours. For the three applications, this work yields smooth and clear results, while the discontinuities in previous work lead to the unpleasant visual appearance (see Figure 9).

5.5. Efficiency

Previous work on superpixel based image editing has proven to work at interactive rates [CMM18] and also solves the propagation by means of a linear system of equations. Compared to it, our approach presents six unknowns per superpixel (multinomial coefficients) while their approach requires only one: the constant value. When using a similar number of superpixels in our experiments (around 700 superpixels for the NYU V_2 dataset, see Figure 4), their approach requires 0.64 seconds on average for each user stroke while our approach requires around 1.56 seconds on average. Although still at interactive rates, our approach seems slower but this is expected: as the number of unknowns increases, solving the linear system and propagating the additional per-pixel values requires more time. However, when given the more expressive nature of our superpixel formulation, we are able to obtain still more accurate results with around 300 superpixels, which only require 0.27 seconds per interaction.

We demonstrate the efficiency performance of our method and the present interactive editing method [CMM18] by Figure 10. Comparing the top two rows and the middle two rows of Figure 10, we know that with the similar superpixel number (383 superpixels for ours, 376 superpixels for theirs), our method performs slightly slower, but producing more accurate results. That is because we use an order two multinomial to represent the values for the pixels in-

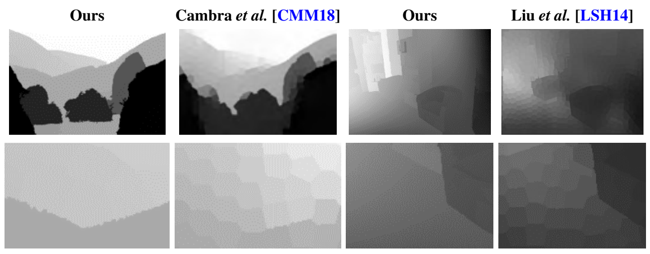

Figure 8: Comparison between our depth propagating and other superpixel based state-of-the-art depth computing techniques. **Top row:** Full size of propagated additional per-pixel values. **Bottom row:** Insets. Our approach enables smooth and continuous per-pixel value, while other approaches produce artificial discontinuities especially among superpixels.

side a superpixel, while they defined constant values. However, by comparing the top two rows and the bottom two rows of [CMM18], it is clear that when we reduce the superpixel number (154 superpixels for ours, 376 superpixels for theirs), our method uses much less time for each interactive while still producing more accurate results. The efficiency and interactivity can be further analyzed in the supplemental video.

Not only we are faster in terms of computational speed, but also our proposed workflow includes more intuitive and semantic brushes, including gradient similarity and edge brushes, which in turn enable a more efficient interaction paradigm in terms of user time. Our tests show that several scenes require fewer user strokes and therefore less user time compared to previous work with fewer brush options [CMM18]. As future work, it would be interesting to perform a more formal user study that statistically validates this claim.

6. Conclusion and future work

This paper proposes a general image editing paradigm, which generates a set of continuous additional per-pixel values from a set of meaningful user strokes. The advantages of our approach are that the paradigm is very intuitive to use, and that is interactive even when including many user scribbles for complex brushes. Therefore, by some easy scribbles, users can iteratively refine the results thanks to the fast feedback.

This approach is based on a new propagation scheme which spreads the additional per-pixel values of an image at interactive speeds (less than a second independently of image resolution). The propagation is a linear optimization system and constructed on top of a superpixel representation that instead of a single value includes an order two multinomial (for each superpixel). Compared with other approaches (that only give a single value for the whole superpixel) our paradigm is able to overcome most of the limitations and artifacts than such approaches yield, and is, therefore, both more accurate and enables higher meaning strokes. Several kinds of constraints are defined for the propagation to make it more robust and flexible including the constraints from per-pixel value continuity in color and gradient fields and the constraints from the meaning inputs of users.

The limitations in our approach come from the static superpixel segmentation: some times a poor segmentation might lead to sub-optimal results and sloppy user interaction in specific regions of the image. Still, the artifacts given by our application are meaningless compared to other superpixel-based approaches. In the future, we would like to provide a *dynamic* segmentation, so that the boundary brush would not only generate new equations but also modify the superpixel structure.

We illustrate the advantages of our paradigm through three applications: depth of field blurring, dehazing and tone mapping. Each of such applications requires different additional per-pixel values (depth, transmittance and constant and brightness) and our paradigm works for all three cases. Our paradigm produces in all the three applications results that are on-par with the state of the art, with the additional advantage that they can be interactively refined by the user. To illustrate this, we provide a supplemental video and the source code of the applications[†]. As further research, it would be exciting to explore other more advanced applications such as intrinsic image decomposition, and also evaluate the interaction paradigm more formally through a user study. We expect that this paradigm will enable more generic and sophisticated editing applications in the future.

Acknowledgments. This work is supported by the National Science Fund of China (61772209), the Science and Technology Planning Project of Guangdong Province (2016A050502050), the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (project CHAMELEON, grant No 682080) and the Spanish Ministry of Economy and Competitiveness (project TIN2016-78753-P).

References

[ADA*04] Agarwala A., Dontcheva M., Agrawala M., Drucker S., Colburn A., Curless B., Salesin D., Cohen M.: Interactive digital photomontage. *ACM Trans. Graph 23* (2004), 294–302. 2

[AP08] An X., Pellacini F.: Appprop: all-pairs appearance-space edit propagation. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 40. 2

[BHY15] Bi S., Han X., Yu Y.: An l1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Trans. Graph. 34*, 4 (July 2015), 78:1–78:12. 2

[BPD09] Bousseau A., Paris S., Durand F.: User assisted intrinsic images. *ACM Transactions on Graphics (SIGGRAPH Asia 2009) 28*, 5 (2009). 2, 3

[BSFG09] Barnes C., Shechtman E., Finkelstein A., Goldman D. B.: Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG) 28*, 3 (2009), 24. 3

[BST*14] Bonneel N., Sunkavalli K., Tompkin J., Sun D., Paris S., Pfister H.: Interactive intrinsic video editing. *ACM Transactions on Graphics 33*, 6 (2014), 197. 2

† https://github.com/scut-mingqinchen/generic_pixel-level_editing

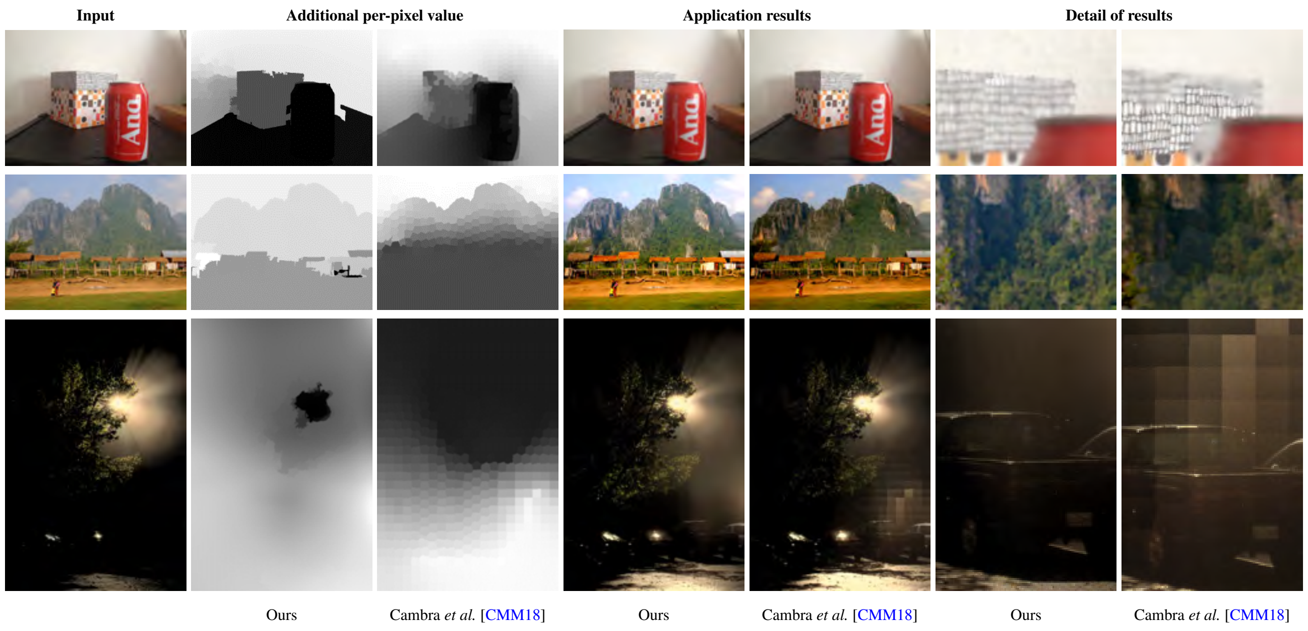

Figure 9: Comparison between our results and the results of Cambra et al [CMM18]. Our method overcomes the artifacts which are common for their method due to the discontinuity of their per-pixel values between superpixels. **From top to bottom:** Depth of field, dehazing and tone mapping.

[CK14] Chen Q., Koltun V.: Fast mrf optimization with application to depth reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 3914–3921. 3

[CMM18] Cambra A. B., Murillo A. C., Muñoz A.: A generic tool for interactive complex image editing. *The Visual Computer 34*, 11 (Nov 2018), 1493–1505. URL: https://doi.org/10.1007/s00371-017-1422-5, doi:10.1007/s00371-017-1422-5. 2, 3, 4, 6, 8, 9, 10, 11, 12

[CXGS12] Collins M. D., Xu J., Grady L., Singh V.: Random walks based multi-image segmentation: Quasiconvexity results and gpu-based solutions. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1656–1663. 2

[CZZT12] Chen X., Zou D., Zhao Q., Tan P.: Manifold preserving edit propagation. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 132. 3

[DLW04] Dani A. L., Lischinski D., Weiss Y.: Colorization using optimization. *ACM Transactions on Graphics 23* (2004), 689–694. 3

[DRCP14] Di Renzo F., Calabrese C., Pellacini F.: Appim: linear spaces for image-based appearance editing. *ACM Transactions on Graphics 33*, 6 (2014), 194. 2

[DTPG11] Dong Y., Tong X., Pellacini F., Guo B.: Appgen: interactive material modeling from a single image. *ACM Trans. Graph. 30*, 6 (2011), 146. 2

[EF15] Eigen D., Fergus R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Proceedings of the IEEE International Conference on Computer Vision* (2015), 2650–2658. 8

[EPF14] Eigen D., Puhrsch C., Fergus R.: Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems* (2014), pp. 2366–2374. 3, 7

[Fat14] Fattal R.: Dehazing using color-lines. *ACM Transaction on Graphics 34*, 13 (2014). 2, 8, 9

[GEZ*17] Garces E., Echevarria J. I., Zhang W., Wu H., Zhou K., Gutierrez D.: Intrinsic light field images. *Computer Graphics Forum 36*, 8 (2017), 589–599. 2

[GJY*18] Ge S., Jin X., Ye Q., Luo Z., Li Q.: Image editing by object-aware optimal boundary searching and mixed-domain composition. *Computational Visual Media 4*, 1 (Mar 2018), 71–82. URL: https://doi.org/10.1007/s41095-017-0102-8, doi:10.1007/s41095-017-0102-8. 2

[GMnLMG12] Garces E., Muñoz A., Lopez-Moreno J., Gutierrez D.: Intrinsic images by clustering. *Computer Graphics Forum 31*, 4 (2012). 2, 3

[Gra06] Grady L.: Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (2006), 1768–1783. 4

[HZW*13] Hu S.-M., Zhang F.-L., Wang M., Martin R. R., Wang J.: Patchnet: a patch-based image representation for interactive library-driven image editing. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 196. 3

[IEK*14] Iizuka S., Endo Y., Kanamori Y., Mitani J., Fukui Y.: Efficient depth propagation for constructing a layered depth image from a single image. In *Computer Graphics Forum* (2014), pp. 279–288. 2, 3

[JL15] Jin F., Li X.: A dense depth estimation method using superpixels. In *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)* (Dec 2015), pp. 290–294. 3

[JMB*14] Jarabo A., Masia B., Bousseau A., Pellacini F., Gutierrez D.: How do people edit light fields? *ACM Transactions on Graphics (SIGGRAPH 2014) 33*, 4 (2014). 2

[KHC*18] Knorr S., Hudon M., Cabrera J., Sikora T., Smolic A.: Deepstereobrush: Interactive depth map creation. In *International Conference on 3D Immersion, IC3D 2018, Brussels, Belgium, December 5-6, 2018* (2018), pp. 1–8. 2

[LDCH17] Li B., Dai Y., Chen H., He M.: Single image depth estimation by dilated deep residual convolutional neural network and soft-weight-sum inference. *arXiv preprint arXiv:1705.00534* (2017). 3, 7

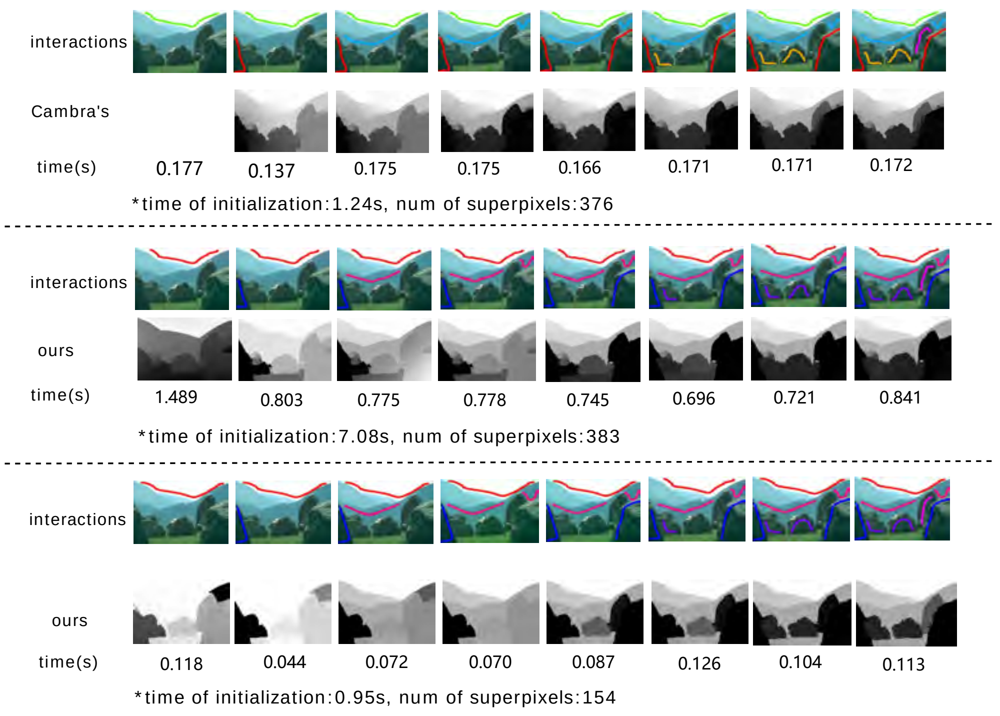

Figure 10: The efficiency comparison between our method and the interactive method proposed in [CMM18]. Using a similar superpixels number, our method produces better depth map while it performs little slower. However, after reducing the superpixels number (from 383 to 154), our method not only produces a better work but also uses less time.

[LFUS06] Lischinski D., Farbman Z., Uyttendaele M., Szeliski R.: Interactive local adjustment of tonal values. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 646–653. 2, 3, 9

[LSE17] Liao J., Shen S., Eisemann E.: Depth map design and depth-based effects with a single image. In *Graphics Interface* (2017). 2

[LSH14] Liu M., Salzmann M., He X.: Discrete-continuous depth estimation from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition* (2014), IEEE, pp. 716–723. 2, 3, 7, 8, 10

[LSP14] Ladický L., Shi J., Pollefeys M.: Pulling things out of perspective. In *IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 89–96. 8

[MCY*13] Musialski P., Cui M., Ye J., Razdan A., Wonka P.: A framework for interactive image color editing. *The Visual Computer 29*, 11 (2013), 1173–1186. 2

[MK10] Mičušík B., Košecká J.: Multi-view superpixel stereo in urban environments. *Int. J. Comput. Vision 89*, 1 (Aug. 2010), 106–119. 3

[MS08] Mantiuk R., Seidel H.-P.: Modeling a generic tone-mapping operator. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 699–708. 8

[NSF12] Nathan Silberman Derek Hoiem P. K., Fergus R.: Indoor segmentation and support inference from rgbd images. In *ECCV* (2012). 6

[PGB03] Pérez P., Gangnet M., Blake A.: Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH'03) 22*, 3 (2003), 313–318. 2

[PHS*18] Portenier T., Hu Q., Szabó A., Bigdeli S. A., Favaro P., Zwicker M.: Faceshop: deep sketch-based face image editing. *ACM Trans. Graph. 37*, 4 (2018), 99:1–99:13. 2

[SYJ17] Somanath G., Yao J., Jiang Y.: A novel framework for fast mrf optimization. *Electronic Imaging 2017*, 15 (2017), 26–31. 2

[SZS*08] Szeliski R., Zabih R., Scharstein D., Veksler O., Kolmogorov V., Agarwala A., Tappen M., Rother C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 6 (June 2008), 1068–1080. 2

[TPI10] Teramoto O., Park I. K., Igarashi T.: Interactive motion photography from a single image. *Visual Computer 26*, 11 (2010), 1339–1348. 3

[VC17] Vernaza P., Chandraker M.: Learning random-walk label propagation for weakly-supervised semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), vol. 3, p. 3. 2

[VN09] Vineet V., Narayanan P.: Solving multilabel mrfs using incremental α-expansion on the gpus. In *Asian conference on computer vision* (2009), Springer, pp. 633–643. 2, 3

[WSL*18] Wang J., Shi H., Liu M., Xiong W., Cheng K., Jiang Y.: Image semantic segmentation algorithm based on self-learning superpixel feature extraction. In *Advances in Internet, Data & Web Technologies* (Cham, 2018), Barolli L., Xhafa F., Javaid N., Spaho E., Kolici V., (Eds.), Springer International Publishing, pp. 773–781. 3

[XLJ*09] Xu K., Li Y., Ju T., Hu S.-M., Liu T.-Q.: Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 118:1–118:6. 2

[YG14] Yu F., Gallup D.: 3d reconstruction from accidental motion. In *IEEE Conf. on Computer Vision and Pattern Recognition* (2014), pp. 3986–3993. 2, 9