Decomposing Time-Lapse Paintings into Layers

Jianchao Tan* George Mason University
Marek Dvorožňák CTU in Prague, FEE
Daniel Sýkora CTU in Prague, FEE
Yotam Gingold George Mason University

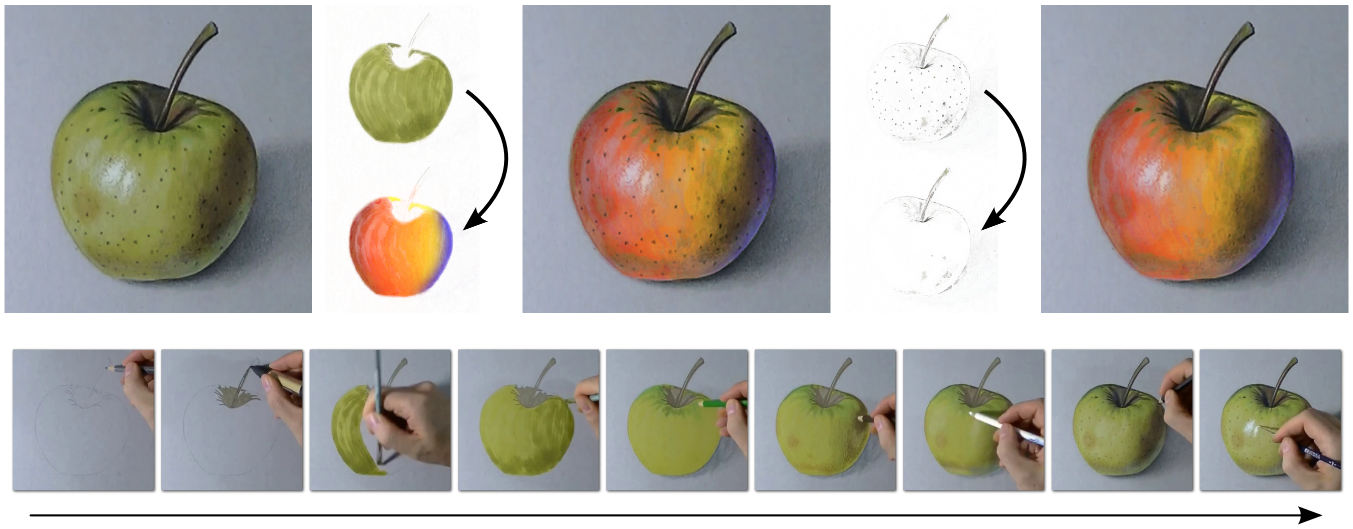


Figure 1: *Our approach enables rich history-based editing operations on physical paintings. From a time lapse recording of the painting process (bottom) we extract translucent paint layers into a temporal creation history. This allows artists to perform otherwise impossible spatio-temporal selections & edits which leverage temporal information to produce complex effects such as color gradients controlled by time (top left) or temporal eraser (top right). (Paint layers in this example use Porter-Duff "over" blending.) Time lapse video © Marcello Barenghi.*

Abstract

The creation of a painting, in the physical world or digitally, is a process that occurs over time. Later strokes cover earlier strokes, and strokes painted at a similar time are likely to be part of the same object. In the final painting, this temporal history is lost, and a static arrangement of color is all that remains. The rich literature for interacting with image editing history cannot be used. To enable these interactions, we present a set of techniques to decompose a time lapse video of a painting (defined generally to include pencils, markers, etc.) into a sequence of translucent "stroke" images. We present translucency-maximizing solutions for recovering physical (Kubelka and Munk layering) or digital (Porter and Duff "over" blending operation) paint parameters from before/after image pairs. We also present a pipeline for processing real-world videos of paintings capable of handling long-term occlusions, such as the painter's hand and its shadow, color shifts, and noise.

CR Categories: I.3.7 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classification;

Keywords: images, surfaces, depth, time, video, channel, segmentation, layers, photoshop, painting

*e-mail:tanjianchaoustc@gmail.com

1 Introduction

A painting is an arrangement of colors on a 2D canvas. During the painting process, artists deposit color throughout the canvas via a sequence of strokes, often with a real (or simulated, in the case of digital paintings) paint brush. The final painting is, from a computational point of view, a grid of unstructured color values. Extracting structure from the final painting is extremely challenging. Yet the temporal record, which is lost in the final painting, is informative about the scene being painted. Complex drawings are drawn according to a hierarchical structure [Taylor and Tversky 1992]. Objects and parts of objects are drawn together using lower-level rules [Van Sommers 1984; Novick and Tversky 1987; Tversky 1999].

Interacting with editing history is a powerful concept in human-computer interaction. (See Nancel and Cockburn's CAUSALITY [2014] for a recent survey and conceptual model.) This rich literature on history systems extends far beyond undo/redo. For *digital* image editing, this literature includes a generalization of layers for scaling, resizing, and recoloring strokes [Nancel and Cockburn 2014], revision control [Chen et al. 2011], grouping command history [Chen et al. 2012], learning from or reapplying previous commands [Grossman et al. 2010; Berthouzoz et al. 2011; Xing et al. 2014]. Wetpaint [Bonanni et al. 2009] explored a tangible "scraping" interaction to visualize layered information, such as a painting's history. Such powerful interactions are unavailable for physical paintings, even when digitized.

To enable these interactions, we propose to extract editing history from paintings by analyzing time lapse videos of their creation and decomposing them into translucent paint layers. These paint layers correspond to the “commands” necessary for history-based editing applications. Throughout this paper (unless stated otherwise), we use “paint” in a general sense to mean any kind of physical marking, including pencils, pens, markers, watercolor, acrylic, etc. Time lapse videos of paintings are readily available online, often for instructional or demonstrative purposes. Our analysis can also be used to recover a command history for time-lapse videos of digital paintings; this is useful for applying history-based editing operations to popular digital painting applications.

There are two primary challenges to extracting paint layers from time lapse videos. The first challenge is that, given two images from a time lapse, the paint layer which effected the observed difference is ambiguous. For example, an observed change from white to pink could be the result of opaque pink paint or translucent red paint. Because opaque paint completely covers or hides colors underneath, it limits the reuse of covered strokes in history-based editing operations. Therefore, we seek maximally translucent paint in order for the layers to be maximally reusable. In Section 3, we present solutions based on the Kubelka-Munk model of pigment layering [Kubelka 1948] and the standard Porter and Duff [Porter and Duff 1984] “over” blending equation used throughout computer graphics. The second challenge is that time-lapse painting videos contain many changes *not* due to the application of paint. Spurious changes may be caused by occlusions, such as the painter’s hand or brush and accompanying shadows; camera motion, refocusing, or color balance changes; compression noise, watermarks, or overlays; and dynamic effects like canvas vibration and the drying of watercolor. Unlike the foreground object subtraction problem in computer vision, some parts of the canvas may be occluded more often than not. In Section 4, we present a solution for processing real-world time-lapse painting videos. Our solution removes even extremely frequent occlusions, noise, and global color shifts. Camera and canvas motion are beyond the scope of this work.

Contributions

- A pipeline for processing real-world time-lapse paintings (Section 4), capable of suppressing long-term occlusions such as the painter’s hand and its shadow, short-term occlusions like brushes, noise due to compression and lighting, and color shifts.
- Extremely efficient algorithms for decomposing image sequences into translucent paint layers according to either the Kubelka-Munk model for physical material layering, an extension of their model for mixing, or the standard linear blending algorithm used in digital painting (Section 3).

Our contributions are in the generation of this data, not in its downstream applications (Section 5).

2 Related Work

Interacting with editing history In addition to the previously mentioned history-based interactions for image editing, related interactions have also been proposed for 2D and 3D vector graphics. Su et al. [2009] presented a technique for 2D vector graphics editing which suggests objects to select based on previous selections in the user’s command history. Noris et al. [2012] presented a scribble-based approach to segment 2D vector graphics sketches based on time of creation, which helps distinguish nearby strokes drawn at very different times. VisTrails [2009] is a commercial tool for reviewing and reusing command history in the commercial 3D modeling package Maya. Denning and Pellacini [2013] presented algorithms for revision control of 3D meshes. Chen et al. [2014] introduced a technique for choosing good views and segmenting 3D models based on the editing history. Two recent works analyze and visualize changes in outdoor, urban scenes [Matzen and Snavely 2014] and construction sites [Karsch et al. 2014]. These approaches operate on a collection of photographs from different viewpoints, relying on structure-from-motion to obtain a 3D reconstruction (the former) or a 3D architectural model (the latter). Finally, while not about editing history per se, McCann and Pollard [2009; 2012] introduced two noteworthy generalizations of layers for image editing.

Decomposing edits Hu et al. [2013] investigated the related problem of recovering an image editing operation from a pair of images. The editing operations they support are duplicating and geometrically transforming an image region, and adjusting the color of an image region. We solve the orthogonal problem of recovering maximally translucent paint layers using both a physical Kubelka-Munk model and the traditional “over” digital compositing operation. Our work also includes a far more efficient algorithm for finding per-pixel opacity for a single-color layer.

Amati and Brostow [2010] analyzed videos of sumi-e paintings, a style of monochromatic (shades of black) art with relatively few strokes. They find a small number of clean frames by comparing binary thresholded frames to the binary thresholded final painting. The result of their analysis is a segmentation of the final painting into parts (e.g. leaves and flowers). We are inspired by this work and share the observation that paint is far more temporally stable than occlusions, and that algorithms for skin detection and foreground subtraction are unsuitably unstable for analyzing painting videos.

Fu et al. [2011] extract an animated stroke order from static line drawings based on such cognitive and geometric properties. They operate at the part level and take as input a drawing segmented into objects. In contrast, we operate on paintings and are already given a time lapse sequence of its creation. Both our work and Fu et al. [2011] rely on a similar assumption, that the order of markings made when drawing or painting is not random.

Xu et al. [2006] introduced an algorithm that decomposes a single image of a Chinese painting into a collection of layered brush strokes. This comprises image segmentation, detecting the curves of painted strokes, and separating colors for overlapping strokes. Richardt et al. [2014] presented a semi-automatic approach to decompose single images into a mix of transparent and opaque vector graphic layers, where the vector graphics can be filled with linear and radial gradients. In contrast, our approach relies on a video of the creation process, which simplifies segmentation though not color separation. We treat color separation in terms of both the Kubelka-Munk physical layering model and the digital compositing “over” operation, and transform the problem into a simple geometric one in RGB-space.

Image matting Color separation is also related to layer extraction and blue-screen matting. Szeliski et al. [2000] presented a solution to the layer extraction problem in which two independent “images” are layered on top of each other as a result of reflection or transparency. Their approach requires that the layered images are moving with respect to each other, a reasonable assumption for a reflection on a window, but one that does not holds in our scenario. Farid and Adelson [1999] introduced another solution to this problem, but require as input two photographs taken with a polarized light filter. Smith and Blinn [1996] study the related blue-screen matting problem of separating a potentially translucent foreground object from one or two known backgrounds. Their analysis of the problem

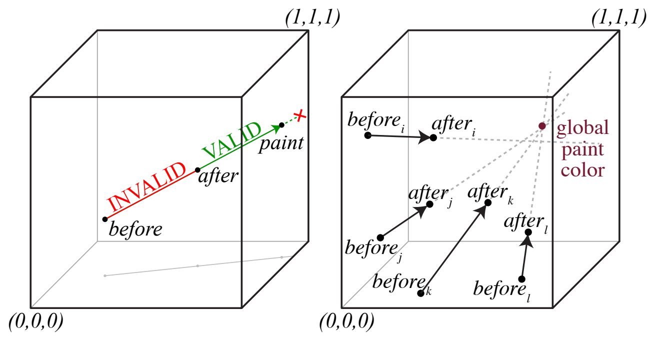
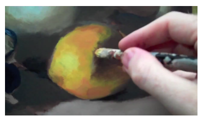


Figure 2: *The cube of valid RGB colors. Left: The color of a pixel before and after a modification to a painting defines a line. With Porter-Duff's "over" compositing, the paint color must lie on the portion of the line beyond after and within the cube. Right: In RGB-space, the after color of every pixel (here, i, j, k, l) affected by a stroke is the result of pulling its before color towards the stroke's color.*

shows that the problem is, in general, underspecified. Zongker et al. [1999] solve a generalized version of the matting problem which allows for reflections and refractions.

3 Decomposing Time Lapse Paintings



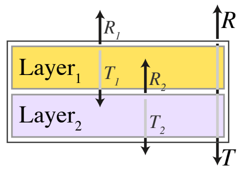
© Will Kemp

Viewed as a time lapse, the darkening of a lemon (right) has an ambiguous interpretation. The darker color could either be the result of painting with an opaque, darker shade of yellow, or else it could be the result of painting with a "translucent" black or brown pigment. (Translucency could be the result of pigment layering or mixing; see below.) The opaque interpretation is always possible, but completely hides all previous information, preventing its use in later editing applications. For history-based editing operations, we claim that the translucent interpretation is more useful and, in general, a better conceptual match to the painting process. In other words, we wish not to distinguish between paint mixed on the easel and paint mixed directly on the canvas, and to view even "opaque" acrylic paint as potentially translucent.

Input We take as input a time lapse recording of a painting in the form of a sequence of albedo or reflectance images $I_{t_1}, I_{t_2}, \ldots, I_{t_n}$. (We describe our process for obtaining such images from real-world videos in Section 4.) An image I_{t_i} stores the state of the painting at time t_i. Each RGB color channel of a pixel store a value in $[0, 1]$ representing the overall fraction of incident light that is reflected at that wavelength. Given two such images $I_{t_{i-1}}$ and I_{t_i}, our goal is to recover the per-pixel parameters of "paint" P_{t_i} such that applying P_{t_i} to $I_{t_{i-1}}$ results in I_{t_1}. We do this using a physical Kubelka-Munk model of material layering (Section 3.1, $P_{t_i} = (R_{t_i}, T_{t_i})$) and using the linear blending equation [Porter and Duff 1984] commonly used for digital compositing and painting (Section 3.2, $P_{t_i} = \text{RGB}_{t_i}$).

3.1 Recovering Physical Paint Parameters

Kubelka and Munk [Kubelka and Munk 1931; Kubelka 1948] modeled the reflectance R and transmittance T of a layer of homogeneous material in terms of the material's absorption and scattering coefficients K and S. (All parameters are per-wavelength.) The model is widely used in paint, plastic, paper, and textiles and has previously been used in computer graphics for accurately simulating paint [Konieczny and Meyer 2009; Curtis et al. 1997; Baxter et al. 2004; Haase and Meyer 1992; Lu et al. 2014; Budsberg 2007; Dannen 2012]. A summary of useful formulae related to the Kubelka-Munk model can be found in [Barbarić-Mikočević and Itrić 2011]. When multiple materials are present, the Kubelka-Munk model can be used in two ways: layering and mixing.



Layering Kubelka [1948] presented formulae for the overall reflectance and transmittance of a stack of non-opaque layers, given each layer's individual reflectance R and transmittance T coefficients.[1] R and T can be expressed in terms of absorption, scattering, and thickness parameters, but it is more straightforward and involves fewer parameters to simply consider R and T. In our scenario, we observe a sequence of reflectance images I_{t_i}. Each I_{t_i} stores the overall reflectance underneath every pixel at time t_i. I_{t_0}, the initial frame, stores the reflectance of the backing material, such as a blank canvas or sheet of paper. We wish to find the reflectance R_{t_i} and transmittance T_{t_i} of the paint layer that results in I_{t_i} when placed above $I_{t_{i-1}}$. The equation is [Kubelka 1948; Kubelka 1954]:

$$I_{t_i} = R_{t_i} + \frac{T_{t_i}^2 I_{t_{i-1}}}{1 - R_{t_i} I_{t_{i-1}}} \tag{1}$$

$R, T \in [0, 1]$ and, due to the conservation of energy, $T + R \leq 1$. (The fraction of illumination *absorbed* by the layer is $1 - T - R$.) In general, there are infinitely many solutions, including the opaque solution $T_{t_i} = 0, R_{t_i} = I_{t_i}$. To maximize the reusability of recovered layers, we find the solution that maximizes the transmittance T:

$$\begin{array}{llll}
\text{if} & I_{t_{i-1}} = 0: & R_{t_i} = I_{t_i}, & T_{t_i} = 1 - I_{t_i} \\
\text{else if} & I_{t_i} = 0: & R_{t_i} = 0, & T_{t_i} = 0 \\
\text{else if} & I_{t_i} + \frac{1}{I_{t_{i-1}}} \leq 2: & R_{t_i} = 0, & T_{t_i} = \sqrt{\frac{I_{t_i}}{I_{t_{i-1}}}} \\
\text{else if} & \frac{I_{t_i}}{I_{t_{i-1}}} \leq 1: & R_{t_i} = 0, & T_{t_i} = \sqrt{\frac{I_{t_i}}{I_{t_{i-1}}}} \\
\text{else:} & & R_{t_i} = X = \frac{\frac{I_{t_i}}{I_{t_{i-1}}} - 1}{I_{t_i} + \frac{1}{I_{t_{i-1}}} - 2} & \\
 & & T_{t_i} = 1 - X &
\end{array}$$

To the best of our knowledge, these transmittance-maximizing solutions have not previously been presented. See Appendix A for a derivation. In our setting, where the I are RGB reflectance images, R and T are also RGB images (with values in $[0, 1]$) and easily visualized (Figure 4). R is an additive image (transparent R is black) and T is a multiplicative image (transparent T is white). Our solution continuously transitions between multiplicative blending with bright backgrounds and additive blending with dark ones. Nearly all changes are partially translucent, and more extreme changes gradually become opaque. We compare our maximally transmissive solutions to scanned watercolor layers in Figure 3.

Mixing The Kubelka-Munk mixing model, while suitable for homogeneous mixtures such as wet paint, is not as suited for our purposes as layering. The mixing model can only approximate a completely opaque layer of paint via very large mixing coefficients (see supplemental materials). Moreover, the scattering and absorption parameters are less intuitive. They have no upper bound, unlike transmittance and reflectance, which range from 0 to 1, and can be visualized as an additive and multiplicative image. For completeness,

[1]Kubelka [1948] points out that the layering equation we use was independently derived and presented by Gurevic in 1930 and by Judd in 1934.

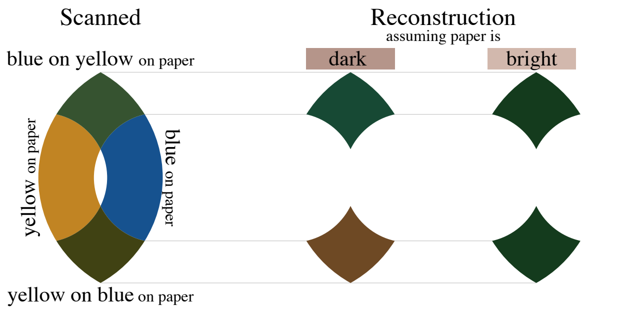


Figure 3: *Our Kubelka-Munk layering model recovers different reflectance and transmittance layers depending on the backing paper's brightness. To illustrate this difference, we scan blue and yellow watercolor paint and their overlap (left column). By assuming dark or bright backing paper, we can correspondingly recover different pairs of reflectance and transmittance layers for the blue and yellow paint, and reconstruct the overlap colors (middle and right columns). When assuming dark backing paper (middle column), recovered layers are both reflective (additive) and transmissive (multiplicative), and the overlap color will depend on the layer order. When assuming bright backing paper (right column), the recovered paint layers are purely transmissive (multiplicative). Purely transmissive layers commute; the overlap color is independent of the layer order, and the two reconstructed overlap colors are the same.*

we include solutions for minimal modification of current mixing parameters in the supplemental materials.

3.2 Recovering Digital Paint Parameters

In digital painting and compositing, the standard blending equation is Porter and Duff [1984]'s "over" operation:

$$\textit{after} = (1-\alpha)\textit{before} + \alpha \cdot \textit{paint} \tag{2}$$

In our setting, we treat *before* and *after* as the observed per-pixel RGB reflectance in $I_{t_{i-1}}$ and I_{t_i}. The layer's translucency $\alpha \in [0,1]$ is the interpolation parameter between *before* and *paint*.

To determine α and *paint*, we view the blending equation geometrically in RGB space (Figure 2). For every changed pixel, possible *paint* colors lie on the line passing through *before* and *after*, and, for a given *paint* color on the line, $\alpha = \frac{\textit{after}-\textit{before}}{\textit{paint}-\textit{before}}$. There are additional constraints: $0 < \alpha \leq 1$ and *paint*'s RGB components must all lie within $[0,1]$. Note that *paint* cannot deviate from this line, or else *before* would not exactly reconstruct *after*.

This is still, however, an under-constrained problem. We propose two techniques for choosing *paint* and α, one that minimizes α and one that computes a consistent *paint* color among pixels. A comparison of these techniques can be seen in Figure 4. To minimize α, each pixel's *paint* is chosen to be the intersection of its line with the RGB cube itself. This is equivalent to choosing the most extreme *paint* possible. This SMALL-ALPHA approach is extremely efficient and results in "minimally" opaque layers.

When minimizing α, however, pixels' *paint* colors aren't necessarily consistent (Figure 4). For short time-lapse intervals, however, we expect that the artist will only have used one color. Our CLOSEST-PAINT approach finds the color that minimizes the squared distance (in RGB-space) to every changed pixel's line. The minimizing color is then projected onto each pixel's $\overleftrightarrow{\textit{before after}}$ line. Solving the least squares problem entails solving a simple 3×3 linear system

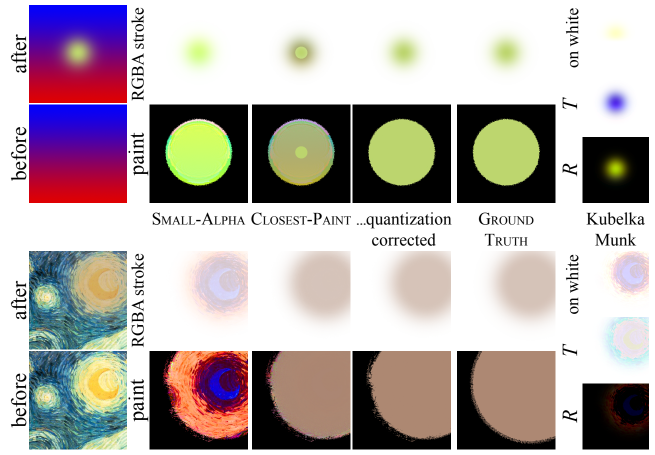


Figure 4: *Before and after a stroke is applied. The RGBA layers recovered by our algorithms* SMALL-ALPHA, CLOSEST-PAINT, CLOSEST-PAINT *with quantization correction, and the ground truth stroke itself. The Kubelka-Munk R and T values for the layer, and the R, T values composited with white.*

of equations (3 being the number of color channels):

$$E = \left\| \frac{u_i \cdot (\textit{paint} - \textit{before}_i)}{\|u_i\|^2} u_i + \textit{before}_i - \textit{paint} \right\|^2, \tag{3}$$

where $u_i = \textit{after}_i - \textit{before}_i$. Building the 3×3 system requires a summation over all n changed pixels. This approach is far more efficient than the $n \times n$ approaches presented in Xu et al. [2006] and Hu et al. [2013] and assumes nothing about α's smoothness.

The intuition behind the CLOSEST-PAINT approach is that every pixel affected by a stroke is pulled towards the stroke's *paint* color, by an amount determined by each pixel's α parameter (Figure 2). After projecting the global paint color onto the valid region of each pixels' line, we compute per-pixel α values.

Two minor improvements to CLOSEST-PAINT account for the quantization of (typically 8-bit) color component values. Taking quantization into account, we seek *paint* and α such that

$$\textit{after} = \text{ROUND}\left((1-\alpha)\textit{before} + \alpha \cdot \textit{paint}\right)$$

Because *after* is the result of rounding, the line that *paint* must lie on can pass through any part of the RGB "pixel cube" that rounds to *after*. The accuracy of each line's direction is proportional to its length, so our first improvement is to weight each term in Equation 3 by $\|\textit{after}_i - \textit{before}_i\|^2$. Once we solve the least squares problem and find the global paint color, our second improvement is to project *after* onto the line between *before* and the minimizing paint color—or as far towards the line as possible without crossing the boundary of *after*'s "pixel cube." We perform this quantization-correcting projection prior to the final projection of the minimizing color onto the valid region of the line passing through *before* and $\widehat{\textit{after}}$.

4 Processing time lapse videos

Time lapse painting videos are readily available online, often for instructional or demonstration purposes. Before these videos can be processed by our layer decomposition algorithms (Section 3), they must first be filtered to eliminate changes *not* due to the application of paint and finally converted to reflectance (albedo) images. Spurious changes can be caused by the environment (occlusions

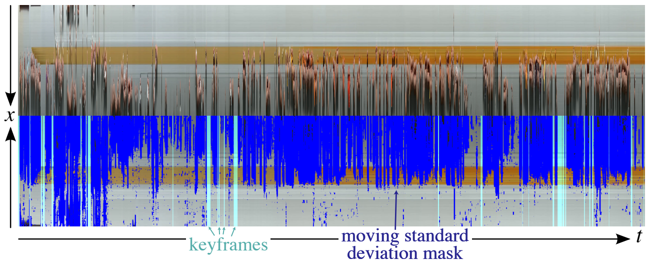


Figure 6: *A slice of the* egg *video in time, and its annotated reflection. Many pixels are occluded for the majority of frames. Video © Marcello Barenghi.*

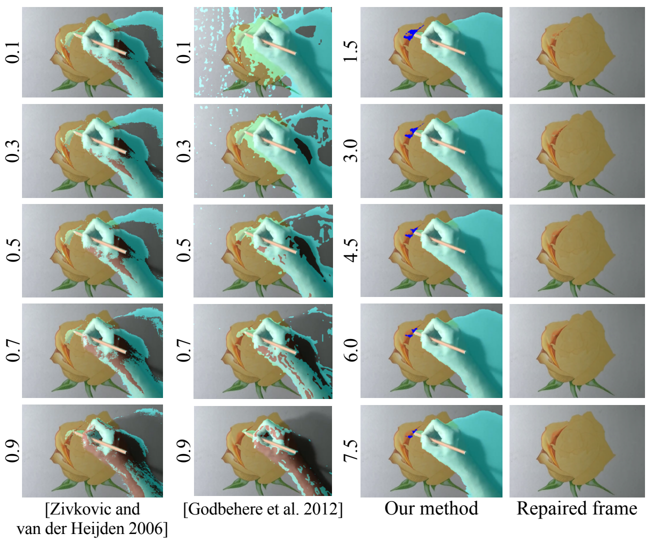


Figure 7: *Background estimation algorithms are challenged by time-lapse painting videos. We train Zivkovic and van der Heijden [2006] and Godbehere et al. [2012] on 200 frames and vary their respective thresholds. Both algorithms classify painted strokes as foreground (or, with a liberal threshold, misclassify the hand and shadow). Our algorithm's keyframe mask (cyan) suppresses the hand and shadow; the moving standard deviation (pure blue), with varying thresholds, masks the pencil tip and little additional paint. Video © Marcello Barenghi.*

of the canvas, such as the painter's hand or instruments, and accompanying shadows); camera (motion or color balance changes); post-processing (compression noise, permanent watermarks or overlays, and other visual effects); and dynamics (canvas vibration or motion and the drying of watercolor). In this work, we process time lapse videos with long-term occlusions, global color shifts, compression noise, and a mild amount of watercolor paint dynamics. We assume that the camera and canvas are fixed. Camera motion, canvas vibration and motion, and permanent post-processing watermarks and overlays are beyond the scope of this work. We further assume that the input video has been manually cropped to the canvas.

Existing background/foreground estimation algorithms are not well-suited for our particular input (a finding shared by Amati and Brostow [2010]). To begin, many pixels are occluded or in shadow for the *majority* of frames (Figure 6). The "background" in our case is the painting, which is constantly changing in tandem and exactly underneath the foreground object. The occluder includes a paintbrush whose tip is exactly the same color as the paint we are interested in. Likewise, we do not want to use a model of human skin, because the hand may be similar to the paint colors. Background estimation algorithms [Zivkovic and van der Heijden 2006; Godbehere et al. 2012] are challenged by our data and would produce large numbers of spurious changes (Figure 7). Finally, watercolor and markers take some time to dry, so they include some dynamics. If we waited until they were completely dry, layer order would be strongly affected.

The recent time lapse motion compensation algorithm of Rubinstein et al. [2011] is limited to very short sequences (300 frames took 50 hours, whereas our input sequences have ~5000 frames). They search for motion in space as well as time; we do not need this significant computational overhead. Nearly all steps in our video processing pipeline are per-pixel (and embarrassingly parallel).

4.1 Our pipeline

Our video processing pipeline (Figure 5) handles global color-shifts, removes noise, and is capable of ignoring frequent or majority occlusions. To motivate our approach, consider the changes to a horizontal slice of a painted egg in time (Figure 6). The color of an unpainted background pixel drifts significantly. The color of all pixels exhibit significant noise (some due to video compression and some due to global illumination effects). Many pixels, especially towards the right side of the painting (the painter's dominant handedness) are occluded in the *majority* of frames.

Our approach is based on two key observations. **I.** The value of an unoccluded pixel should be piecewise constant in time; equivalently, changes to a pixel on the canvas should be sparse in time. (The stability of paint versus occluders has also been noted by Amati and Brostow [2010].) Our approach is based on detecting and ignoring unstable pixels via moving standard deviation. **II.** Identical sequences of frames, which indicate that no occluders are present, provide crucial checkpoints for the progress of the painting. We call such identical frames **keyframes**. We use keyframes to mask and eliminate spurious changes in the intervening frames.

Color correction The first step in our pipeline corrects color drift between adjacent frames. We do not include all pixels in the calculation, as many pixels may have changed due to occlusion or paint. We solve for the per-channel linear function (offset and slope) that minimizes the color difference in a least-squares sense, considering only pixels whose magnitude of change lies between first and second octiles. (Pixels which changed the least may be due to outliers, e.g. oversaturated pixels.) We found this approach to be stabler than the non-linear color correction described in [Hu et al. 2013].

Keyframe detection The second step in our pipeline searches for keyframes, or sequences of repeated frames. We assume that a sequence of repeated frames implies no foreground occluders. We consider two frames to be repeated if they differ in less than n pixels; we detect differences with an L*a*b*-space threshold. (See Table 1 for all parameters.) After a keyframe is detected, we reduce noise by replacing its pixels with the per-pixel average in time. With these known good frames, we perform our color correction step a second time, this time aligning every frame with the most recent keyframe.

Inter-keyframe processing Keyframes allow us to reduce the number of pixels under consideration. A pixel which does not change from one keyframe to another should not change in the intervening frames; any such changes should be ignored. We compute a difference mask between adjacent keyframes for use in inter-keyframe processing. We again detect differences with a L*a*b*-space threshold and close pixel-sized gaps in the mask with a 3×3 topological closing operation. Outside of the mask, we linearly interpolate colors between keyframes. Inside of the mask, we detect and repair

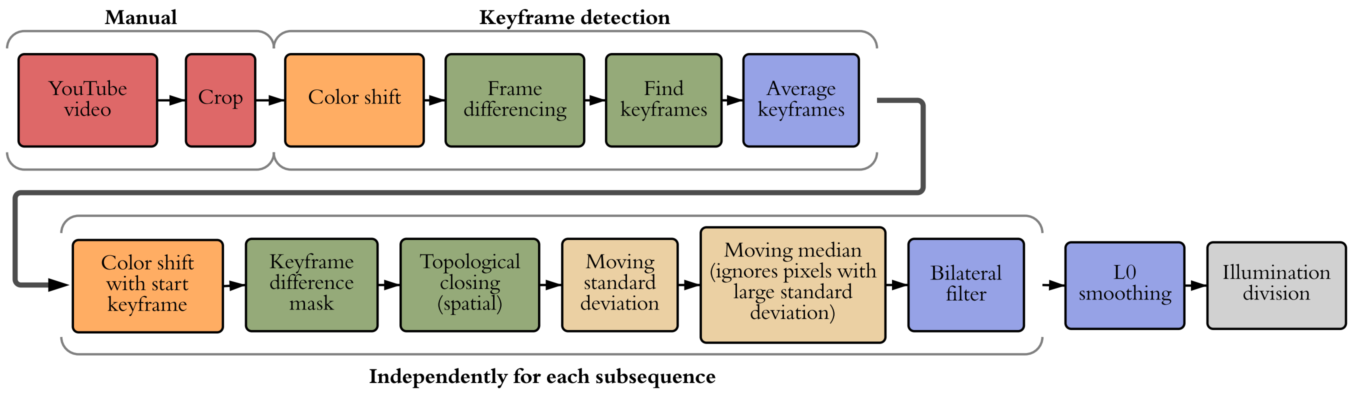

Manual
Keyframe detection
YouTube video
Crop
Color shift
Frame differencing
Find keyframes
Average keyframes
Color shift with start keyframe
Keyframe difference mask
Topological closing (spatial)
Moving standard deviation
Moving median (ignores pixels with large standard deviation)
Bilateral filter
L0 smoothing
Illumination division
Independently for each subsequence

Figure 5: *Our pipeline for processing time lapse painting videos. See Section 4.1 for details.*

	rose	egg	eye	apple	cube	candy	cola	graffiti
detection L*a*b* threshold	10	10	10	7	10	10	12	8
detection # differing pixels	50	50	50	50	50	50	80	50
detection # repeated frames	4	4	4	4	4	4	4	2
mask L*a*b* threshold	8	8	8	9	9	8	8	4
standard deviation threshold	1.5	0.5	0.5	0.6	1	0.8	1	1.5

Table 1: *Keyframe parameters in our video processing pipeline.*

occlusions. Painted marks are sparse in time and smoothly varying or piecewise constant; we detect unstable values—occluders—at each pixel with a thresholded L*a*b*-space moving standard deviation (in time, window size 7). We repair unstable pixels with a moving median filter (in time) [Cheung and Kamath 2004] whose window is made up of the $m = 9$ most recent stable pixels. The moving standard deviation threshold is the most sensitive parameter of our entire video processing pipeline. A small threshold minimizes false positives (occlusions) but degrades the temporal resolution of layers in the processed video. We have not found this to be a problem in our history-editing based applications (Section 5). See the supplemental materials for processed videos in which we manually selected thresholds for each subsequence to be as large as possible while still suppressing occluders.

Smoothing To suppress temporal noise and enforce temporal sparsity, we perform adaptive bilateral filtering [Tomasi and Manduchi 1998] on each inter-keyframe sequence (in time, with window size 15 and maximum $\sigma = 200$), and then we perform L_0 smoothing [Xu et al. 2011] (in time, $\kappa = 1.1, \lambda = 0.001$) for each pixel across all frames in the entire video. In 1D, Xu et al.'s L_0 smoothing algorithm amounts to repeatedly solving a tridiagonal system of equations; we constrain the very first and last pixels' values (with a hard constraint) to ensure that they remain unchanged.

Illumination division To convert our scrubbed videos into albedo (reflectance) images, we must divide each pixel's value by the incoming illumination. We assume that the spatial variation in illumination is unchanged throughout the entire sequence (e.g. the light source does not move) and divide each pixel's value by an illumination estimate computed from the first frame. In the first frame, the canvas is blank or nearly blank, so we perform a large, spatial median (window size 55) and use each pixel's maximum value (in any channel) as the unnormalized illumination divisor. Although the canvas material's true reflectance is unknown, estimates for the reflectance of paper placed on a non-white surface [Hubbe et al. 2008] typically range from 0.5 to 0.7. We normalize all pixels' illumination divisors with the globally brightest value encountered at any point in the video. This value is typically around 0.6; we cap it at 0.7. This step ensures that the brightest value in our albedo images is less than 1.

Input videos and the result of our pipeline can be seen in Figure 8 and in the supplemental material.

5 Applications

The layer decomposition for a painting stores its time lapse history as a sequence of RGBA or reflectance and transmittance images containing the individual layers (see Figure 8 and the supplementary materials) extracted from the original time lapse sequence using one of the algorithms from Section 3. As these layers typically affect only a small portion of the entire painting and are spatially coherent, they can be represented compactly in memory using simple run-length encoding. To achieve interactive response, we also store bounding boxes of layers and compare them with the bounding box of the edit so that the compositing operations can be done on a small fraction of the total pixels. Moreover, when the edit affects a certain interval in time, we can pre-composite prior and subsequent layers into two RGBA images or four reflectance and transmittance images, which are then blended together with the modified content.

Translucency-maximizing solutions are stable, general, and useful. The CLOSEST-PAINT method, which seeks to recover the true color and transparency, is appropriate for clean data where the difference between frames truly is a Porter-Duff over composite with a single color, such as digital painting sequences (supplemental materials). Physical painting sequences are noisy, so we use translucency-maximizing solutions, which make no assumptions about the input data. The Kubelka-Munk layering model is a physically-based approach. When editing, the reflectance and transmittance layers can be hue-shifted or modulated (increasing transparency and decreasing the reflectance). The linear (Porter-Duff) model is "correct" for extracting digital layers. It also works well for physical paintings. It has the advantage that it is simple to work with, as it is built into digital editing tools and GPU's.

The apple in Figures 1 & 9 and the rose in Figure 9 were edited with the Porter-Duff model. The eye in Figure 12 was edited with the Kubelka-Munk layering model. The Kubelka-Munk model typically produces layers that are more "translucent" than those produced in the Porter-Duff model. The graffiti example in Figure 10 provides a side-by-side comparison of the two models.

For videos with approximately 500×500 resolution, our entire pipeline runs at around 2 frames per second on a single core of an Intel Core i7-3520M CPU (except for the whole-video L_0 smoothing stage). Most stages are pixelwise, and the algorithms are embar-

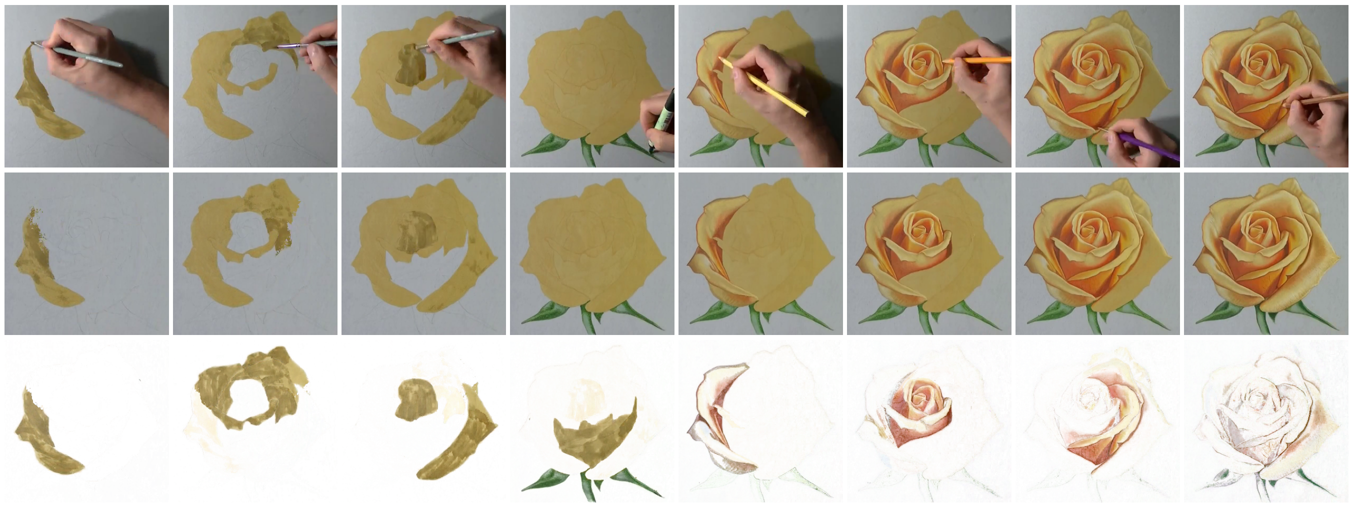


Figure 8: *Examples of layers (bottom) reconstructed from a time lapse video (top) which was pre-processed using our pipeline (middle). Individual layers were grouped into larger clusters to enhance their visibility. Time lapse © Marcello Barenghi.*

rassingly parallelizable. The final whole-video L_0 smoothing stage was parallelized and run on a 60-core cluster of Intel Xeon E5-2670 CPU's; this stage took around 10 minutes for a ~5000 frame video. The decomposition of the ~5000 frame rose example takes ~60 MB to store as a sequence of RGBA images (PNG format) and ~140 MB as a sequence of Kubelka-Munk reflectance and transmittance images (as gzipped double-precision floating point values).

Given a painting and its layer decomposition, we can perform various selection and editing operations (see Figure 1, 9, 10, 11, 12, and the supplementary materials) that generalize the notion of layers in digital painting programs.

Spatio-temporal selection & layering There are several possibilities for using temporal information for selection. Users can position the mouse at a specific spatial location and pick the temporal value of the pixel underneath. They can also alter the selected temporal value by scrubbing through time with the mouse wheel. When two or more locations are specified, the tool can select layers which lie in the specified time interval. A more complex selection can be achieved using selection strokes—scribbles. In this scenario algorithms based on temporal information such as [Noris et al. 2012] can be used to improve segmentation results. In Figure 11 we present a simplified solution where temporal statistics of all pixels underneath the scribble are analyzed and then a set of dominant temporal clusters is retrieved which can then be offered to the user as possible candidates for selection. This approach can also be understood as a supervised layer decomposition that helps artist convert the layer decomposition into a smaller set of meaningful layers. The user can perform various edits which composite nicely with prior and subsequent layers, as in a digital image editing workflow (see Figure 1). The great advantage of our approach is that the user can create different sets of layers *ex post* even for physical paintings, in contrast with the traditional, digital workflow which involves planning in advance.

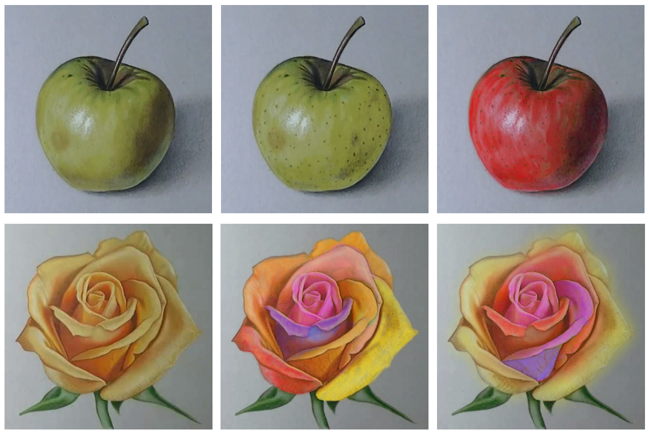


Figure 9: *Edits created using our Porter-Duff layer decomposition on the apple sequence from Figure 1 (top), and on the rose sequence (bottom, original image is on the left). Time lapse sequences © Marcello Barenghi.*

Modification of extracted layers Besides recoloring extracted layers or clustering them, the user can also modify their structure directly or use temporal information to perform additional operations. It is possible to erase or re-order selected layers, which has the effect of cloning, undo-ing or replaying a stroke at a different spatial or temporal location. Existing algorithms can be used to perform local layering [McCann and Pollard 2009] as well as soft stacking [McCann and Pollard 2012] to rearrange recovered layers and let them appear partially at multiple time frames. The user may also draw new strokes at specific times, create a spatial selection at a specific time and then move this selection to a different time to perform edits, or use time-of-creation as a parameter which can affect edits, such as the creation of color gradients (see Figure 1, 9, 10, 12, and supplementary material for illustrations of these operations).

6 Conclusions and Future Work

Our video processing pipeline tailored to time lapse painting videos can clean the videos of even persistent occlusions, and produce reflectance images suitable for paint layer decomposition and editing. The less-used Kubelka-Munk layering equations are a convenient alternative to their material mixing model and more suitable for real-world data than the "over" blending operation used throughout computer graphics. These equations can be "solved" efficiently to find maximally transparent stroke layers. With a single-stroke assumption for digital images, our 3×3 system can recover the

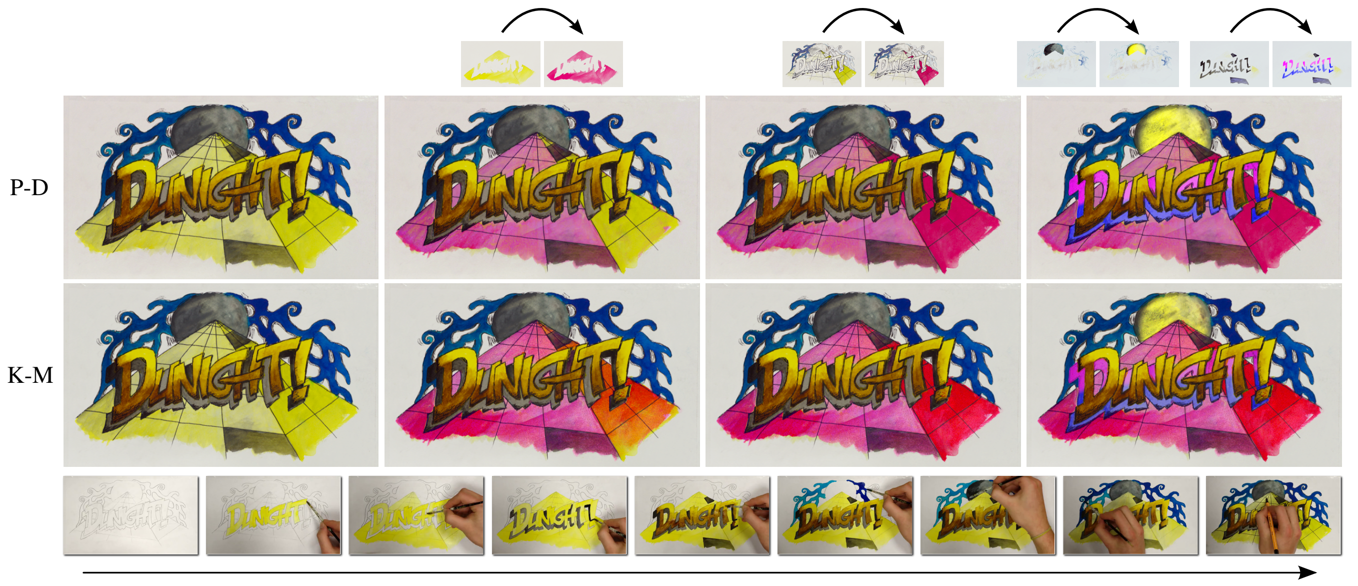


Figure 10: *Comparing edits created with our layer decomposition using the Porter-Duff (first row) and Kubelka-Munk (second row) models. In each model, similar clusters of layers were recolored. Note the difference in layer translucency and the effect of recoloring. The original unmodified painting is on the left while the final composition is on the right. Thumbnails of individual edits are shown on top of the Figure. Time lapse video © Matyáš Veselý.*

original stroke color and its alpha channel extremely robustly (within $\frac{1}{255}$ in each channel).

The history of a painting contains valuable information that can be stored efficiently using run-length encoding and used for a variety of history-based editing applications to generalize the notion of layers. A live implementation for decomposing on-the-fly would allow artists to use physical tools as their artistic input device.

One limitation of our approach is that we do not extract vector strokes. This is challenging because physical tools may create quite complex brush "shapes," and an approximation, when replayed, would not exactly match the final painting.

If an input video has too-low temporal resolution, multiple overlapping strokes could be painted between adjacent frames, and the original painted colors may not be recoverable. Moreover, the greater the magnitude of change, the less transparent the solution. Extreme noise manifests as spurious strokes that appear and then disappear. Artist drawing order will affect applications such as selection based on time ranges (including our clustering). This is true for all applications of editing history, regardless of the problem domain. Our work provides a new data source, but does not aim to solve problems in the literature on interacting with editing history.

In the future, we plan to replace software instrumentation in systems such as Chronicle [Grossman et al. 2010] and Chen et al. [2011; 2012] with our framework, in order to deliver similar functionality for both digital and real-world paintings. One can imagine, for example, an interactive gallery where visitors can inspect the creation process of individual exhibited paintings using visualization systems like WetPaint [Bonanni et al. 2009]. Recovered layers of real world paintings can also be used to generate painting tutorials in the spirit of Grabler et al. [2009]. Our technique can also enrich the tool set of appearance operators for inverse image editing [Hu et al. 2013]. Such operators can be used for content-adaptive macros [Berthouzoz et al. 2011], to predict the final appearance by example, and to perform automatic completion as in [Xing et al. 2014]. We wish to explore applications in forensics and artwork restoration. Finally, we wish to expand the class of videos we are able to process. Canvas vibrations are a challenge unique to our domain.

A Kubelka-Munk Layering Derivation

Recall Equation 1 and our solution. The first two solutions, when $I_{t_i} = 0$ or $I_{t_{i-1}} = 0$, are self-evident. For $I_{t_{i-1}} \neq 0$, we can express Equation 1 as $T_{t_i}^2 = (R_{t_i} - I_{t_i})(R_{t_i} - \frac{1}{I_{t_{i-1}}})$. Recall our physical constraints $0 \leq R_{t_i}, T_{t_i}, I_{t_i}, I_{t_{i-1}} \leq 1$ and $R_{t_i} + T_{t_i} \leq 1$. The latter is equivalent to $T_{t_i}^2 \leq (1 - R_{t_i})^2$. Thus we have an equality for $T_{t_i}^2$ which is a parabola in R_{t_i}, and an inequality for $T_{t_i}^2$, also a parabola in R_{t_i}. Both parabola open upwards, and both reach their minima when $R_{t_i} > 0$ (because neither I_{t_i} nor $I_{t_{i-1}}$ equal 0), and so are already increasing in $T_{t_i}^2$ when $R_{t_i} = 0$. The remaining solutions follow algebraically by intersecting the parabolas with each other and with the line $T_{t_i}^2 = 1$.

Acknowledgements

We would like to thank Bob Sumner, Paula Crawford, Jay Hendrick, Nathan Loda, and all anonymous reviewers for their constructive comments. We are grateful to Stephen Marschner for an illuminating discussion about the Kubelka-Munk mixing model and to Chris Kim, Will Xu, and Jae Yun for initial experiments. Authors Tan and Gingold were supported in part by the United States National Science Foundation (IIS-1451198 and IIS-1453018) and a Google research award. Authors Dvorožňák and Sýkora were supported by the Technology Agency of the Czech Republic under the research program TE01020415 (V3C – Visual Computing Competence Center), by the Czech Science Foundation under research program P202/12/2413 (OPALIS), and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods). Several experiments were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA. (URL:`http://orc.gmu.edu`).

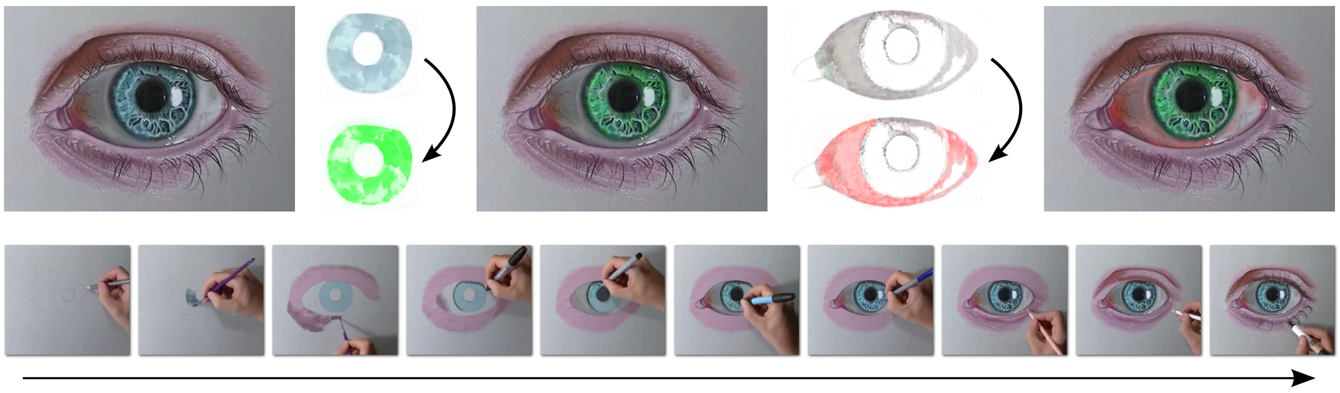


Figure 12: *Editing a decomposed painting with the Kubelka-Munk model. The user modifies colors in two extracted layers to adjust the original painting. Unprocessed time lapse recording of the painting process is depicted below. Time lapse video © Marcello Barenghi.*

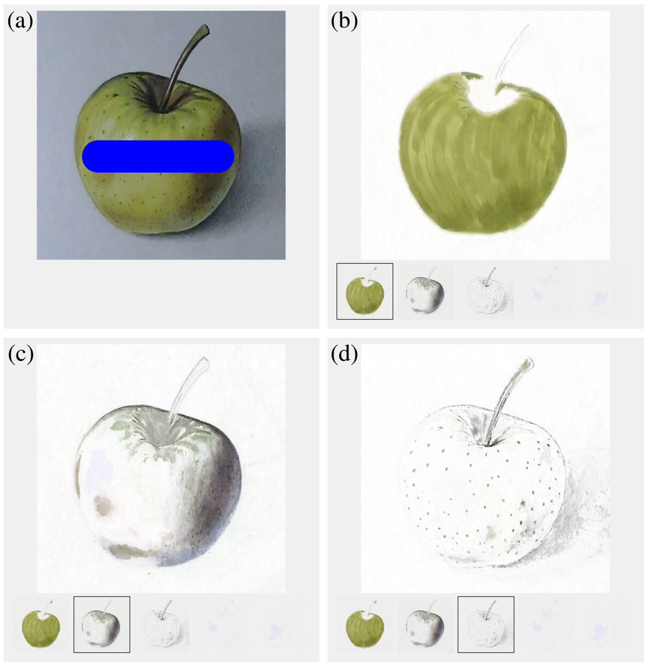


Figure 11: *Image selection & layering: the user draws a selection scribble (a), the system then suggests several temporal clusters containing pixels underneath the scribble (b-d). The user can select the appropriate cluster and use it as a layer for further editing. Source time lapse video © Marcello Barenghi.*

References

AMATI, C., AND BROSTOW, G. J. 2010. Modeling 2.5D plants from ink paintings. In *Proceedings of Eurographics Symposium on Sketch-Based Interfaces and Modeling Symposium*, 41–48.

BARBARIĆ-MIKOČEVIĆ, V. D.-M. Ž., AND ITRIĆ, K. 2011. Kubelka-Munk theory in describing optical properties of paper (i). *Technical Gazette 18*, 1, 117–124.

BAXTER, W. V., WENDT, J., AND LIN, M. C. 2004. IMPaSTo: A realistic, interactive model for paint. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*, 45–56.

BERTHOUZOZ, F., LI, W., DONTCHEVA, M., AND AGRAWALA, M. 2011. A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Transactions on Graphics 30*, 5, 120.

BONANNI, L., XIAO, X., HOCKENBERRY, M., SUBRAMANI, P., ISHII, H., SERACINI, M., AND SCHULZE, J. 2009. Wetpaint: Scraping through multi-layered images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 571–574.

BUDSBERG, J. B. 2007. *Pigmented Colorants: Dependency on Media and Time*. Master's thesis, Cornell Univrsity, Ithaca, New York, USA.

CHEN, H.-T., WEI, L.-Y., AND CHANG, C.-F. 2011. Nonlinear revision control for images. *ACM Transactions on Graphics 30*, 4, 105.

CHEN, T., WEI, L.-Y., HARTMANN, B., AND AGRAWALA, M. 2012. Data-driven history list for image editing. Tech. Rep. TR-2012-07, The University of Hong Kong.

CHEN, H.-T., GROSSMAN, T., WEI, L.-Y., SCHMIDT, R. M., HARTMANN, B., FITZMAURICE, G., AND AGRAWALA, M. 2014. History assisted view authoring for 3D models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2027–2036.

CHEUNG, S.-C. S., AND KAMATH, C. 2004. Robust techniques for background subtraction in urban traffic video. In *Proceedings of SPIE*, vol. 5308, 881–892.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. In *ACM SIGGRAPH Conference Proceedings*, 421–430.

DANNEN, C., 2012. The magical tech behind Paper for iPad's color-mixing perfection, Nov. Accessed: January 10th, 2015.

DENNING, J. D., AND PELLACINI, F. 2013. MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Transactions on Graphics 32*, 4, 35.

FARID, H., AND ADELSON, E. H. 1999. Separating reflections from images by use of independent component analysis. *Journal of the Optical Society of America A 16*, 9, 2136–2145.

FU, H., ZHOU, S., LIU, L., AND MITRA, N. J. 2011. Animated construction of line drawings. *ACM Transactions on Graphics 30*, 6, 133.

GODBEHERE, A., MATSUKAWA, A., AND GOLDBERG, K. 2012. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *Proceedings of American Control Conference*, 4305–4312.

GRABLER, F., AGRAWALA, M., LI, W., DONTCHEVA, M., AND IGARASHI, T. 2009. Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics 28*, 3, 66.

GROSSMAN, T., MATEJKA, J., AND FITZMAURICE, G. 2010. Chronicle: Capture, exploration, and playback of document workflow histories. In *Proceedings of ACM Symposium on User Interface Software and Technology*, 143–152.

HAASE, C. S., AND MEYER, G. W. 1992. Modeling pigmented materials for realistic image synthesis. *ACM Transactions on Graphics 11*, 4, 305–335.

HU, S.-M., XU, K., MA, L.-Q., LIU, B., JIANG, B.-Y., AND WANG, J. 2013. Inverse image editing: Recovering a semantic editing history from a before-and-after image pair. *ACM Transactions on Graphics 32*, 6, 194.

HUBBE, M. A., PAWLAK, J. J., AND KOUKOULAS, A. A. 2008. Paper's appearance: A review. *BioResources 3*, 2, 627–665.

KARSCH, K., GOLPARVAR-FARD, M., AND FORSYTH, D. 2014. Constructaide: Analyzing and visualizing construction sites through photographs and building models. *ACM Transactions on Graphics 33*, 6, 176.

KONIECZNY, J., AND MEYER, G. 2009. Airbrush simulation for artwork and computer modeling. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*, 61–69.

KUBELKA, P., AND MUNK, F. 1931. An article on optics of paint layers. *Zeitschrift für Technische Physik 12*, 593–601.

KUBELKA, P. 1948. New contributions to the optics of intensely light-scattering materials. Part I. *Journal of the Optical Society of America 38*, 5, 448–448.

KUBELKA, P. 1954. New contributions to the optics of intensely light-scattering materials. Part II: Nonhomogeneous layers. *Journal of the Optical Society of America 44*, 4, 330–334.

LU, J., DIVERDI, S., CHEN, W. A., BARNES, C., AND FINKELSTEIN, A. 2014. RealPigment: Paint compositing by example. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*, 21–30.

MATZEN, K., AND SNAVELY, N. 2014. Scene chronology. In *Proceedings of European Conference on Computer Vision*. 615–630.

MCCANN, J., AND POLLARD, N. 2009. Local layering. *ACM Transactions on Graphics 28*, 3, 84.

MCCANN, J., AND POLLARD, N. 2012. Soft stacking. *Computer Graphics Forum 31*, 2, 469–478.

NANCEL, M., AND COCKBURN, A. 2014. Causality: A conceptual model of interaction history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1777–1786.

NORIS, G., SÝKORA, D., SHAMIR, A., COROS, S., WHITED, B., SIMMONS, M., HORNUNG, A., GROSS, M., AND SUMNER, R. 2012. Smart scribbles for sketch segmentation. *Computer Graphics Forum 31*, 8, 2516–2527.

NOVICK, L. R., AND TVERSKY, B. 1987. Cognitive constraints on ordering operations: The case of geometric analogies. *Journal of Experimental Psychology: General 116*, 1, 50.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. *ACM SIGGRAPH Computer Graphics 18*, 3, 253–259.

RICHARDT, C., LOPEZ-MORENO, J., BOUSSEAU, A., AGRAWALA, M., AND DRETTAKIS, G. 2014. Vectorising bitmaps into semi-transparent gradient layers. *Computer Graphics Forum (Proceedings of EGSR) 33*, 4 (July), 11–19.

RUBINSTEIN, M., LIU, C., SAND, P., DURAND, F., AND FREEMAN, W. T. 2011. Motion denoising with application to time-lapse photography. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 313–320.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *ACM SIGGRAPH Conference Proceedings*, 259–268.

SU, S. L., PARIS, S., AND DURAND, F. 2009. QuickSelect: History-based selection expansion. In *Proceedings of Graphics Interface*, 215–221.

SZELISKI, R., AVIDAN, S., AND ANANDAN, P. 2000. Layer extraction from multiple images containing reflections and transparency. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 246–253.

TANGE, O. 2011. GNU Parallel: The command-line power tool. *;login: The USENIX Magazine 36*, 1 (Feb), 42–47.

TAYLOR, H. A., AND TVERSKY, B. 1992. Descriptions and depictions of environments. *Memory & Cognition 20*, 5, 483–496.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of IEEE International Conference on Computer Vision*, 839–846.

TVERSKY, B. 1999. What does drawing reveal about thinking? *Visual and Spatial Reasoning in Design*, 93–101.

VAN SOMMERS, P. 1984. *Drawing and cognition: Descriptive and experimental studies of graphic production processes.* Cambridge University Press.

VISTRAILS, I., 2009. Vistrails provenance explorer for maya. `http://www.vistrails.com/maya.html`.

XING, J., CHEN, H.-T., AND WEI, L.-Y. 2014. Autocomplete painting repetitions. *ACM Transactions on Graphics 33*, 6, 172.

XU, S., XU, Y., KANG, S. B., SALESIN, D. H., PAN, Y., AND SHUM, H.-Y. 2006. Animating chinese paintings through stroke-based decomposition. *ACM Transactions on Graphics 25*, 2, 239–267.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via L0 gradient minimization. *ACM Transactions on Graphics 30*, 6, 174.

ZIVKOVIC, Z., AND VAN DER HEIJDEN, F. 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters 27*, 7, 773–780.

ZONGKER, D. E., WERNER, D. M., CURLESS, B., AND SALESIN, D. H. 1999. Environment matting and compositing. In *ACM SIGGRAPH Conference Proceedings*, 205–214.