Example-Based Video Color Grading

Nicolas Bonneel[1]* Kalyan Sunkavalli[2] Sylvain Paris[2] Hanspeter Pfister[1]
[1]Harvard University [2]Adobe

(a) *Input segmented video* **(b)** *Model segmented video "Transformers"* **(c)** *Our result*

Figure 1: *Color grading is the process of adjusting the color and tonal balance of a movie to achieve a specific look. This is a critical step of the movie editing pipeline. However, even with dedicated software, it remains a painstaking task that can be done only by skilled artists. We propose a new model-based approach that automatically transfers the look of a professionally edited sequence to another video. To produce sophisticated effects like the contrasted orange-teal look of this example, we use a user-provided foreground-background segmentation. This allows us to process the input sequence (a) to reproduce the characteristic visual style of "Transformers" (b) to convey a similar tense mood (c). Our approach produces results that are free from artifacts and temporally coherent, as can be seen in the companion video. Video credits: Mike Lerner (input), "Transformers"(2007) ©Paramount Pictures (model).*

Abstract

In most professional cinema productions, the color palette of the movie is painstakingly adjusted by a team of skilled colorists – through a process referred to as *color grading* – to achieve a certain visual look. The time and expertise required to grade a video makes it difficult for amateurs to manipulate the colors of their own video clips. In this work, we present a method that allows a user to transfer the color palette of a model video clip to their own video sequence. We estimate a per-frame color transform that maps the color distributions in the input video sequence to that of the model video clip. Applying this transformation naively leads to artifacts such as bleeding and flickering. Instead, we propose a novel differential-geometry-based scheme that interpolates these transformations in a manner that minimizes their curvature, similarly to curvature flows. In addition, we automatically determine a set of keyframes that best represent this interpolated transformation curve, and can be used subsequently, to manually refine the color grade. We show how our method can successfully transfer color palettes between videos for a range of visual styles and a number of input video clips.

CR Categories: I.4.3 [Computing Methodologies]: Image Processing and Computer Vision—Enhancement;

Keywords: color grading, color transfer, video, visual look

Links: DL PDF

*e-mail:nbonneel@seas.harvard.edu

1 Introduction

The color palette used in a movie often plays a critical role in establishing its visual look. It can be used to locate a movie in place and time – for example, the Coen Brothers' 2000 film, *O' Brother, Where Art Thou?* uses a sepia-tinted color scheme to evoke its setting of rural Mississippi during the time of the Great Depression[1]. In other instances, the color scheme is manipulated to evoke certain emotions or reinforce a certain mood (as demonstrated by Jean-Pierre Jeunet's use of rich, warm colors to reinforce the vibrant, happy mood of his 2001 film, *Amélie*). Over time, certain looks have come to represent entire genres of movies – *Film Noir's* use of low-key lighting and contrast between light and shadows is one such iconic visual style.

This relationship between visual styles and the process of storytelling [Oldenborg 2006] makes color management a critical part of film production. The visual style of a movie is often carefully devised by the cinematographer, and executed by a team of skilled colorists who manipulate the colors of the movie footage – through a process known as *color grading* – to match his or her vision. While in the past color grading was done using photo-chemical processing, most modern post-production pipelines digitize the movie footage and use a combination of hardware and software tools to digitally color grade the movie [Selan 2012]. Today, color grading tools are even part of popular video processing software such as After Effects and Final Cut Pro.

However, in spite of the range of tools available today, color grading is still a tedious process that requires a skill level and time budget that puts it out of the reach of amateur video enthusiasts. The goal of our work is to make it possible for amateur users to apply popular color grading styles to their own home videos with minimal user interaction. We achieve this using an example-based approach; users are asked to specify a *model* video (or image) that represents the color grading style they like, and our technique transfers the color palette of this model video to their clip. This approach offers two advantages; first, it allows users to specify the visual style they would like in an intuitive manner, and second, it allows us to leverage the

[1]The short documentary *Painting with Pixels: O' Brother, Where Art Thou?* offers a fascinating perspective into this process.

Figure 2: *(a) We evaluate the temporal consistency of several color transfer options on four frames. The first three frames are nearly consecutive and the last one is more distant. Computing the color transform once at the beginning and applying it to the entire sequence yields results that degrade as time passes (e.g., the bluish face on the last frame). (b) Evaluating the transform at each frame produces temporally inconsistent results when the content changes (e.g., when the man appears in the second frame). (c) In comparison, our result is stable and does not drift. Video credits: Greg Henkel.*

skill and time that went into the grading of the model video clip.

By posing video color grading as the problem of transferring color distributions between video sequences, we can leverage an extensive body of work on color matching in images [Reinhard et al. 2001; Pitié et al. 2005]. However, these methods cannot be easily extended to video sequences – applying color matching naively to every frame of video sequences leads to artifacts such as color bleeding and temporal incoherence (see Fig. 2). Instead, in our work, we estimate per-frame color transformations, and filter these transformations using a novel temporal filtering technique based on new approximate curvature flow and differential geometry principles. This filtering technique considers the sequence of transformations as a high-dimensional curve in a manifold, and replaces pieces of this curve that vary at a high frequency by a geodesic interpolation. This allows us to preserve the video content faithfully while simultaneously handling temporal consistency issues at both short scales (for example, high-frequency flickering) and longer scales (for example, changes in the scene or lighting). In addition, our filtering technique also allows us to automatically determine a small set of keyframes that can be used to further artistically manipulate the color palette. As we show in our results, our technique is able to handle a number of visual styles (including *Film Noir*, *bleach bypass*, *orange-teal*) and a wide range of input video sequences.

Contributions We describe a technique to transfer the look of professionally color graded footage onto amateur videos. We rely on a robust, low-dimensional representation of the visual style of each frame that can handle a wide variety of looks and contents. The core of our technique is based on a new filtering approach to temporal consistency. We explain how to extend the concept of curvature-flow smoothing often used on 3D meshes to higher-dimensional function spaces. In our context, we apply this smoothing scheme in the space of color transformations and we demonstrate that it successfully produces temporally consistent results without degrading the video content. Further, our final result is defined by a small number of parameters stored at a few sparse keyframes that can be easily edited by artists, thereby preserving their full artistic control over the output.

2 Previous work

Color transfer for images Matching colors in images has attracted much attention since the pioneering work of Reinhard et al. [2001]. These methods have either tried to match the entire color distribution of the model image [Pitié et al. 2005] or focused on low order statistics so as to avoid artifacts such as overly high contrasts [Reinhard et al. 2001; Pitié and Kokaram 2007]. Accounting for spatial knowledge has shown to improve results when the content of the two images differs, by transferring colors locally between corresponding segments or using user specified strokes [Reinhard et al. 2001; Tai et al. 2005; An and Pellacini 2010; Pouli and Reinhard 2011], or spatially matching [HaCohen et al. 2011] or aligning [Kagarlitsky et al. 2009] the images. Color transfer is important to a number of applications; it has been used to mimic the style of a specific camera [Wang et al. 2011], to convey semantic concepts [Murray et al. 2011], and to increase the realism of composite images [Xue et al. 2012; Johnson et al. 2011]. In addition to color, other aspects of appearance also effect the visual look of an image. Bae et al. [2006] transfer tonal balance and detail across images using a non linear two-scale decomposition.

In our work, we adapt features of these previous techniques to capture a large range of color palettes. We use a color transfer model inspired by color grading software used by artists, e.g., Da Vinci Resolve and SpeedGrade. It consists of a nonlinear luminance curve and affine remapping of the chrominance values. These transformations are applied in three luminance bands and for each user-defined segment of the video.

Temporally consistent video processing Extending image-based algorithms to video sequences is non-trivial, and is especially diffi-

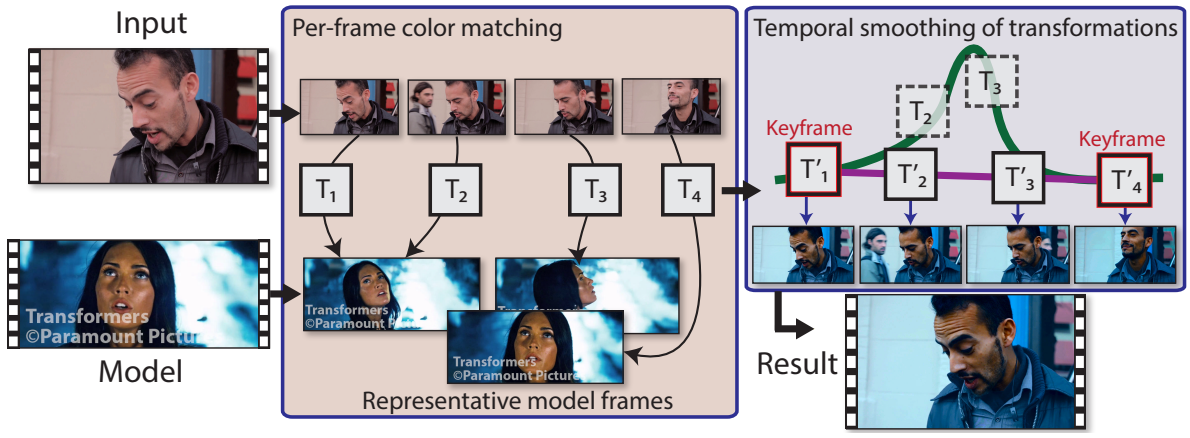


Figure 3: *Our color grading method works in two stages. In the first stage, we match each frame of the input video to one image out of a set of representative model video frames. We estimate color transformations between these matching frames. In the second stage, we filter these transformations using a novel approximate curvature flow technique. The basic idea behind this technique is to treat the set of transformations as a curve in high-dimensional space and detect points of low curvature (i.e., keyframes). Interpolating the color transformations at these keyframes produces a temporally coherent set of transformations, that when applied to the input video, results in a high-quality color graded result. Video credits: "Transformers" (2007) ©Paramount Pictures (model).*

cult for the kind of videos we are targeting – clips captured by amateurs under largely uncontrolled conditions, and characterized by complex scenes, arbitrary motion, and changes in illumination and camera parameters. If these effects are not accounted for properly, the color graded results often exhibit artifacts such as high-frequency flickering or low-frequency color drifts (Fig. 2).

Paris [2008] analyzed an extension of Gaussian convolution to video streams, and used it to devise temporally-consistent video filtering techniques (including edge-preserving smoothing). Lang et al. [2012] used efficient edge preserving filtering techniques to estimate temporally coherent optical flow and used it to make graphics applications such as colorization temporally coherent. Farbman and Lischinski [2011] stabilize tonal fluctuations in videos by using optical flow estimated at sparse locations to propagate tonal differences across frames. Oskam et al. [2012] employ spatio-temporally consistent radial basis functions to color balance sequences for augmented reality. However, all these methods directly filter the pixel colors and would generate overly smooth results, which would be undesirable in our context. We address this issue by working in the space of color transforms. Our scheme preserves the frame details while ensuring temporal smoothness.

Kiser et al. [2012] devise a temporally stable tone mapping system that relies on a leaky integrator to smooth out the parameters estimated at every frame. Similarly, Chang et al. [2007] use anisotropic diffusion to ensure that subtle color transformations are temporally coherent. In our context, low-pass filtering the color transformations computed at every frame might smooth out noisy transformations, but does so at the cost of corrupting the transformations at neighboring frames, i.e., large variations in color distributions over short periods of time become smaller variations that are extended over a longer period. In our work, we aim at explicitly detecting and removing the outlier color transformations; this ensures that the results are temporally consistent and also accurately model the true scene content.

3 Overview

Fig. 3 shows an overview of our process. Given a user-specified model video $\mathbf{M}$ and an input clip $\mathbf{I}$, we seek to transfer the color palette of $\mathbf{M}$ to $\mathbf{I}$ to create the color graded result $\mathbf{O}$. Optionally, a segmentation of the two videos into foreground and background regions can be used for improved matching results. There are two simple approaches to this problem that seem viable at the outset. One is to estimate a *single* global color transfer function T that best matches the colors of the input video to those of the output video, and apply that transfer function to every input video frame, i.e., $\mathbf{O}_t = T(\mathbf{I}_t)$. The other approach is to estimate a color transfer function $\mathbf{T}_t$ for *every* input video frame $\mathbf{I}_t$ that maps its colors to those of a corresponding model video frame, and apply it to produce the result, i.e., $\mathbf{O}_t = \mathbf{T}_t(\mathbf{I}_t)$. While these approaches would work for some videos, they do not generalize well to the range of videos that we would like to handle – videos captured with arbitrary scene content, complex motion, and changes in lighting and camera settings that lead to changes in pixel values over time. This is illustrated in Fig. 2, where using a single global color transform leads to drift in the colors over time (due to changes in the pose and the camera exposure and white balance of the input video over time). On the other hand, color-matching every frame leads to high-frequency flickering (due to sudden changes in scene content).

Instead, we propose a novel two-stage approach, shown in Fig. 3, that is designed to handle these color inconsistencies. In the first step, we compute per-frame transformations $\mathbf{T}_t$ (Sec. 4) that match the color palette of each input video frame to a corresponding *representative model frame* (Sec. 4.2). By estimating color transformations at every frame, we capture the changes in the scene content of both the input and model videos. Representative model frames summarize the important aspects of the content in the model video, and are chosen automatically so as to span the appearance space of the model. However, applying these transformations naively to the input video leads to temporal artifacts. To avoid this, in the second step (Sec. 5), we filter the per-frame transformations to make them temporally coherent. We achieve this filtering through a novel differential-geometry analysis of the series of color transformations $\mathbf{T}$ as a curve in a higher-dimensional transform space. We show that points at which this curve has a high curvature directly correspond to instants where directly applying the transformations $\mathbf{T}_t$ to the input video $\mathbf{I}_t$ would produce artifacts such as temporal flickering. Instead, we detect points on this curve that have a low curvature (called *keyframes*) and interpolate the transformations at these points

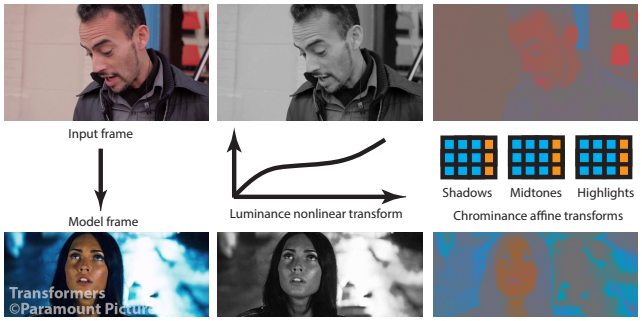


Figure 4: *Our set of transformations consists of a nonlinear luminance function and a set of affine chrominance transformations. These transformations can be computed for each segment of a segmented video if a segmentation is available. Video credits: "Transformers" (2007) ©Paramount Pictures (model).*

to build a set of temporally coherent color transformations $\mathbf{T}'$. By temporally smoothing out these color transformations, we filter out high-frequency noise while still adapting to low-frequency changes in the input video. Performing this analysis in a space of color transformations rather than at the pixel levels makes it computationally efficient since we only filter a small number of numerical values. Applying the interpolated transformations $\mathbf{T}'_t$ to the input video frames $\mathbf{I}_t$ produces the final color graded output video frames $\mathbf{O}_t$ (Fig. 2, bottom).

4 Single-frame Color Matching

Our method first computes per-frame color transformations $\mathbf{T}_t$ to match each input frame $\mathbf{I}_t$ to a desired image or model frame $\mathbf{M}_t$. In this section, we first describe how we model color transformations and then describe how we pair each input frame with a model frame and actually estimate a per-frame color transform.

4.1 Color Transfer Model

We seek to transfer the color distribution of a model video frame $\mathbf{M}_t$ to an input video frame $\mathbf{I}_t$. A color transfer method that is too flexible such as full color histogram matching would be highly dependent on the footage content, e.g., for an outdoor scene, the proportion of sky between the input and the model would need to be exactly the same for the transfer to be successful. Such sensitivity is undesirable in our context since we are unlikely to get such good input-model matches. On the other hand, a transfer model without enough flexibility such as matching only the mean color would not be expressive enough to handle the visual looks that we are interested in. We strike a balance by first transferring the luminance values using a smooth remapping curve and then matching the chrominance values using three affine transforms, one for each of shadows, midtones, and highlights (see Fig. 4). In cases where a segmentation into foreground and background objects is provided, we do this transformation for foreground and background, respectively, for a total of six transformations per frame. We finalize the per-frame transfer with an edge-aware smoothing filter that ensures clean edges with no halos. For the color space, we follow the recommendation of Reinhard and Pouli [2011] and work in CIE-Lab and the D65 illuminant. We describe the details of this model below.

Luminance We represent the tonal aspect of a frame by its intensity histogram. To transfer the tonal properties of the model onto the input, we use standard histogram matching, i.e., we use the transfer function $\mathbf{T}^\ell = H^{-1}(\mathbf{I})\big(H(\mathbf{M})\big)$, where H denotes the cumulative luminance histogram operator, respectively applied to $\mathbf{I}$ and $\mathbf{M}$. The generalized inverse [Villani 2003] (p.73) of H is used to account for non-invertible cumulative distribution functions. To prevent extreme histogram stretching in the cases where the two frames are widely different or are noisy (as in the case of low-quality video input), we regularize the transfer function by convolving it with a boundary-preserving Gaussian kernel of standard deviation $\sigma = 10\%$.

Chrominance Our chrominance model is inspired by standard color grading tools that let users control the color transform separately for the shadows, midtones, and highlights. To allow for this, we separate each frame into three luminance bands with an equal number of pixels. To avoid discontinuities, we make these bands overlap and smoothly transition from one to the other. We achieve this by making bands overlap by 10%, and setting pixels weights to 1 inside their corresponding band with a cubic fallof outside. Finally, the per-pixel weights across all the bands are normalized to 1.

For each band, we model chrominance properties by the weighted mean and covariance [Galassi et al. 2011] of the a and b chrominance channels – a 2x1 vector and a 2x2 matrix, respectively. Given these two properties, we estimate an affine transform from the model video frame to the input video frame. The translation component (a 2x1 vector) of this transform corresponds to the difference of the means. The mapping between the covariances (a 2x2 scaling matrix) is computed in closed form using the technique of Pitié and Kokaram [2007]. The advantage of this representation is that it is low-dimensional enough not to overly dependent on the scene content, while still being flexible enough to capture a wide range of effects, including color filters that are represented by the translation component and sophisticated saturation variations represented by a full 2x2 linear transformation in the ab plane.

Since the bands have smooth transitions, a pixel can be affected by more than one chrominance transforms. To handle this, we cross-fade the individual matrices using the pixel weights for each band to obtain the final pixel's transformation matrix. Fig. 5 illustrates the advantage of using our color transfer model.

Foreground-background segmentation Some color grades rely on a specific treatment of the foreground and the background. For instance, Fig. 1b shows an example of a orange-teal look that emphasizes the tension of the scene. Professional movie makers can achieve this effect by carefully setting up the shoot (props, costumes, lighting) and fine-tuning it in post-processing using global adjustments. However, amateur videos are taken under uncontrolled lighting environments and global edits cannot produce such foreground-vs-background effects (Fig. 6). We enable these visual styles by letting users specify a segmentation of the scene into foreground and background, and estimating the color transforms for these regions separately. In practice, we use the Snapcut tool [Bai et al. 2009] to create a mask for the foreground. To reduce segmentation artifacts, we further refine this binary mask to produce an alpha matte. For each frame, we erode and dilate the provided mask to determine an area of uncertainty around edges. This produces a tri-map that is used to build a soft mask for each frame [Levin et al. 2008]. We then process these masks with the edge-aware spatio-temporal filter of Lang et al. [Lang et al. 2012]. This creates a temporally consistent matte that we use to adjust the luminance band weights multiplicatively. We apply the color transform estimation technique described above to construct luminance and chrominance transforms separately for the foreground and background.

Spatial filtering Although refining the segmentation greatly contributes to having clean edges, halos may still appear in some cases (Fig. 7, left). We follow an approach inspired by Rabin et al. [2010] and Pouli and Reinhard [2011]. We compute the difference between

(a) Model video frame "Transformers" (b) Input video frame "Amélie" (c) Naive histogram matching (d) Our color transfer model

Figure 5: *We would like to transfer the color palette of the model video frame (a) to an input video frame (b). (c) The simplest way to do this would be to apply histogram matching in each color channel (and the foreground and background) independently. While this might match the color styles, it often produces artifacts (note, the color artifacts on Amélie's face). (d) Our color transfer model is able to produce an equivalent result without any artifacts. Video credits: "Transformers"(2007) ©Paramount Pictures (model), "Amélie"(2001) ©Miramax Films (input).*

(a) Model video frame "D.O.A" (b) Input video frame (c) Color transfer without segmentation (d) Color transfer with segmentation

Figure 6: *Some color styles have spatially-varying characteristics that cannot be replicated with global color adjustments. (a) For e.g., in this example of the Film Noir style, the face is bright, while the background is darker, but the input frame (b) does not match this. Transferring this look to the input video frame using global color adjustments does not replicate this style. (d) By using a user-specified segmentation (shown in a and b), we can produce a rendering that is more faithful to this style. Video credits: "D.O.A"(1950) (model), Tilke Judd (input).*

the original and transformed images, apply an edge-aware smoothing filter to this difference, and add this filtered difference back to the original image. However, we observed color bleeding when we processed all three color channels and found that processing only the luminance produced better results. In practice, we chose the Domain Transform filter [Gastal and Oliveira 2011] over the bilateral filter [Pouli and Reinhard 2011] and the Yaroslavsky filter [Rabin et al. 2010] because of its speed.

4.2 Representative model frames

Our approach supports several options for specifying the model data. Users can provide a single model image, or for more control, specify model images for several hand-picked input video frames. That said, we believe that most amateur users would find it burdensome to painstakingly specify these correspondences. Instead, we present a technique that only requires users to provide an input and a model video, and automatically pairs each input video frame with a model video frame.

First, to avoid doing a computationally expensive exhaustive search among all the model frames for each input frame, we summarize the model sequence with a few *representative frames* whose color distribution is representative of that of the entire video. We perform this clustering using the K-medoids clustering technique of Park and Jun [2009]. We chose K-medoids over standard K-means because medoids are constrained to be one of the input samples, which fulfills our goal of selecting frames from a video sequence. In practice, we run the K-medoids algorithm on the frames of the model video to produce one representative model frame for every 30 frames of the model video, i.e., one for every second of a model video sampled at 30fps. We refer to the original K-medoids article for the details of the algorithm. To compare frames, we use the metric proposed by Ferradans et al. [2012] that relies on the means, μ, and covariances, Σ, of the color distributions to estimate the distance $d(p,q)^2$ between frames t_p and t_q as:

$$\sum_k \mathrm{tr}(\Sigma_p^k + \Sigma_q^k) - 2(\Sigma_p^{k\frac{1}{2}}\,\Sigma_q^k\,\Sigma_p^{k\frac{1}{2}})^{\frac{1}{2}} + \|\mu_p^k - \mu_q^k\|^2, \qquad (1)$$

where the summation is over the three luminance bands. We remove outliers by ignoring medoids with less than 30 samples, which represent less than a second at 30fps. Given the representative frames, we match each input frame to the representative frames using the same metric, and estimate the corresponding color transformation. This process is illustrated in Fig. 8, where using an arbitrary model video frame produces a bad color transfer, but using our automatically estimated match improves the quality of the result substantially.

5 Differential Color Transform Filtering

After the previous stage, we have a color transform $\mathbf{T}_t$ for each frame. Each of these transformations has been computed in isolation and applying them directly to the video frames produces a temporally inconsistent result. For instance, in Fig. 2, the colors of the wall and of the person in the foreground change when a passer-by briefly walks across the background, even though this is an momentary event that ideally should not affect the result. One could apply a spatio-temporal filter to the video pixels to average this out but this would produce a smooth result in which high-frequency details are lost. We address this issue by temporally filtering the *color transforms* applied to video frames. This ensures that both the temporal coherence observed in the input sequence and the spatial details in the video frames are preserved. However, this poses a major challenge: the space of color transforms is high-dimensional, and extending the notion of a filter to this domain is nontrivial. An

(a) Input video frame (b) Color transfer without spatial filtering (c) Color transfer with spatial filtering

Figure 7: *Applying color transformations to the foreground and background independently can lead to artifacts. (b) This is illustrated for this example, where applying the Film Noir style (Fig. 6a) to the input video frame (a) produces haloing artifacts at segment boundaries. (c) Using an edge aware filter to smooth the difference improves the quality of the result. Video credits: Markus Wills (input).*

(a) Input video frame (b) Bad model frame "The Dark Knight" (c) Color transfer with bad model match (d) Good model frame "The Dark Knight" (e) Color transfer with good model match

Figure 8: *To ensure that the per-frame color transforms computed are meaningful, it is important that input video frames are matched to meaningful model video frames. When an input video frame (a) is color graded with a bad model frame (b), the results are often bad (c). Instead, in our work, we pre-cluster the model video into a small number of representative model frames and match each video frame to the most similar representative frame (d). This matching produces a meaningful color graded result (e). Video credits: Akash Viren Prasad (input), "The Dark Knight" (2008) ©Warner Bros. (model).*

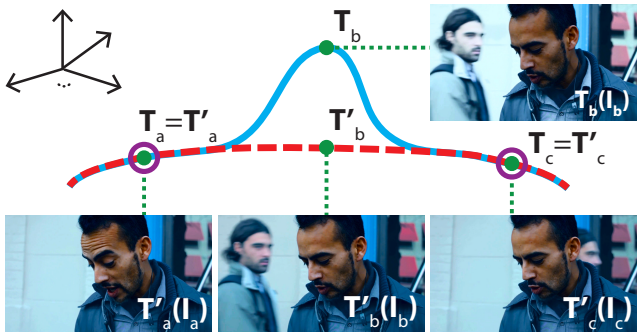


Figure 9: *The per-frame color transformations $\mathbf{T}_t$ can be analyzed as points on a curve in a high-dimensional space (blue curve, green dots). Points on this curve with a high curvature correspond to transformations that cause temporal artifacts in the output video (for e.g., the background in the top right image brightens when a person walks across the back). Our color transform filtering technique detects keyframes (purple circles) by sampling regions of low curvature and computes a smooth transform curve by interpolating the original transformations at the keyframes (red dashed curve, green dots). Applying the interpolated transforms to the input frames produces a temporally consistent result (bottom middle).*

important contribution of this work is to introduce a smoothing filter that operates on color transforms. Our approach relies on existing tools developed in the field of high-dimensional differential geometry. A comprehensive review of this field is beyond the scope of this paper. Below, we provide an intuitive presentation of the concepts involved as well as a short technical discussion. We refer the reader to the supplemental material for additional information and a C++ implementation. For a detailed presentation of differential geometry please see books such as do Carmo [1992] and Amari and Nagaoka [2000].

A curvature-based filter Our filter is inspired by the curvature flow filter defined on surfaces in 3D [Ilmanen 1995]. The observation behind this work is that adding noise to an otherwise smooth surface generates points with a high local curvature. The curvature-flow filter moves these points in a direction that reduces their local curvature, which effectively smooths the surface. In our context, we analyze the color transforms, $\mathbf{T}$ estimated in the previous section, as a 1D curve in the space of color transforms. Rapid changes of colors resulting from sudden changes in the scene (for e.g., when the passer-by enters the frame in Fig. 2) correspond to frames where the estimated transforms vary sharply, and consequently *the curvature of the transformation curve is high at these points*. Conversely, frames with a *low curvature* correspond to time instants when the transforms are stable making these frames good candidates to be keyframes for estimating a smooth color transformation curve. Our approach builds upon this observation. First, we extend the notion of curvature to the color transforms previously computed. Second, we find points of low curvature and select them as keyframes. Finally, we interpolate the color transforms at the detected keyframes to compute a temporally smooth set of color transformations for the entire sequence; this effectively replaces the high-curvature points that lie between the keyframes with transforms interpolated from the neighboring keyframes. Fig. 9 illustrates our approach.

5.1 Estimating the curvature

We would like to define a notion of curvature for the 1D color transformation curve, $\mathbf{T}$, in the space of color transformations, and this requires tools from differential geometry. Readers might be

more familiar with the definition of curvature in 2D or 3D; while the exact form of the analysis in the higher-dimensional color transform space differs from the corresponding analysis in 2D, the high-level intuition remains the same. The curvature of the color transform curve is defined as the rate of change of the tangent to the curve. In order to compute this quantity, we need to define the gradient of the color transform curve and a metric to estimate how fast the gradient varies.

A simple approach to implement our strategy is to work in a Euclidean space. This implies using a Euclidean metric, defining curvature as the magnitude of the second derivative of the color transform curve with respect to time, $||\ddot{\mathbf{T}}||$, and using linear interpolation to interpolate the color transforms between keyframes. However, linear interpolation does not accurately model the rotations of the color wheel that are commonly used for color grading and looses correlations between color channels. Therefore, we use a Euclidean space only for the translational components of the color transformations. We instead handle rotations and scalings via a Wasserstein space that accounts for their properties [Takatsu 2011] more accurately; for e.g., interpolation is modeled as a mass transport problem that blends rotations and scalings appropriately. A challenge when working in such a non-Euclidean space is that standard formulas for curvature need to be adapted. While the tangent to the curve is still defined as the first derivative $\dot{\mathbf{T}}$, the definition of its curvature is slightly more involved since it needs to account for the curvature of the space itself. That said, while the exact definitions are different, the high-level intuition about curvature and interpolation remains the same; interpolating transforms along a geodesic in this high-dimensional space produces a "flat trajectory", i.e., a zero curvature path in the space of transforms generates a temporally coherent output that is visually pleasing.

One notion of curvature of a curve is captured by the rate of change of tangent vectors along the curve, and is evaluated using the *covariant derivative*, $\nabla_{\dot{\mathbf{T}}}\dot{\mathbf{T}}$ [do Carmo 1992] that generalizes the notion of a directional derivative in Euclidean spaces to manifolds. Standard curvature flow techniques operate on a modified version of the covariant derivative – the *second fundamental form* [Ilmanen 1995] – but this does not account for all the variations we are interested in. For instance, a non-constant speed variation along a geodesic results in a vanishing second fundamental form, and hence, zero curvature. Instead, we compute the curvature at time instant t, K_t, as the magnitude of the covariant derivative vector at that time instant, i.e., $K_t = ||\nabla_{\dot{\mathbf{T}}}\dot{\mathbf{T}}||$.

To compute the curvature, we need to define the covariant derivative vector for the color transformation space. As described in Sec. 4.1, our color transformation model consists of a non-linear luminance mapping, and three rotations/scalings and translations for the chrominances. A meaningful metric in the infinite-dimensional space of nonlinear luminance transforms is out of reach. Instead, we approximate the 1D luminance mapping by a 1D affine transformation (i.e., a scale and a translation) only for the purpose of estimating the transform curvature. As a result, the color transform space is a finite-dimensional space of dimension 17 (or 34 when using a segmentation) that is a Cartesian product of the multiple subspaces corresponding to each transform component. These 17 dimensions consist of two 1D spaces (luminance scaling and translation), three 2D spaces (chrominance translations for shadows, midtones and highlights), and three 3D spaces (one 2×2 symmetric matrix representing the chrominance rotation and scaling for shadows, midtones, and highlights). We use the property that the squared curvature of the color transform curve in the high-dimensional space can be computed as the sum of the squared curvature for each individual component in its appropriate subspace [do Carmo 1992]. This allows us to define the appropriate formulas for the covariant derivative vector for each subspace independently.

Each sub-component of the color transforms can be represented by a matrix (for e.g., the luminance scaling and translations are 1×1 matrices, the chrominance translations are 2×1 matrices, and the chrominance rotations are 2×2 symmetric matrices). In the rest of this section, we consider the case of a single sub-component that is represented by a single matrix, and derive the form for its covariant derivative vector. In addition, the subpace of the luminance and chrominance translations is Euclidean, and the covariant derivative corresponds exactly to the standard second derivative of each component of the translation vector with respect to time. As such, we will focus on the luminance scaling and the chrominance rotation sub-components of the color transformations.

We first express the components of our matrix in a particular basis of dimension d that gives us a parametrization of the space of transformations. For instance, for the 2×2 symmetric chrominance rotation and scaling matrices, a natural basis consists of the basis matrices $\{\mathbf{x_1} = \left[\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right], \mathbf{x_2} = \left[\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right], \mathbf{x_3} = \left[\begin{smallmatrix}0 & 0\\ 0 & 1\end{smallmatrix}\right]\}$. Any 2×2 symmetric matrix can be decomposed as the weighted sum of these $d = 3$ basis matrices. We denote the i^{th} components of the matrix $\mathbf{T}_t$ in this basis as T^i $(i = 1 \dots d)$.

Obtaining the covariant derivative then requires a set of real coefficients called the *Christoffel symbols*, $\Gamma^i_{k,\ell}$, of the Wasserstein metric that account for the space curvature [Amari and Nagaoka 2000], where i, k and ℓ range from 1 to the dimension d of the space ($d = 1$, 2 or 3), defined in Eq. 3 and 4. Using these Christoffel symbols, we can compute the i^{th} component of the covariant derivative $\nabla_{\dot{\mathbf{T}}}\dot{\mathbf{T}}$ as:

$$\nabla^i_{\dot{\mathbf{T}}}\dot{\mathbf{T}} = \ddot{T}^i + \sum_{k,\ell} \Gamma^i_{k,\ell}\, \dot{T}^k\, \dot{T}^\ell, \quad i = 1 \dots d \tag{2}$$

where $\dot{T}^i$ and $\ddot{T}^i$ are the first and second derivatives of the i^{th} component of the transformations $\mathbf{T}$ in the basis $\mathbf{x_k}$ w.r.t. time and are approximated with standard finite differences.

In the supplemental material, we show that for the 1D luminance scaling:

$$\Gamma^1_{1,1} = -\frac{1}{2\,s}, \tag{3}$$

where s is the luminance scaling at the current point on the curve.

In the 3D space in which the chrominance rotations and scalings are represented, the Christoffel symbols are expressed by:

$$\Gamma^i_{k,\ell} = \frac{1}{2}\sum_{m=1}^{3} g^{im}\left(\frac{\partial g_{mk}}{\partial x^\ell} + \frac{\partial g_{m\ell}}{\partial x^k} - \frac{\partial g_{k\ell}}{\partial x^m}\right), \quad i,k,\ell = 1 \dots d, \tag{4}$$

where the partial derivatives can be seen as derivatives along the basis matrices $\mathbf{x_i}$. The coefficients g_{ij} are determined using Takatsu's metric [2011] – analogous to the Euclidean dot product of two vectors – as $g_{ij} = g(\mathbf{m_i}, \mathbf{m_j}) = \mathrm{tr}\,(\mathbf{m_i}\,\mathbf{T}_t\,\mathbf{m_j})$ where tr denotes the standard matrix trace, $\mathbf{T}_t$ the current transformation, and $\mathbf{m_i}$ (resp. $\mathbf{m_j}$) is a 2×2 matrix that solves the equation $\mathbf{x_i} = {}^{\mathsf{T}}\mathbf{m_i}\,\mathbf{T}_t + {}^{\mathsf{T}}\mathbf{T_t}\,\mathbf{m_i}$. Once the coefficients g_{ij} have been computed, they define a 3×3 matrix: inverting this matrix produces a new matrix whose coefficients are conventionally denoted by g^{ij}, as used in Eqn.4. More details on the terms introduced, and a closed form derivation of these coefficients and directional derivatives are provided in supplemental material, as well as in a sample C++ code.

The curvature at time t, K_t, is finally computed as the square root of the sum of the squared norms of the covariant derivatives in the individual subspaces. The squared norm in one subspace is computed as $g(\nabla_{\dot{\mathbf{T}}}\dot{\mathbf{T}}, \nabla_{\dot{\mathbf{T}}}\dot{\mathbf{T}})$.

5.2 Approximate curvature flow

Having computed a curvature value, K_t, for each frame of the input video, we use this information to drive a decimation process leading to smoother transformations over time. Our method can be regarded as an approximate curvature flow. Curvature flow is a mathematical method that advects a manifold locally with a speed proportional to its local curvature, resulting in a smoother manifold, and is often used to smooth 2D surfaces using the surface mean curvature (the trace of its second fundamental form). We mimic this behavior with an approximate, computationally simpler, method, that allows for keyframe detection. We reduce the total curvature K of our curve $\mathbf{T}$ by detecting segments of high curvature and replacing them by (smoother) geodesics. To achieve this, we subsample the curve by retaining transformations only in regions of low curvature, and interpolating the transformations in-between.

We first generate a set of r keyframes in areas of low curvature of the transform curve by sampling the probability function corresponding to the inverse of the curvature, i.e.,

$$p(t) = \sum_i K_i / K_t. \tag{5}$$

This can done using inversion sampling [Devroye 1986], i.e., by generating a uniform set of samples and warping these samples using the cumulative density function of the probability function $p(t)$. This produces more samples in areas of low curvature. We further locally optimize these samples by moving them to the location of lowest curvature in their neighborhood (15 frames of either side of the current sample). We use one sample for every 30 frames of the input video. This typically results in $r = 5$ for a 5 second video at 30fps.

Once the keyframes have been detected, we interpolate the individual sub-components of $\mathbf{T}$ between each keyframe to produce the desired smooth color transforms $\mathbf{T}'$. In particular, the symmetric positive definite matrices representing the chrominance scalings are interpolated using a closed-form interpolation that appropriately follows geodesics in this Wasserstein space [Ferradans et al. 2012]. The non-linear luminance mappings are linearly interpolated resulting in a displacement interpolation of the transformed luminance histograms [Bonneel et al. 2011]. Finally, the interpolated color transformations, $\mathbf{T}'$, are applied to the input video frames to produce the color graded output video frames.

6 Results

We demonstrate color transfer results on a wide range of input and model sequences. We show a subset of our results in Fig.10. However, temporal consistency can not be evaluated on still images and we encourage interested readers to refer to the supplemental material and video for additional video color transfers results.

A typical 4-second (108 frames) HD segmented sequence can be matched in approximately 3 minutes on an 8-core machine. This consists in 1min 30s for colorspace conversions, 20 seconds for the per-frame matching (first step), 6 milliseconds for the smoothing step and 21 seconds for the spatial luminance smoothing. Obtaining a spatio-temporally consistent matte from the user-provided binary segmentation requires 1.5 additional minutes using our unoptimized implementation of Lang et al. [2012] and 3 minutes for the authors' implementation of Levin et al. [2008].

While hand-tuned naive methods occasionally work, this is not always the case. For instance, Fig 2 shows that per-frame color matching produces artifacts when scene content is changing at high frequencies while a single-frame color matching applied to all frames produces artifacts even when scene content is varying at low frequencies. While our method does not always produce satisfactory results (see Sec. 7), we show a variety of successful transfers on a wide range of input and model sequences. In particular, our method is able to color grade videos that have already been stylized (see Fig. 11). We call this process "color re-grading". Despite the strong stylization of these input videos, our method is able to capture the color grade of the model video.

(a) Input (b) Model (c) Result

Figure 12: *Failure cases. (Top) The black and white initial frame makes the covariance matrices singular; no color information can be recovered from the model frame apart from the mean. (Bottom) Color grading using a non-relevant model frame produces a non-meaningful result. Video credits: "Saving Private Ryan" (1998) ©Paramount Pictures, "The Dark Knight" (2008) ©Warner Bros.*

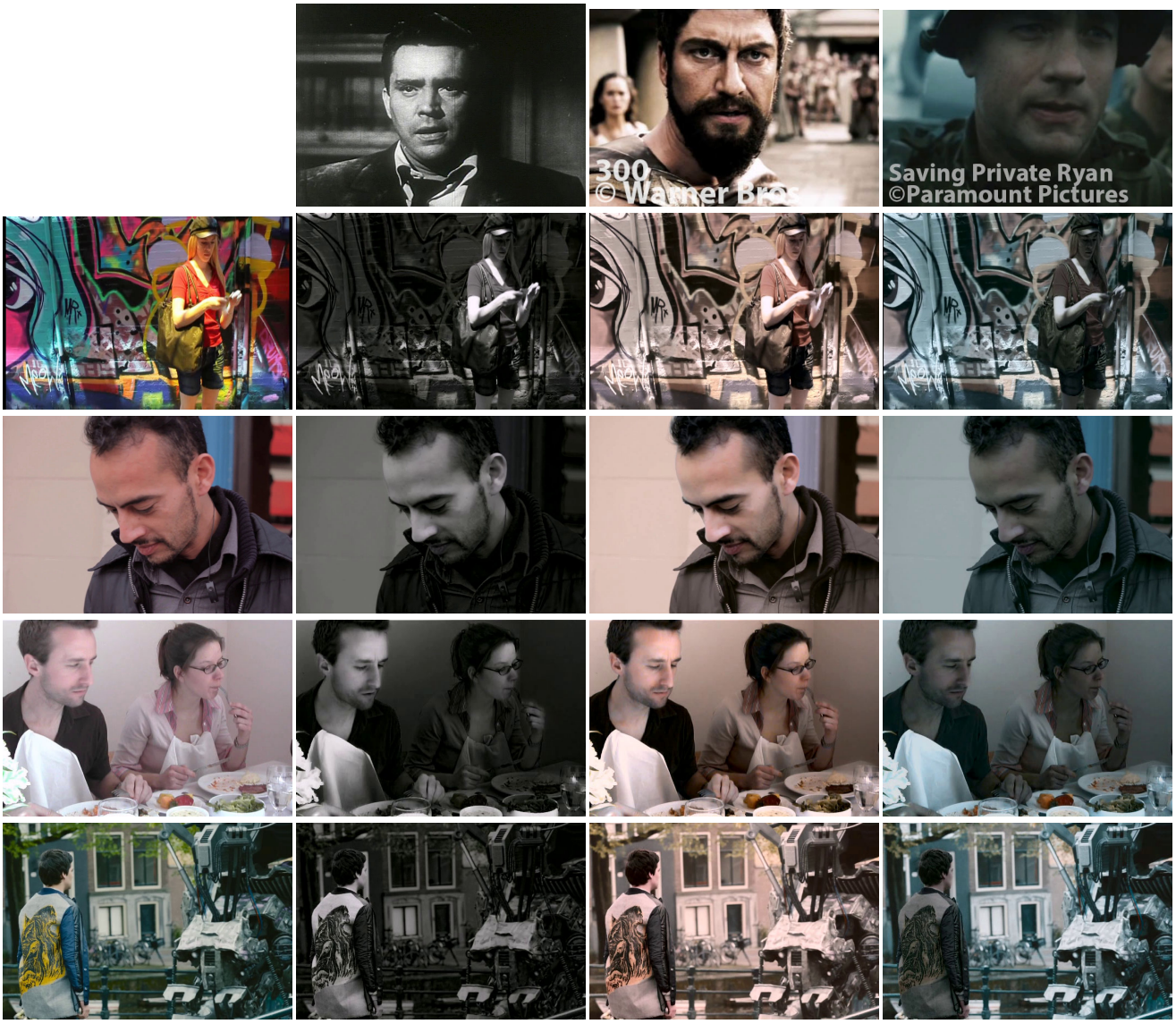
All our results used a single set of parameters and required little human intervention. Although a fine adjustment of our parameters can yield improved results, a fixed parameter set benefits amateur users.

7 Limitations and future work

The semi-automatic segmentation [Bai et al. 2009] that extracts foreground and background – when necessary – still requires significant user input. While out of the scope of our paper, our method would benefit from future work in fully automated video segmentation and matting. Similarly, our method would benefit from a multi-label video segmentation, that would allow for better capturing spatial variations in colors.

When either the chrominance matrices of the input videos are rank deficient or luminance histograms are concentrated on a few bins, our method can lead to visible artifacts. We show this behavior when trying to match a grayscale sequence to a colored one in Fig. 12 (first row). Such transfer cannot be achieved by our method: we refer the reader to video colorization techniques [Yatziv and Sapiro 2006] for such applications. In addition, if the content of the model sequence differs significantly from the input sequence, our method lacks the semantics that would be needed to yield a meaningful result. For instance, Fig. 12 (second row) shows that matching the white joker's face to a neutral face without makeup results in color variations on the resulting face that are not what would be expected from an appropriate color grade. In this case, the resulting sequence does not succeed in capturing the mood of the model.

Our curvature computation can be used to locally optimize user-defined keyframes. While the benefits of a local optimization remains to be demonstrated, it could lead to better user interfaces for processing video sequences. We also believe that our approximate curvature flow filtering technique could be used for a number of other applications in video processing including tonal stabilization.



(a) Input Video Frame (b) Color graded to "D.O.A" (Film Noir) (c) Color graded to "300" (sepia tones) ©Warner Bros (d) Color graded to "Saving Private Ryan" (bleach-bypass) ©Paramount Pictures

Figure 10: *Our color grading method can successfully transfer a variety of color styles to a range of input video sequences. Here we demonstrate this for three such styles – sepia tones, Film Noir, and bleach-bypass – characterized by the films "300", "D.O.A", and "Saving Private Ryan" respectively. In addition to capturing the color palette of the model videos, our color graded results are temporally consistent. Please refer to the accompanying video and supplemental material to evaluate the temporal consistency and to see more results. Video credits: Tilke Judd (third row), "Tears of Steel"/The Blender Foundation (fourth row), ""300" (2006) ©Warner Bros., "Saving Private Ryan" (1998) ©Paramount Pictures.*

(a) Color graded input video frame (b) Re-graded to "D.O.A" (c) Re-graded to "300" (d) Re-graded to "Amélie"

Figure 11: *Our technique can take stylized video clips and change their visual look. We call this process "color re-grading". Here we show results for re-grading (top to bottom) "Transformers" and "Saving Private Ryan" to the styles of the films (left to right) "D.O.A.", "300", and "Amélie". Our technique can handle transitions between subtle color grades (for e.g., "Saving Private Ryan"'to "300") as well as more dramatic color grades (for e.g., "Transformers" to "Amélie"). Please see the accompanying video and supplemental material for more results. Video credits: "Saving Private Ryan" (1998) ©Paramount Pictures, "Transformers" (2007) ©Paramount Pictures, "300" (2006) ©Warner Bros., "Amélie"(2001) ©Miramax Films.*

8 Conclusion

Color grading is a fundamental color management process that is used to alter or enhance the color palette of a movie to give it a characteristic visual look. We have presented a semi-automatic tool that allows even casual users to color grade amateur video clips using existing example videos or images. Our tool is able to match the color palette of the model videos while ensuring that the output color graded results are temporally consistent. To achieve tractable and principled temporal consistency, we have proposed an original technique that casts the temporal filtering as curve simplification problem using differential geometry techniques. While we inherit the limitations discussed in Sec. 7 from statistically-based image color transfer techniques, we show that reasonable results can be obtained on a wide range of videos. We believe that our method has the potential to benefit the increasingly large number of casual videos shared by amateurs on the internet while paving the way to a differential geometry analysis of colors in videos.

Acknowledgements

We thank Harvard Professor Siu-Cheong Lau for useful advice on curvature computation. We also thank PavelM/5pm, former contributor on Math.StackExchange, for detailed clarifications on Takatsu's paper. We thank the authors of all the videos used to demonstrate the techniques in this paper. We also thank the SIGGRAPH reviewers for their helpful comments. This work was partially supported by NSF CGV-1111415.

References

AMARI, S., AND NAGAOKA, H. 2000. *Methods of Information Geometry*, vol. 191 of *Translations of Mathematical monographs*. Oxford University Press.

AN, X., AND PELLACINI, F. 2010. User-controllable color transfer. *Computer Graphics Forum 29*, 2, 263–271.

BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2006) 25*, 3, 637 – 645.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video SnapCut: robust video object cutout using localized classifiers. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2009)*, 70:1–70:11.

BONNEEL, N., VAN DE PANNE, M., PARIS, S., AND HEIDRICH, W. 2011. Displacement interpolation using lagrangian mass transport. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH Asia 2011)*, 158:1–158:12.

CHANG, Y., SAITO, S., AND NAKAJIMA, M. 2007. Example-based color transformation of image and video using basic color categories. *Image Processing, IEEE Trans. on 16*, 2, 329–336.

DEVROYE, L. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag. Section 2.2. Inversion by numerical solution of F(X) = U.

DO CARMO, M. P. 1992. *Riemannian Geometry*. Springer, Jan.

FARBMAN, Z., AND LISCHINSKI, D. 2011. Tonal stabilization of video. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2011) 30*, 4, 89:1 – 89:9.

FERRADANS, S., XIA, G.-S., PEYRÉ, G., AND AUJOL, J.-F. 2012. Optimal transport mixing of gaussian texture models. Tech. rep., Preprint Hal-00662720.

GALASSI, M., DAVIS, J., GOUGH, B., JUNGMAN, G., BOOTH, M., AND ROSSI, F., 2011. GNU scientific library - reference manual, version 1.15, sec. 21.7 weighted samples.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2011. Domain transform for edge-aware image and video processing. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2011)*, 69:1–69:12.

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B., AND LISCHINSKI, D. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011) 30*, 4, 70:1–70:9.

ILMANEN, T. 1995. Lectures on mean curvature flow and related equations. In *Lecture Notes, ICTP, Trieste*.

JOHNSON, M. K., DALE, K., AVIDAN, S., PFISTER, H., FREEMAN, W. T., AND MATUSIK, W. 2011. CG2Real: improving the realism of computer generated images using a large collection of photographs. *IEEE Trans. on Visualization and Computer Graphics 17*, 9 (Sept.), 1273–1285.

KAGARLITSKY, S., MOSES, Y., AND HEL-OR, Y. 2009. Piecewise-consistent color mappings of images acquired under various conditions. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2311–2318.

KISER, C., REINHARD, E., TOCCI, M., AND TOCCI, N. 2012. Real time automated tone mapping system for HDR video. In *Proc. of the IEEE Int. Conference on Image Processing*, IEEE.

LANG, M., WANG, O., AYDIN, T., SMOLIC, A., AND GROSS, M. 2012. Practical temporal consistency for image-based graphics applications. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2012) 31*, 4 (July), 34:1–34:8.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2008. A closed-form solution to natural image matting. *IEEE Trans. on Pattern Analysis and Machine Intelligence 30*, 2 (Feb.), 228 –242.

MURRAY, N., SKAFF, S., MARCHESOTTI, L., AND PERRONNIN, F. 2011. Towards automatic concept transfer. In *Proc. of the Eurographics Symposium on Non-Photorealistic Animation and Rendering*, ACM, NPAR '11, 167–176.

OLDENBORG, M. 2006. *A comparison between techniques for color grading in games*. PhD thesis, University of Skövde, School of Humanities and Informatics.

OSKAM, T., HORNUNG, A., SUMNER, R. W., AND GROSS, M. 2012. Fast and stable color balancing for images and augmented reality. In *2nd Int. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 49 –56.

PARIS, S. 2008. Edge-preserving smoothing and mean-shift segmentation of video streams. In *Proc. of the 10th European Conference on Computer Vision*, ECCV '08, 460–473.

PARK, H.-S., AND JUN, C.-H. 2009. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications 36*, 2, Part 2 (Mar.), 3336–3341.

PITIÉ, F., AND KOKARAM, A. C. 2007. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. In *4th European Conference on Visual Media Production, 2007. IETCVMP*, 1 –9.

PITIÉ, F., KOKARAM, A. C., AND DAHYOT, R. 2005. N-dimensional probablility density function transfer and its application to colour transfer. In *Proceedings of the Tenth IEEE Int. Conf. on Computer Vision - Vol. 2*, ICCV '05, 1434–1439.

POULI, T., AND REINHARD, E. 2011. Progressive color transfer for images of arbitrary dynamic range. *Computers & Graphics 35*, 1 (Feb.), 67–80.

RABIN, J., DELON, J., AND GOUSSEAU, Y. 2010. Regularization of transportation maps for color and contrast transfer. In *2010 17th IEEE Int. Conf. on Image Processing (ICIP)*, 1933 –1936.

REINHARD, E., AND POULI, T. 2011. Colour spaces for colour transfer. In *Proceedings of the Third international conference on Computational color imaging*, Springer-Verlag, Berlin, Heidelberg, CCIW'11, 1–15.

REINHARD, E., ASHIKHMIN, M., GOOCH, B., AND SHIRLEY, P. 2001. Color transfer between images. *IEEE Comput. Graph. Appl. 21*, 5 (Sept.), 34–41.

SELAN, J. 2012. Cinematic color: From your monitor to the big screen. *Visual Effects Soc. Tech. Committee White Paper* (Oct.).

TAI, Y.-W., JIA, J., AND TANG, C.-K. 2005. Local color transfer via probabilistic segmentation by expectation-maximization. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 747 – 754 vol. 1.

TAKATSU, A. 2011. Wasserstein geometry of gaussian measures. *Osaka Journal of Mathematics 48*, 4 (Dec.), 1005–1026. Mathematical Reviews number (MathSciNet): MR2648273.

VILLANI, C. 2003. *Topics in Optimal Transportation*. Graduate Studies in Mathematics Series. Amer Mathematical Society.

WANG, B., YU, Y., AND XU, Y.-Q. 2011. Example-based image color and tone style enhancement. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2011)*, 64:1–64:12.

XUE, S., AGARWALA, A., DORSEY, J., AND RUSHMEIER, H. E. 2012. Understanding and improving the realism of image composites. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2012) 31*, 4, 84:1–84:10.

YATZIV, L., AND SAPIRO, G. 2006. Fast image and video colorization using chrominance blending. *IEEE Trans. on Image Processing 15*, 5 (May), 1120 –1129.