J Math Imaging Vis (2017) 57:225–238
DOI 10.1007/s10851-016-0677-1

Temporally Consistent Tone Mapping of Images and Video Using Optimal K-means Clustering

Magnus Oskarsson[1]

Received: 15 March 2016 / Accepted: 7 July 2016 / Published online: 21 July 2016

Abstract The field of high dynamic range imaging addresses the problem of capturing and displaying the large range of luminance levels found in the world, using devices with limited dynamic range. In this paper we present a novel tone mapping algorithm that is based on K-means clustering. Using dynamic programming we are able to not only solve the clustering problem efficiently, but also find the global optimum. Our algorithm runs in $O(N^2K)$ for an image with N input luminance levels and K output levels. We show that our algorithm gives comparable results to state-of-the-art tone mapping algorithms, but with the additional large benefit of a minimum of parameters. We show how to extend the method to handle video input. We test our algorithm on a number of standard high dynamic range images and video sequences and give qualitative and quantitative comparisons to a number of state-of-the-art tone mapping algorithms.

Keywords High dynamic range images · High dynamic range video · Clustering · Dynamic programming

1 Introduction

The human visual system can handle massively different levels in input brightness. This is necessary to cope with the large range of luminance levels that appear around us—for us to be able to navigate and operate in dim night light as well as in bright sun light. The field of high dynamic range (HDR) imaging tries to address the problem of capturing and displaying these large ranges using devices (cameras and displays) with limited dynamic range. During the last years, the HDR field has grown, and today, many camera devices have built in functionality for acquiring HDR images. This can be done in hardware using sensors with pixels that can capture very large differences in dynamic range. It can also be done by taking several low dynamic range (LDR) images at different exposures and then combining them using software [10,19,45]. Recently the capability of recording HDR video data has arisen. One important part when working with HDR data is the ability to visualize it on LDR displays. The process of transferring an HDR image to an LDR image is known as *tone mapping*. Depending on the application the role of the tone mapping operator (TMO) can be different, but in most applications the ability to capture both detail in darker areas and very bright ones is important. Tone mapping can also be an important component in image enhancement for, e.g., images taken under poor lighting [34,59].

In [15] a number of criteria that an ideal TMO should exhibit are listed, namely

1. Temporal model free from artifacts such as flickering, ghosting and disturbing (too noticeable) temporal color changes.
2. Local processing to achieve sufficient dynamic range compression in all circumstances while maintaining a good level of detail and contrast.
3. Efficient algorithms, since large amount of data need processing, and turnaround times should be kept as short as possible.
4. No need for parameter tuning.
5. Calibration of input data should be kept to a minimum, e.g., without the need of considering scaling of data.
6. Capability of generating high-quality results for a wide range of video inputs with highly different characteristics.
7. Explicit treatment of noise and color.

✉ Magnus Oskarsson
magnuso@maths.lth.se

[1] Lund University, PO Box 118, 221 00 Lund, Sweden

We will in this paper present a new framework for doing automatic tone mapping of HDR image and video data. Our procedure addresses all criteria except 2 above. Our approach is spatially global but temporally local, to ensure temporal consistency. Even though we do not do any local processing in the image domain, we believe that our approach gives a high level of contrast. Depending on how the HDR data were constructed or recorded, there may be a need for noise reduction of the data. We do not consider this aspect in this paper, but concentrate solely on the tone mapping.

We will look at the tone mapping problem as a clustering problem. If we are given an input image with large dynamic range, i.e., with a large range of intensity values, we want to map these intensity values to a much smaller range. We can describe this problem as a clustering of the input intensity levels into a smaller set of output levels. In our setting we are looking at an HDR input with a very large input discretization, and performing clustering in three dimensions is not tractable. Iterative local algorithms will inevitably lead to local minima. Instead we work with only the luminance channel, and we show how we can find the global optimum using dynamic programming. This leads to a very efficient and stable tone mapping algorithm. We call our algorithm democratic tone mapping since all input pixels get to vote on which output levels we should use. The method has a very natural extension to handle HDR video input. The material presented here is partly based on the conference paper [38].

1.1 Related Work

A large number of tone mapping algorithms have been proposed over the years. Some of the first work include [53,58]. One can divide the tone mapping algorithms into global algorithms, that apply a global transformation on the pixel intensities, and local algorithms, where the transformation also depends on the spatial structure in the image. For a discussion on the differences see [60]. The global algorithms include simply applying some fixed function such as a logarithm or a power function. In [12] the authors present a method that adapts a logarithmic function to mimic the human visual system's response to HDR input. In [27] the image histogram is used and a variant of histogram equalization is applied but with additional properties based on ideas from human perception. Using histogram equalization will often lead to not efficiently using the colorspace, due to discretization effects.

The local algorithms usually apply some form of local filtering to be able to increase contrast locally. This often comes at the cost of higher computational complexity and can lead to strange artifacts. In [13] the authors use bilateral filtering to steer the local tone mapping. In [37] a perceptual model is used to steer the contrast mapping, which is performed in the gradient domain. In [35] the authors address the problem of designing display-dependent tone mappings. In [44] the authors propose an automatic version of the zone system developed by Ansel Adams for conventional photographic printing. The method also includes local filtering based on the photographic procedure of dodging and burning. The global part of their method is in spirit similar to our approach. In [29] they use K-means to cluster the image into regions and then apply individual gamma correction to each segment. For general overview of tone mapping we refer to the book [43].

Tone mapping is not only important for images, but also for HDR video. This field has received increasing interest during the last years, much due to the fact that HDR video data are gaining popularity [22,25,40,52]. Some of the earliest extensions to video simply apply TMOs image wise, adding temporal filtering to avoid flickering [22,35,41]. These simple operators can be targeted at real-time processing [23]. A number of operators are based on ideas mimicking the human visual system, some of which are global, e.g., [16,21,39,55] and some local, e.g., [5,28]. Some recent methods use more involved local processing and filtering schemes to achieve high local contrast and also reduce noise [2,4,6,7,14]. For overview and evaluation of video tone mapping operators see [15,40].

In this paper, we address the problem of tone mapping as a clustering problem. This is not an entirely new idea in the realm of quantization of images. The idea of clustering color values was popular during the 1980s and 1990s, when the displays had very low dynamic range, and the object was to take an ordinary 24-bit color image and map it to a smaller palette of colors that could be displayed on the screen. In this case there have been a number of algorithms that use variants of K-means clustering, see [8,47,48]. Here the clustering was done on three-dimensional input, i.e., color values. The algorithms used variants of the standard K-means [31] to avoid local minima. The number of input points was quite small, and the number of output classes, i.e., the palette, was relatively small so these methods worked well, but (as shown in Sect. 6.2) they are prone to get stuck in local minima, when the size of the problem increases.

2 Problem Formulation

We will begin by formulating our clustering problem for a luminance input image. Then, we will describe how this can be applied to RGB images and video inputs in the following sections. Let us consider the following problem. We are given an input gray value image, $I(x, y)$, with a large number (N) intensity levels. We would like to find an approximate image $\hat{I}(x, y)$ with a smaller number (K) intensity levels, i.e., we would like to solve

$$\min_{c_1,c_2,\ldots,c_K} ||I - \hat{I}||_2, \tag{1}$$

where

$$I(x, y) \in \{u_1, u_2, \ldots, u_N\} \tag{2}$$

and

$$\hat{I}(x, y) \in \{c_1, c_2, \ldots, c_K\}. \tag{3}$$

If we calculate the histogram corresponding to the input image's distribution, we can reformulate the problem as:

Problem 1 (*K-means clustering tone mapping*) Given $K \in \mathbb{Z}^+$ and a number of gray values $u_i \in \mathbb{R}$ with a corresponding distribution histogram $h(i)$, $i = 1, \ldots, N$, the K-means tone mapping problem is finding the K points $c_l \in \mathbb{R}$ that solve:

$$D(N, K) = \min_{c_1,c_2,\ldots,c_K} \sum_{i=0}^{N} h(i) d(u_i, c_1, \ldots, c_K)^2, \tag{4}$$

where

$$d(u_i, c_1, \ldots, c_K) = \min_l |u_i - c_l|. \tag{5}$$

This is a weighted K-means clustering problem. One usually solves it using some form of iterative scheme that converges to a local minimum. A classic way of solving it is alternating between estimating the cluster centers c_l and the assignment of points u_i to clusters. If we have assigned n points u_i to a cluster l, then the best estimate of c_l is the weighted mean

$$c_{\{1,\ldots,n\}} = \frac{\sum_{i=1}^{n} h(i) u_i}{\sum_{i=1}^{n} h(i)}. \tag{6}$$

For ease of notation we will henceforth use the notation c_l for cluster number l or $c_{\{1,\ldots,n\}}$ for the cluster corresponding to points $\{u_1, \ldots, u_n\}$. The contribution of this cluster to the error function (4) is then equal to:

$$f(u_1, \ldots, u_n) = \sum_{i=1}^{n} h(i)(u_i - c_{\{1,\ldots,n\}})^2. \tag{7}$$

The assignment that minimizes (4) given the cluster centers c_l is simply taking the nearest c_l for each point u_i. One can keep on iteratively alternating between assigning points to clusters and updating the cluster centers according to (6). It can easily be shown that this alternating scheme converges to a local minimum, but there are no guarantees that this is a global minimum. In fact for most problems, it is highly dependent on the initialization. There are numerous ways of initializing. See [50] for an extensive review of K-means clustering methods.

The K-means clustering problem is in general NP-hard, for most dimensions, sizes of input and number of clusters, see [1,9,33,57] for details. However, since the points we are working with are one-dimensional, i.e., $u_i \in \mathbb{R}$, we can actually find the global minimum of problem 1 using dynamic programming. This is what makes our method tractable. In the next section we will describe the details of our approach. We will in Sect. 4 describe how we use our solver to construct a tone mapping method for color images, and in Sect. 5 how we extend it to video input.

3 A Dynamic Programming Scheme

Problem 1 is a weighted K-means clustering problem, with data points in $\mathbb{R}$. We will now show how we can devise a dynamic programming scheme that accurately and quickly gives the minimum solution to our problem. For details on dynamic programming see, e.g., [24].

We use an approach similar to [3,57] and modify it to fit our weighted K-means problem. We will now show the recurrence relation for our problem:

Theorem 1 *For any $n > 1$ and $k > 1$ we have*

$$D(n, k) = \min_{k \le i \le n} (D(i - 1, k - 1) + f(u_i, \ldots, u_n)). \tag{8}$$

Proof Since our data points are one-dimensional, we can sort them in ascending order. Assume that we have obtained a solution $D(n, k)$ to (4) and let u_i be the smallest point that belongs to cluster k. Then it is clear that $D(i - 1, k - 1)$ is the optimal solution for the first $i - 1$ points clustered into $k - 1$ sets. □

Equation (8) defines the Bellman equation for our dynamic programming scheme and gives us our tools to solve problem 1. We iteratively solve $D(n, k)$ using (8) and store the results in an $N \times K$ matrix. The initial values for $n = 1$ or $k = 1$ are given by the trivial solutions. We can read out the optimal solution to our original problem at position (N, K) in the matrix. The clustering and the cluster centers of the optimal solution are then found by backtracking in the matrix. In order to efficiently calculate $D(n, k)$ we need to be able to iteratively update the function f from (7). We start by calculating the cumulative distribution $H(i)$ of $h(i)$, given by

$$H(i) = \sum_{j=1}^{i} h(j), \quad i = 1, \ldots, N. \tag{9}$$

Algorithm 1 K-means clustering using dynamic programming

1: Given input points $\{u_1, \ldots, u_N\}$, a distribution $h(i)$, $i = 1, \ldots, N$ and K.
2: Iteratively solve $D(n, k)$ using (8) and (12) for $n = 2, \ldots, N$ and $k = 2, \ldots, K$.
3: Find the centers c_l, $l = 1, \ldots, K$ and the clustering by backtracking from the optimal solution $D(N, K)$.

Theorem 2 *For a point set $\{u_1, \ldots, u_n\}$ the error contribution of those points can be updated by*

$$f(u_1, \ldots, u_n) = \sum_{i=1}^{n} h(i)(u_i - c_{\{1,\ldots,n\}})^2 \tag{10}$$

$$= \sum_{i=1}^{n-1} h(i)(u_i - c_{\{1,\ldots,n-1\}})^2 \tag{11}$$

$$+ \frac{h(n)H(n-1)}{H(n)}(u_n - c_{\{1,\ldots,n-1\}})^2, \tag{12}$$

where the weighted mean of the point set $\{u_1, \ldots, u_n\}$ is updated by:

$$c_{\{1,\ldots,n\}} = \frac{h(n)u_n + H(n-1) \cdot c_{\{1,\ldots,n-1\}}}{H(n)}. \tag{13}$$

Proof We can, without loss of generality, assume that we have transformed the coordinates so that $c_{\{1,\ldots,n-1\}} = 0$ and hence $\sum_{i=1}^{n-1} h(i)u_i = 0$. This gives according to (13):

$$c_{\{1,\ldots,n\}} = h(n)u_n/H(n), \tag{14}$$

giving us

$$f(n) = \sum_{i=1}^{n} h(i)(u_i - c_{\{1,\ldots,n\}})^2 \tag{15}$$

$$= \sum_{i=1}^{n} h(i)\left(u_i - \frac{h(n)u_n}{H(n)}\right)^2. \tag{16}$$

We can write this as

$$f(n) = f(n-1) + h(n)\left(u_n - \frac{h(n)u_n}{H(n)}\right)^2 \tag{17}$$

$$= f(n-1) + h(n)u_n^2 \frac{H(n-1)^2}{H(n)^2}. \tag{18}$$

The first part can be simplified as

$$f(n-1) = \sum_{i=1}^{n-1} h(i)\left(u_i - \frac{h(n)u_n}{H(n)}\right)^2 \tag{19}$$

$$= \sum_{i=1}^{n-1} h(i)\left(u_i^2 - 2u_i\frac{h(n)u_n}{H(n)} + \frac{h(n)^2u_n^2}{H(n)^2}\right) \tag{20}$$

$$= \sum_{i=1}^{n-1} h(i)u_i^2 + \frac{H(n-1)h(n)^2u_n^2}{H(n)^2}. \tag{21}$$

Combining (17) and (19) then yields

$$f(n) = \sum_{i=1}^{n-1} h(i)u_i^2 + \frac{H(n-1)h(n)^2u_n^2 + h(n)u_n^2H(n-1)^2}{H(n)^2} \tag{22}$$

$$= \sum_{i=1}^{n-1} h(i)u_i^2 + \frac{H(n-1)h(n)u_n^2}{H(n)}\frac{(h(n) + H(n-1))}{H(n)} \tag{23}$$

$$= \sum_{i=1}^{n-1} h(i)u_i^2 + \frac{h(n)H(n-1)}{H(n)}u_n^2. \tag{24}$$

□

Without using (12) each entry $D(n, k)$ would take n^2 iterations to calculate, and the total complexity would become $N \cdot K \cdot N^2 = N^3K$. However using (12) we can compute $f(u_1, \ldots, u_n)$ in constant time, and this gives a total complexity of N^2K.

4 Tone Mapping of HDR Color Images

The discussion in the previous section was concerned with grayscale images. In this section we will describe the whole algorithm for an HDR color input image. We assume an RGB input image. Algorithm 1 is based on that the input points u_i are one-dimensional. There are a number of ways in which one could apply the clustering on a color image, including working in numerous different color space representations. We will here describe two fast, efficient and color-preserving methods. They are both based on running Algorithm 1 on the luminance channel.

We start by estimating the intensity channel I_{gr} from the RGB image. One could use several estimates of the intensity, such as estimating the luminance using a standard weighted average. We have found however that our method performs best when we use the maximum of the three channels as our intensity. We then do a preprocessing step by taking the logarithm of I_{gr}, giving us $I_{\log}$. We can then run Algorithm 1. When we have clustered the intensity channel

into the desired K levels, we calculate our transfer function $F(s) : \{u_1, \ldots, u_N\} \rightarrow \{1, \ldots, K\}$ by finding the nearest neighbor of each input level s:

$$F(s) = \underset{l}{arg\,min}\ |c_l - s|. \tag{25}$$

The output image I_{out} is then constructed in one of two ways,

$$I_{\text{out}}^{\text{ch}} = F\left(I_{\text{log}}^{\text{ch}}\right), \tag{26}$$

or

$$I_{\text{out}}^{\text{ch}} = \frac{I^{\text{ch}}}{I_{\text{gr}}} \cdot F(I_{\text{gr}}), \tag{27}$$

where ch denotes the color channel, i.e. , R, G or B. Equation (26) simply applies the function F on the whole RGB image pixel-wise, whereas (27) applies the function F on the intensity channel, and then each color channel is given by multiplying with the color weight of each pixel. Using (27) corresponds to the way that was proposed in [49]. The different steps are summarized in Algorithm 2. Using equation (26) usually gives very good results, but can in some cases give somewhat subdued colors. Using equation (27) on the other hand will in general give more saturated colors, but can in some cases give exaggerated colors. This has been noted before, and in [54], the suggestion was to instead use

$$I_{\text{out}}^{\text{ch}} = \left(\frac{I^{\text{ch}}}{I_{\text{gr}}}\right)^{q} \cdot F(I_{\text{gr}}), \tag{28}$$

where q controls color saturation. It can actually be shown that (26) actually corresponds closely to (28) for a special choice of q, [36].

Algorithm 2 Democratic Tone Mapping (DTM)

1: Given a high bit color input image: I_{in}
2: Calculate the intensity channel I_{gr} of I_{in} .
3: Take the log to get $I_{log} = \log I_{\text{gr}}$
4: Calculate the histogram $h(s)$ of I_{log}.
5: Find the centers c_l using Algorithm 1.
6: Estimate $F(s) : u_i \rightarrow c_l$ using nearest neighbors.
7: $I_{\text{out}}^{\text{ch}} = F\left(I_{\text{log}}^{\text{ch}}\right)$ or $I_{\text{out}}^{\text{ch}} = \frac{I^{\text{ch}}}{I_{\text{gr}}} \cdot F(I_{\text{gr}})$.

In Fig. 1 the two extreme cases are exemplified for two test images. The top shows the result of running Algorithm 2 using (26), and the middle row shows the result of using (27). For the left image the colors are better reproduced in the middle image, but for the right image the top row is better. A simple way of controlling the color is taking a weighted average of the two outputs. This is shown in the bottom of Fig. 1. For the example images that we have tested, we have found that this gives a good trade-off. In [36] a different choice of color correction model was suggested, namely

Fig. 1 The figure shows the different color outputs for two example images. *Top* shows the result of running Algorithm 2 using (26), the *middle* shows the result of using (27), and the *bottom row* shows the output using a linear combination of the two (in this case $0.7 \cdot (26) + 0.3 \cdot (27)$). For the *left* image the colors are better reproduced in the *middle row*, but the *right* image is better in the *top row*. The *bottom row* shows the linear combination, which gives a very good compromise for the example images that we have tested

$$I_{\text{out}}^{\text{ch}} = \left(\left(\frac{I^{\text{ch}}}{I_{\text{gr}}} - 1\right) q + 1\right) \cdot F(I_{\text{gr}}), \tag{29}$$

where again q controls the color saturation. They further suggest choosing q as a certain sigmoid function of the contrast factor of the tone mapping function. We have tried various versions of this, but found that they give over-saturated colors for our tone mapping function. In Fig. 2 example outputs are shown. Here the luminance preserving sigmoid function was chosen, and the slope of the tone mapping function was used as contrast factor (see [36] for details). Since our tone mapping function is piece-wise constant, the slope is zero everywhere. But a natural choice of slope is to use the derivative of a continuous piece-wise linear function as approximation.

5 Tone Mapping of HDR Video

The tone mapping procedure described in the previous sections can easily be extended to efficiently handle video input. The most basic approach is to run Algorithm 2 on every frame. We will modify this approach in two ways, to give an efficient and temporally coherent output. First of all one can

Fig. 2 The figure shows the different color outputs for two example images using the color model (29). The luminance preserving sigmoid function was chosen, and the slope of the tone mapping function was used as contrast factor (see [36] for details). The *right* image suffers from over-saturation of colors

notice that the most time-consuming step of Algorithm 2 is running the dynamic programming. The complexity of this depends on the number of input bins and output bins, but not on the image size since it uses the *distribution* of gray values. This means that it would be just as costly to run the dynamic programming on one frame as it would be to run it on the whole sequence. Running it on only one frame will in many cases lead to unrobust behavior. Using it on the whole sequence will on the other hand lead to not using the maximum range for individual frames. We suggest using the distribution of a small number of neighboring frames for each frame.

For most scenes the gray value distribution will not change dramatically within a couple of frames. If running times are a priority, we propose the use of key frames, where the tone mapping function is estimated for each key frame using the dynamic programming scheme, and for intermediate frames the tone mapping function is interpolated linearly between the two nearest key frames.

Key frames can be chosen either with fixed intervals or when the gray value distributions differ significantly. A suitable metric for determining the difference is the Wasserstein metric or the Earth mover's distance (EMD), see [30,56]. For two finite discrete one-dimensional distributions, represented by their histograms h_1 and h_2, the EMD distance d can be calculated in closed form. If H_1 and H_2 are the cumulative distributions of h_1 and h_2, then

$$d = \sum_{i=1}^{N} |H_1(i) - H_2(i)|, \tag{30}$$

where N is the number of bins in the histograms.

The different steps are summarized in Algorithm 3. Note that Algorithm 3 is described for an offline scenario, but it could just as easily be run as an online algorithm for a real-time application, where the key frames are estimated online. In the experiments in the paper, we used a fixed distance of $\Delta t = 20$ frames between key frames. We further used $n = 1$ (i.e. , the key frame histogram is based on three frames). We need to estimate the two neighboring key frames to linearly interpolate between them. This means that we get a lag of $20 + 1 = 21$ frames, which for many applications is not a problem.

Algorithm 3 Democratic Tone Mapping of Video (DTMV)

1: Given a high bit color input video: $I_{in}(x, y, t)$
2: Determine a number of key frames at times $t = a_k$ (either with fixed difference $a_{k+1} - a_k = \Delta t$ or using the EMD distance (30).
3: For each key frame calculate the transfer function $F_k(s)$ using Algorithm 2, based on the gray value distribution of a set of $2n + 1$ neighboring frames $\{I_{in}(x, y, a_k - n), \ldots, \{I_{in}(x, y, a_k + n)\}$.
4: For each frame $I_{\text{out}}(x, y, t)$, linearly interpolate the transfer function $F_t(s)$ using the two nearest key frames, k and $k + 1$,

$$F_t(s) = w_k F_k(s) + w_{k+1} F_{k+1},$$

with $w_k = (t - a_k)/(a_{k+1} - a_k)$ and $w_{k+1} = 1 - w_k$.
5: The output frame is computed in the same way as in Algorithm 2,

$$I_{\text{out}}^{\text{ch}} = F_t(I_{\text{log}}^{\text{ch}}) \text{ or } I_{\text{out}}^{\text{ch}} = I^{\text{ch}} \cdot \frac{\exp(F_t(I_{\text{gr}}))}{\exp(I_{\text{gr}}))}.$$

6 Results

We have implemented our tone mapping algorithm and conducted a number of tests. First we show in Sect. 6.1 that our method gives a substantially better optimum compared to standard K-means clustering. In Sect. 6.2 we study the time complexity of our algorithm, and then in Sect. 6.3, we show results on a set of standard HDR images and compare with a number of different tone mapping algorithms. We have consistently used $K = 256$ in our experiments, corresponding to 8-bit output, but this could be set to any output quantization you like. In Sect. 6.4 we test our algorithms on HDR video input.

6.1 Comparison with Standard Iterative K-means

A standard iterative K-means algorithm will converge to a local minimum. Algorithm 1 will converge to the global minimum, but one may ask how often the local iterative scheme gets stuck in a local minimum, and how far this is from the global optimum. In order to investigate this we did some simple qualitative experiments where we ran Algorithm 1 on an input image (with 5000 gray levels). Our hypothesis is that for larger K the ordinary K-means will get stuck in local optima. To check this, we then ran the standard iterative K-means clustering and compared the resulting solution to the global optimum. We repeated this for a large number of runs with random initialization. We tried this for a smaller K (= 5) and a larger K (= 256). In Fig. 3 the results are shown. It shows at the top, histograms over the l^2 differences between the local solution for the cluster centers and the true solution. The center points were of course sorted before the norm was taken. Also shown are plots of the resulting error

energy (4) for the different runs, with the global optimum also plotted. The figures clearly show that in this case K-means finds the global optimum for the smaller K, but there was a large difference between the local solutions and the true solution for the large K, and in no case was the true optimum found using the local iterative method (right of Fig. 3). To further investigate the solutions we constructed silhouette plots [46] that can be seen in Fig. 4. For the small K there is little difference, but for the large K the optimal solution has a more even silhouette than the example run, based on standard K-means. For visibility purposes only five random clusters out of the 256 are shown. In Table 1, statistics from the silhouettes are shown. The table shows the mean and standard deviation of the silhouettes, for the optimal method compared to three runs of standard K-means. Again one can see that there is little difference for $K = 5$, but for $K = 256$, the optimal algorithm achieves less spread than standard K-means.

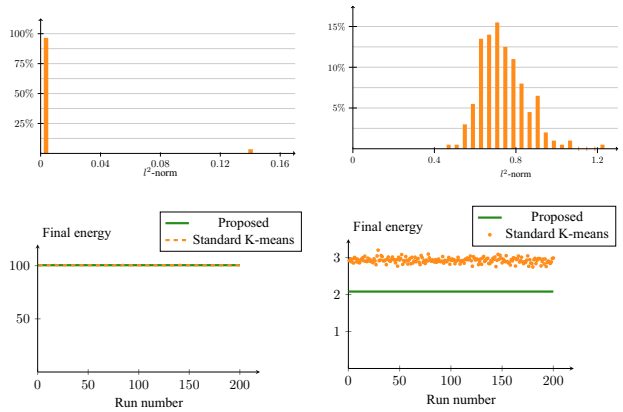

Fig. 3 Comparison between standard K-means and our optimal approach using $K = 5$ clusters (*Left*) and $K = 256$ clusters (*Right*). *Top* shows histograms over the l^2-norms of the differences between the global optimum and local optimums (based on 200 random initializations of standard K-means.) *Bottom* shows the final energy of the 200 solutions, with also the global optimum depicted in solid green. One can see that for $K = 5$ we get very similar results, and the local optimization leads to solutions close to the global one in most cases. For $K = 256$ on the other hand, the local optimization gives solutions far from the optimum, both in terms of the actual solution and the final error

Table 1 Statistics from the silhouette plots

Method	$K = 5$		$K = 256$	
	Mean	Std	Mean	Std
Optimal	0.584	0.0694	0.579	0.0601
Standard K-means #1	0.584	0.0708	0.576	0.1297
Standard K-means #2	0.584	0.0708	0.570	0.1063
Standard K-means #3	0.584	0.0708	0.572	0.1147

The table shows the mean and standard deviation of the silhouettes, for the optimal method compared to three runs of standard K-means. For $K = 256$, the optimal algorithm achieves less spread than standard K-means

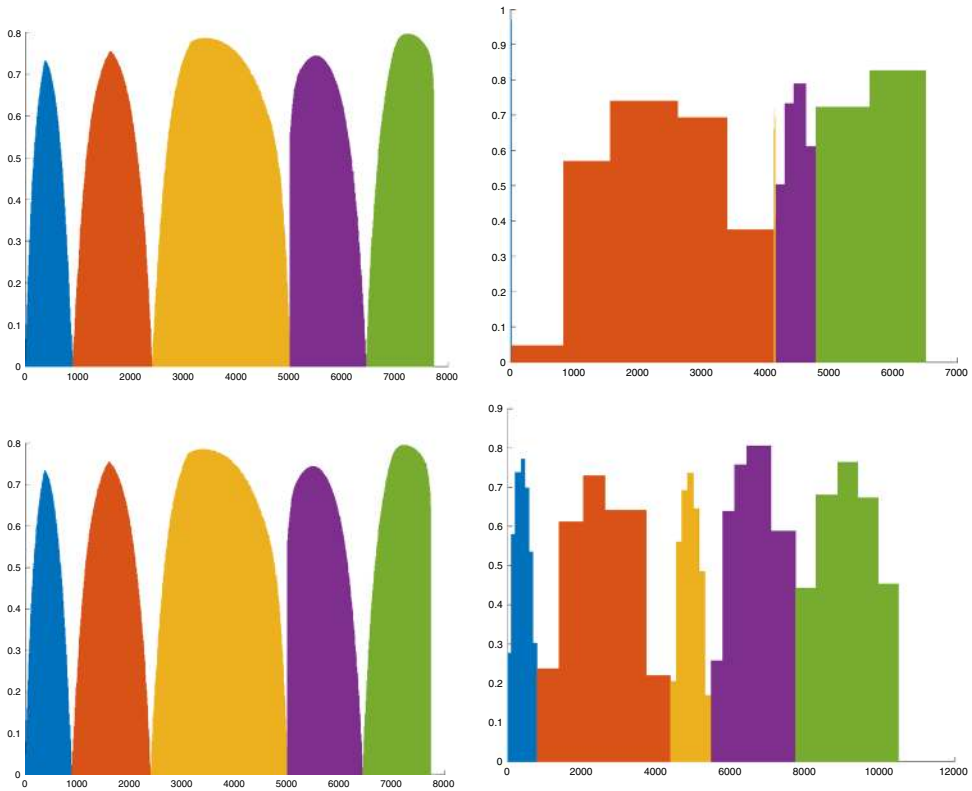

Fig. 4 Silhouette plots from the different clustering examples. *Left* shows the five clusters, when $K = 5$. *Top* shows one run of standard K-means, and *bottom* shows our optimal method. In this case both methods give very similar results, and both show balanced clustering silhouettes. *Right* shows the clustering when $K = 256$. *Top* is again one run of standard K-means, and *bottom* is our method. For visualization purposes, only a random five of the 256 clusters are shown. One can see that in this case, our method gives more balanced silhouettes than standard K-means

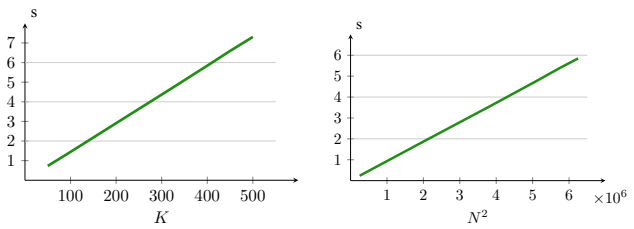

Fig. 5 *Left* shows execution time for running the complete Algorithm 2 as a function of the output discretization K. *Right* shows the execution time as a function of the square of the input discretization N. The plots follow the predicted behavior of the algorithm, i.e. , linear in the number of output bins K and quadratic in the number of input bins N

As a final check we calculated Fleiss' kappa index [17]. This measures how well a set of different clusterings agree, on a scale where a value that is negative or close to zero indicates low agreement, and a value close to one indicates high agreement. In this case we got 0.96 for $K = 5$, which means that all the clusterings from the different runs (including the optimal solution) agree to a very high extent. For $K = 256$ the Fleiss' kappa index was equal to 0.05, with little agreement. All the different measures point in the same direction that for the high-dimensional case that we are interested in, the optimal dynamic programming gives a much better solution than standard K-means. For our tone mapping application, the choice of $K = 256$ is a natural one, but one could still ask whether the number of clusters could be chosen by other means. One way of selecting K is by investigating the so-called gap statistic [51]. For the images that we have tested, we get a much lower value than 256 if we look at the gap statistic, typically around 5. This probably reflects more of the global modality of the distribution, than the smaller characteristics of the distribution that we want to capture. It does suggest that if speed is of very high priority, a smaller K could be chosen, and then some simpler interpolation scheme could be used in between the clusters. We have however not investigated this further.

6.2 Algorithm Complexity and Stability

Next we wanted to check whether the total algorithm followed the expected complexity. In order to do this we ran Algorithm 2 on a randomly generated input image and varied the number of input gray levels N and the number of output gray levels K. Our implementation was done in MATLAB, with the most time-consuming step, i.e., the dynamic programming part, done using compiled mex-functions. The MATLAB and mex files can be downloaded from [11]. All tests were conducted on a desktop computer running Ubuntu, with an Intel Core i7 3.6 GHz processor. The results are shown in Fig. 5, where the respective dependences on K and N are shown. Here we set $K = 256$ when N varied, and $N = 2000$ when K varied. The most time-consuming part of the algorithm is the dynamic programming step, and this is linear in K and quadratic in N which is validated in the graph.

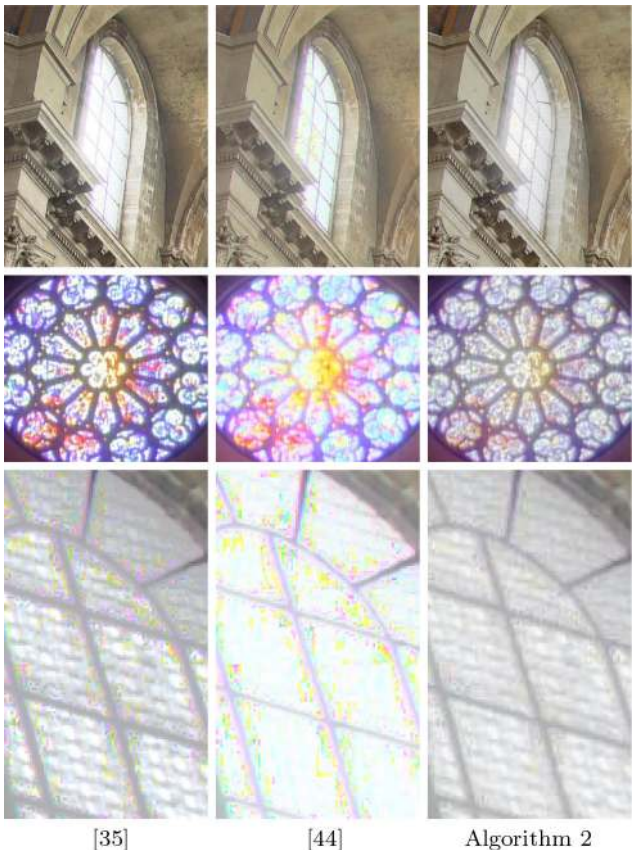

Fig. 6 The figure shows cutouts from the results on (*top* to *bottom*) NancyChurch3, Rosette and NancyChurch1. HDR radiance maps courtesy of Mantiuk [20] and Debevec [42]. The figure shows from left to right [35], [44] and our method. One can see that the compared methods suffer from over-saturation, color artifacts and loss of detail. The results are best viewed on screen

6.3 Results on HDR Images

We have tested our method on a number of HDR input images and compared with a number of standard tone mapping algorithms. The images were collected from R. Mantiuk [20] and P. Debevec [42]. We used the HDR image tool [32] to do the processing. It contains implementations of a number of tone mapping algorithms. Throughout our tests, we have only used the default parameter settings as supplied by *Luminance HDR*. It is probably so that in some cases better results can be found by tweaking the parameters manually, but since our method does not contain any parameters and the goal for us was to have an automatic system we opted for the default parameters. We have compared our method to the methods of [12,13,35,37,44]. Of these we found that [35] and [44]

Fig. 7 The result of running our Algorithm 2 on a number of HDR images. No parameters need to be set to produce the output. The results are best viewed on screen. HDR radiance maps courtesy of Debevec [42] and Mantiuk [20]

gave significantly better results over the set of test images. Our method gave very similar results to these two methods. In Fig. 6 we show magnified cutouts from three example images. Here we can see that the two compared methods (on the left) exhibit problems with over-saturation, loss of detail resolution and color artifacts. In Fig. 7 the output of our algorithm is shown for a number of different input HDR images.

Throughout our tests we used a fixed discretization ($N = 5000$) for the input intensity distribution. We have experimented with higher values of N but this does not give any noticeable effects. Further more, in our algorithm, empty bins are removed before the dynamic programming step, and the complexity of the algorithm will only depend on the number of non-empty bins. Sampling the input distribution with higher N will result in slightly finer modeling of the input luminance distribution, but most added bins will be empty. (So in this sense the complexity depends mostly on the actual input distribution and not on the fixed value of 5000).

Table 2 The tested HDR sequences from [25,40]

Sequence	Resolution	Frames	Time (s)	Framerate (fps)
Window	720p	236	49.8	4.7
Hallway	720p	331	51.6	6.4
Hallway 2	720p	351	58.3	6.0
Students	720p	251	62.5	4.0
Driving	720p	151	55.8	2.7
Exhibition	720p	189	44.8	4.2

The table shows the processing time for the tone mapping and the resulting framerate

6.4 Results on HDR Video

We have run our algorithm on a number of test HDR videos from [25,26,40], that were also used in the evaluation in [15]. An overview of the sequences can be seen in Table 2. We used the same parameter settings for all input videos. The number of input bins was fixed at $N = 5000$. We used seven neighboring images to estimate the key frame gray value distributions, and we used a fixed distance of 20 frames between key frames. We then ran Algorithm 3. In Fig. 9 the output of two of the frames for a number of sequences is shown. Even though the luminance greatly changes both spatially and temporally, we achieve significant contrast overall. We do not experience any flickering, color artifacts or ghosting. The dynamic programming solver was the same as for the still images, i.e. , a mex MATLAB implementation. The rest of the steps in Algorithm 3 were implemented in MATLAB. The total running times for our tone mapping on the different test sequences are shown in Table 2. The resulting corresponding framerates are also shown. In order to investigate the time dependence of our algorithm further, we conducted the same experiment as described in [15]. In this test, the output intensity for two spatial locations is plotted as functions of time or frame number. We chose the same sequence (the *student* sequence) and the same two locations as in [15], corresponding to two points with different temporal behavior. Four frames from our output are shown in Fig. 8 with the two points overlaid in red and green, respectively. The intensities for the two points are shown in Fig. 10. One can see that in this case there is no apparent flickering, as well as no saturation or overshooting issues.

We have also done comparisons to a number of state-of-the-art HDR video tone mapping algorithms. We used four sequences from the dataset of Froelich et al. [18], namely *Poker fullshot*, *Smith hammering*, *Cars fullshot* and *Showgirl 2*.

We have compared our results with three state-of-the-art video tone mapping algorithms, the zonal brightness coherency method of Boitard et al. [7], the temporally coher-

Fig. 8 Four frames from the output of Algorithm 3 with the *student* sequence as input. Also overlaid in red and green are the two locations whose time dependence is shown in Fig. 10

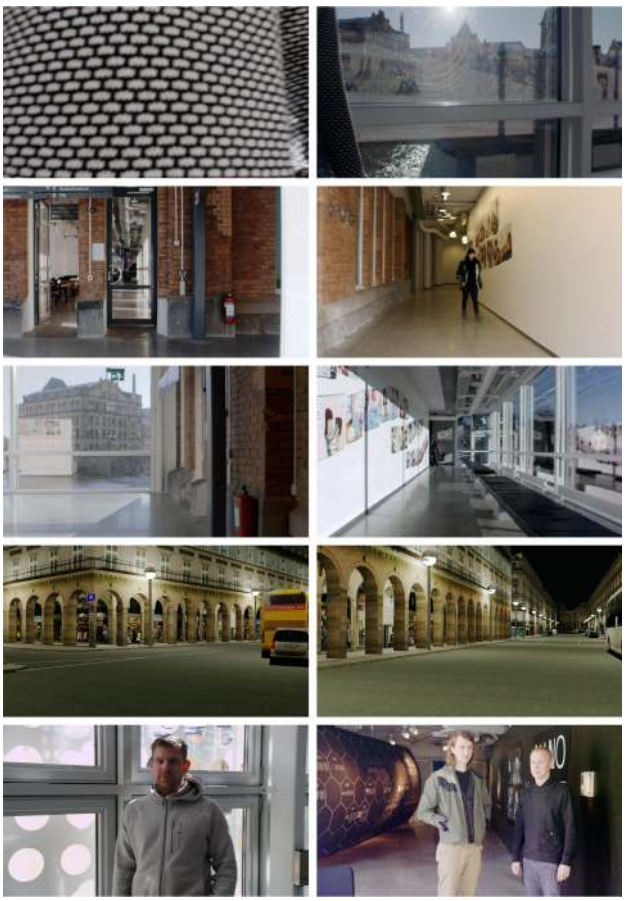

Fig. 9 The figure shows two frames (*left* and *right*, respectively) of the output from Algorithm 3, using (from *top* to *bottom*) the *window* sequence, the *hallway* sequence, the *hallway 2* sequence, the *driving* sequence and the *exhibition* sequence. The results are best viewed on screen

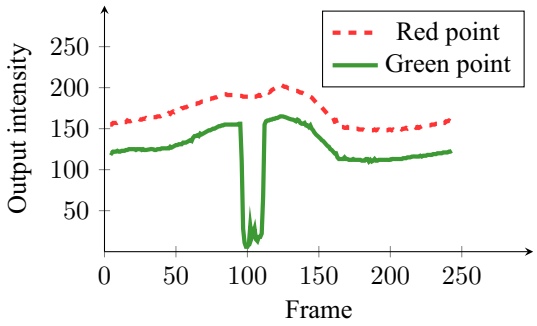

Fig. 10 The time dependence of the output intensity at the two points shown in Fig. 8 (indicated by the *red* and *green dots*). The graph shows the intensity that results from running Algorithm 3 on the *student* sequence. One can see that there are no apparent problems with flickering, overshooting or over-saturation for these points

ent local tone mapping method of Aydin et al. [2] and the real-time noise-aware tone mapping of Eilertsen et al. [14]. In Fig. 11 some resulting output frames are shown, with magnified cutouts. All methods generally give outputs with overall good brightness and contrast, with little temporal flickering and artifacts. Due to the local filtering of the compared methods, they exhibit stronger local contrast, but this can in some cases lead to artificial details and cartoonishness. We have also conducted a qualitative subjective comparison between the different tone mapping operators. We follow the setup in [14], where a number of persons evaluated the four test sequences, with respect to artifacts and image quality. Specifically the tone mapping operators were graded on overall *brightness*, overall *contrast*, overall color *saturation*, temporal color *consistency*, temporal *flickering*, *ghosting*, excessive *noise* and detail *reproduction*. We used a 32" 1920 × 1080 BenQ BL3200PT LCD monitor, with a peak luminance of 300 cd/m^2. The sequences were shown double blind in random order to ten persons, who graded them according to the above scale. The results can be seen in Fig. 12, where the results for the four tested tone mapping operators are shown, from top to bottom our method, Aydin et al. [2], Boitard et al. [7] and Eilertsen et al. [14]. One can see that all methods introduce very little artifacts. There is slightly more variation in the evaluation of the image characteristics, which can be expected. One can see that our method compares favorably with the other state-of-the-art methods. Note that our method is the only global method of the four tested algorithms.

7 Conclusion

We have in this paper presented a novel tone mapping algorithm that is based on K-means clustering. We solve the clustering problem using a dynamic programming approach. This enables us to not only solve the clustering problem

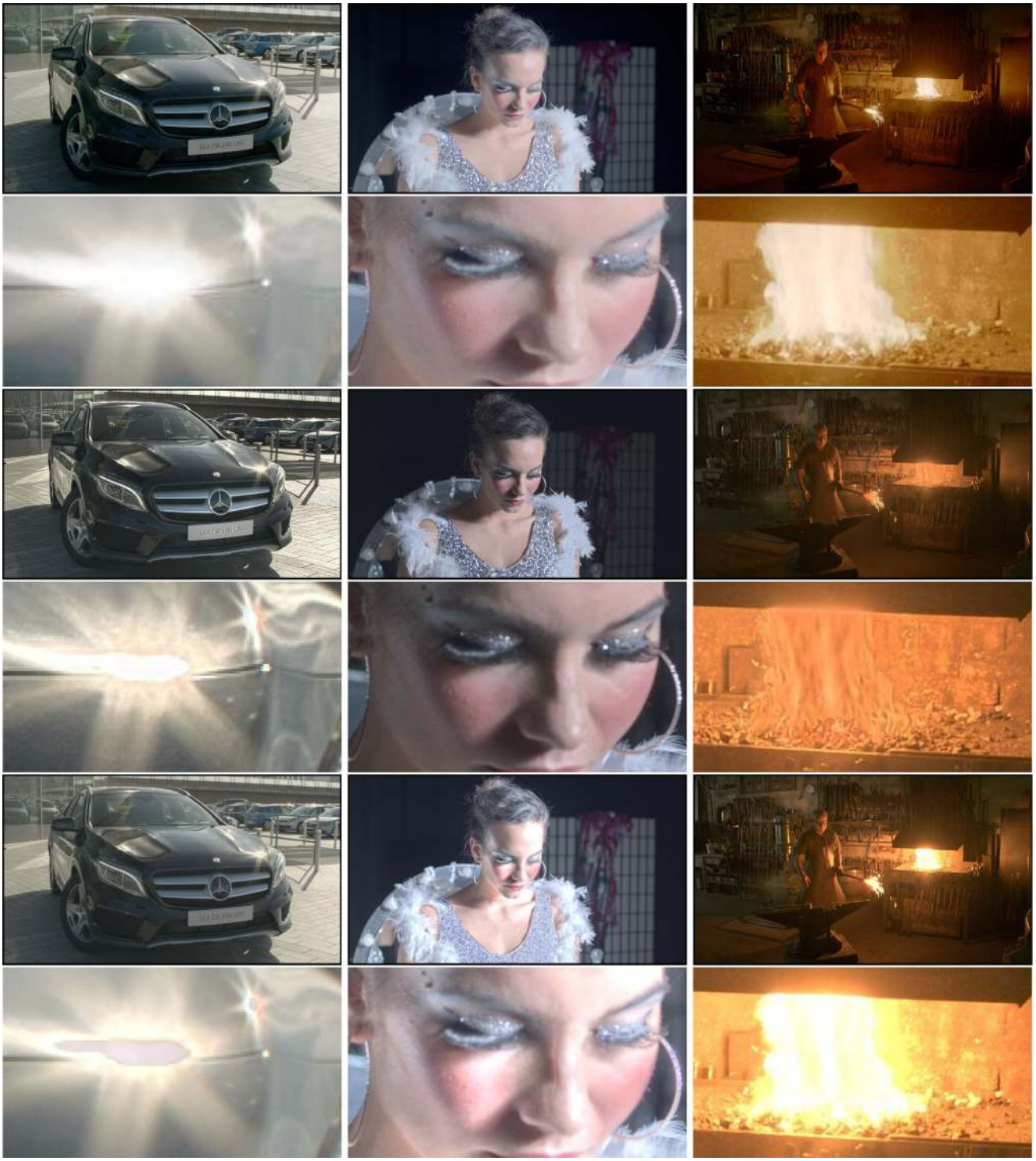

Fig. 11 Individual frames from the sequences in [18]. *Top two rows* show the output from our method, with magnified sections. The *next two rows* show the result of Eilertsen et al. [14], and the *lower two rows* show the result of Aydin et al. [2]. The competing methods give in some areas slightly higher local contrast, but also suffer from some artifacts. In the highlights of the *Cars fullshot* sequence, both Aydin and Eilertsen show color saturation artifacts. In the *Showgirl 2* sequence, the local contrast amplification of Aydin gives rise to the appearance of facial scarring. In the challenging *Smith hammering* sequence, Aydin again has problems with saturation and Eilertsen lacks medium-level contrast at the cost of having high detail contrast. This gives the fire a very artificial color appearance. The results are best viewed on screen

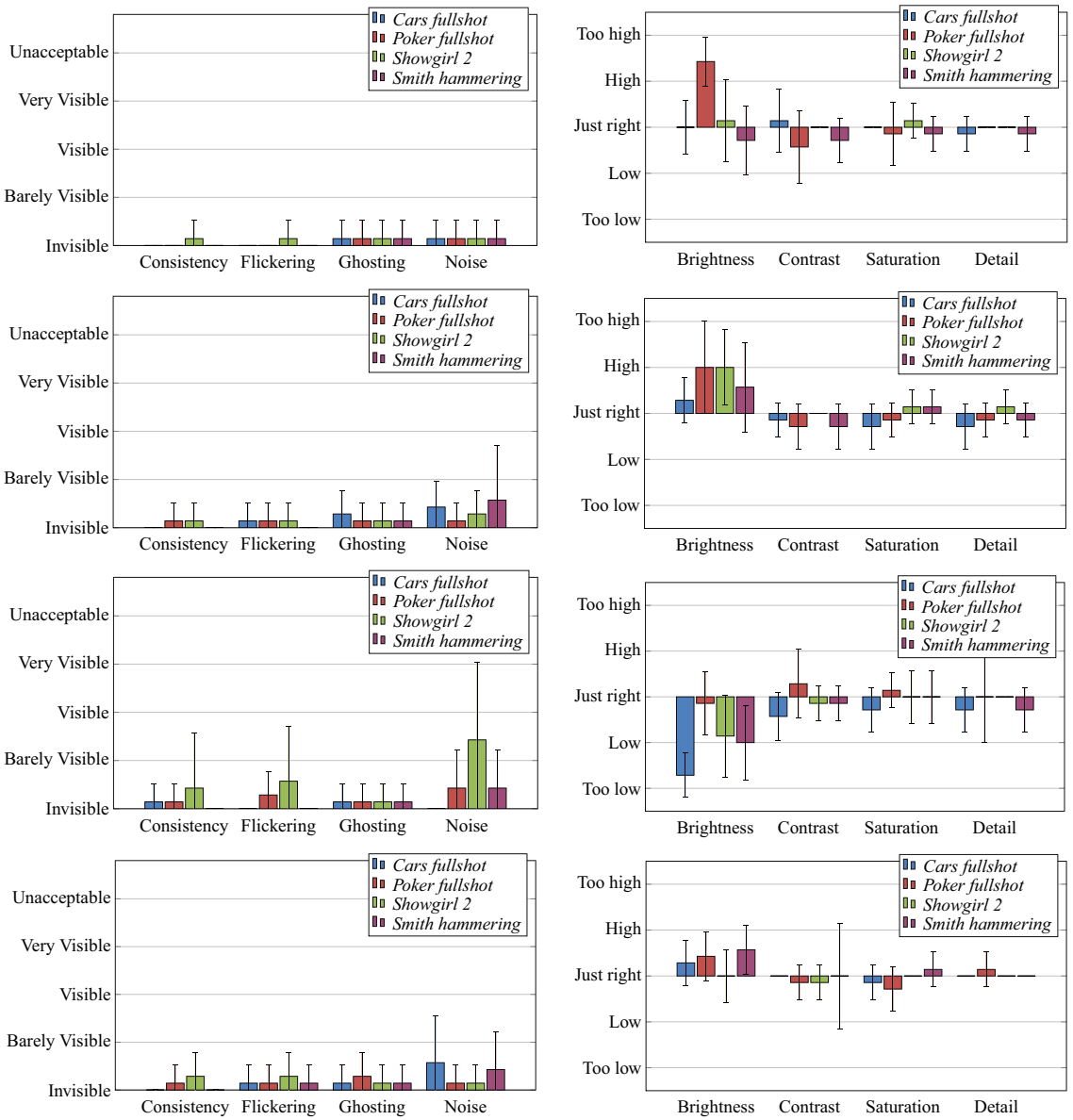

Fig. 12 The result of the subjective evaluation of the four tested tone mapping algorithms. The figure shows from top to bottom the result using our method, Aydin et al. [2], Boitard et al. [7], and Eilertsen et al. [14]. All methods give satisfactory results, with minor variations. Note that our method is the only purely global method

efficiently but also find the global optimum. The clustering algorithm runs in $O(N^2K)$ for N input luminance levels and K output levels. We have shown how this can be used to tone map both HDR color image and video data. Our algorithm gives comparable result to state-of-the-art tone mapping algorithms and with the large benefit of minimal need for parameter setting. In some cases in HDR imaging it is beneficial to be able to adjust the output manually, and some of the compared local methods could be tuned to get better effect in local color and contrast. But in many cases a totally automatic procedure is highly desirable.

References

1. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: Np-hardness of Euclidean sum-of-squares clustering. Mach. Learn. **75**(2), 245–248 (2009)

2. Aydin, T.O., Stefanoski, N., Croci, S., Gross, M., Smolic, A.: Temporally coherent local tone mapping of hdr video. ACM Trans. Gr. (TOG) **33**(6), 196 (2014)
3. Bellman, R.: A note on cluster analysis and dynamic programming. Math. Biosci. **18**(3), 311–312 (1973)
4. Bennett, E.P., McMillan, L.: Video enhancement using per-pixel virtual exposures. ACM Trans. Gr. (TOG) **24**(3), 845–852 (2005)
5. Benoit, A., Alleysson, D., Herault, J., Le Callet, P.: Spatio-temporal tone mapping operator based on a retina model. In: Computational Color Imaging, pp. 12–22. Springer (2009)
6. Boitard, R., Bouatouch, K., Cozot, R., Thoreau, D., Gruson, A.: Temporal coherency for video tone mapping. In: SPIE Optical Engineering+ Applications, pp. 84,990D–84,990D. International Society for Optics and Photonics (2012)
7. Boitard, R., Cozot, R., Thoreau, D., Bouatouch, K.: Zonal brightness coherency for video tone mapping. Signal Process. Image Commun. **29**(2), 229–246 (2014)
8. Celenk, M.: A color clustering technique for image segmentation. Comput. Vis. Gr. Image Process. **52**(2), 145–170 (1990)
9. Dasgupta, S., Freund, Y.: Random projection trees for vector quantization. IEEE Trans. Inf. Theory **55**(7), 3229–3242 (2009)
10. Debevec, P.E., Malik, J.: Recovering high dynamic range radiance maps from photographs. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pp. 369–378. ACM (2007)
11. https://github.com/hamburgerlady/democratic-tonemap/
12. Drago, F., Myszkowski, K., Annen, T., Chiba, N.: Adaptive logarithmic mapping for displaying high contrast scenes. Comput. Gr. Forum **22**(3), 419–426 (2003)
13. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. ACM Trans. Gr. (TOG) **21**(3), 257–266 (2002)
14. Eilertsen, G., Mantiuk, R.K., Unger, J.: Real-time noise-aware tone mapping. ACM Trans. Gr. **34**(6), 198 (2015)
15. Eilertsen, G., Wanat, R., Mantiuk, R.K., Unger, J.: Evaluation of tone mapping operators for hdr-video. Comput. Gr. Forum **32**(7), 275–284 (2013)
16. Ferwerda, J.A., Pattanaik, S.N., Shirley, P., Greenberg, D.P.: A model of visual adaptation for realistic image synthesis. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 249–258. ACM (1996)
17. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological. Bull. **76**(5), 378 (1971)
18. Froehlich, J., Grandinetti, S., Eberhardt, B., Walter, S., Schilling, A., Brendel, H.: Creating cinematic wide gamut hdr-video for the evaluation of tone mapping operators and hdr-displays. In: IS&T/SPIE Electronic Imaging, pp. 90,230X–90,230X. International Society for Optics and Photonics (2014)
19. Grossberg, M.D., Nayar, S.K.: High dynamic range from multiple images: Which exposures to combine. In: Proceedings of the ICCV Workshop on Color and Photometric Methods in Computer Vision (CPMCV), Nice, France (2003)
20. http://pfstools.sourceforge.net/hdr_gallery.html. Accessed 01 Sept 2014
21. Irawan, P., Ferwerda, J.A., Marschner, S.R.: Perceptually based tone mapping of high dynamic range image streams. In: Rendering Techniques, pp. 231–242 (2005)
22. Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High dynamic range video. ACM Trans. Gr. (TOG) **22**(3), 319–325 (2003)
23. Kiser, C., Reinhard, E., Tocci, M., Tocci, N.: Real time automated tone mapping system for hdr video. In: IEEE International Conference on Image Processing, pp. 2749–2752 (2012)
24. Kleinberg, J., Tardos, É.: Algorithm Design. Addison-Wesley, Boston (2005)
25. Kronander, J., Gustavson, S., Bonnet, G., Unger, J.: Unified hdr reconstruction from raw cfa data. In Computational Photography (ICCP), 2013 IEEE International Conference on IEEE, pp. 1–9. (2013)
26. Kronander, J., Gustavson, S., Bonnet, G., Ynnerman, A., Unger, J.: A unified framework for multi-sensor hdr video reconstruction. Signal Process. Image Commun. **29**(2), 203–215 (2014)
27. Larson, G.W., Rushmeier, H., Piatko, C.: A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Trans. Vis. Comput. Gr. **3**(4), 291–306 (1997)
28. Ledda, P., Santos, L.P., Chalmers, A.: A local model of eye adaptation for high dynamic range images. In: Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pp. 151–160. ACM (2004)
29. Lee, J.W., Park, R.H., Chang, S.: Local tone mapping using the *k*-means algorithm and automatic gamma setting. Consum. Electron. IEEE Trans. **57**(1), 209–217 (2011)
30. Levina, E., Bickel, P.: The earth mover's distance is the mallows distance: some insights from statistics. In: Computer Vision, 2001. ICCV 2001. Proceedings of the Eighth IEEE International Conference on IEEE, vol. 2, pp. 251–256. (2001)
31. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
32. http://qtpfsgui.sourceforge.net. Accessed 01 Sept 2014
33. Mahajan, M., Nimbhorkar, P., Varadarajan, K.: The planar *k*-means problem is np-hard. In: WALCOM: Algorithms and Computation, pp. 274–285. Springer (2009)
34. Malm, H., Oskarsson, M., Warrant, E., Clarberg, P., Hasselgren, J., Lejdfors, C.: Adaptive enhancement and noise reduction in very low light-level video. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on IEEE, pp. 1–8. (2007)
35. Mantiuk, R., Daly, S., Kerofsky, L.: Display adaptive tone mapping. ACM Trans. Gr. (TOG) **27**(3), 68:1–68:10 (2008)
36. Mantiuk, R., Mantiuk, R., Tomaszewska, A., Heidrich, W.: Color correction for tone mapping. Comput. Gr. Forum **28**(2), 193–202 (2009)
37. Mantiuk, R., Myszkowski, K., Seidel, H.P.: A perceptual framework for contrast processing of high dynamic range images. ACM Trans. Appl. Percept. (TAP) **3**(3), 286–308 (2006)
38. Oskarsson, M.: Democratic tone mapping using optimal *k*-means clustering. In: Scandinavian Conference on Image Analysis, pp. 354–365. Springer (2015)
39. Pattanaik, S.N., Tumblin, J., Yee, H., Greenberg, D.P.: Time-dependent visual adaptation for fast realistic image display. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 47–54. ACM Press/Addison-Wesley Publishing Co. (2000)
40. Petit, J., Mantiuk, R.K.: Assessment of video tone-mapping: Are cameras s-shaped tone-curves good enough? J. Vis. Commun. Image Represent. **24**(7), 1020–1030 (2013)
41. Ramsey, S.D., Johnson III, J.T., Hansen, C.: Adaptive temporal tone mapping. In: Proceedings of the 7th IASTED International Conference on Computer Graphics and Imaging, pp. 124–128. Citeseer (2004)
42. http://www.pauldebevec.com/Research/HDR. Accessed 01 Oct 2014
43. Reinhard, e, Heidrich, W., Debevec, P., Pattanaik, S., Ward, G., Myszkowski, K.: High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting. Morgan Kaufmann, Burlington (2010)
44. Reinhard, E., Stark, M., Shirley, P., Ferwerda, J.: Photographic tone reproduction for digital images. ACM Trans. Gr. (TOG) **21**(3), 267–276 (2002)
45. Robertson, M.A., Borman, S., Stevenson, R.L.: Dynamic range improvement through multiple exposures. In: Proceedings Inter-

national Conference on Image Processing, ICIP 99, Kobe, Japan, vol. 3, pp. 159–163. IEEE (1999)
46. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
47. Scheunders, P.: A comparison of clustering algorithms applied to color image quantization. Pattern Recognit. Lett. **18**(11), 1379–1384 (1997)
48. Scheunders, P.: A genetic c-means clustering algorithm applied to color image quantization. Pattern Recognit. **30**(6), 859–866 (1997)
49. Schlick, C.: Quantization techniques for visualization of high dynamic range pictures. In: Photorealistic Rendering Techniques, pp. 7–20. Springer (1995)
50. Steinley, D.: *K*-means clustering: a half-century synthesis. Br. J. Math. Stat. Psychol. **59**(1), 1–34 (2006)
51. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. J. R. Stat. Soc. Ser. B (Statistical Methodology) **63**(2), 411–423 (2001)
52. Tocci, M.D., Kiser, C., Tocci, N., Sen, P.: A versatile hdr video production system. ACM Trans. Gr. (TOG) **30**(4), 41 (2011)
53. Tumblin, J., Rushmeier, H.: Tone reproduction for realistic images. Comput. Gr. Appl. IEEE **13**(6), 42–48 (1993)
54. Tumblin, J., Turk, G.: Lcis: A boundary hierarchy for detail-preserving contrast reduction. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 83–90. ACM Press/Addison-Wesley Publishing Co. (1999)
55. Van Hateren, J.: Encoding of high dynamic range video with a model of human cones. ACM Trans. Gr. (TOG) **25**(4), 1380–1399 (2006)
56. Vaserstein, L.N.: Markov processes over denumerable products of spaces, describing large systems of automata. Probl. Pereda. Inf. **5**(3), 64–72 (1969)
57. Wang, H., Song, M.: Ckmeans. 1d. dp: optimal *k*-means clustering in one dimension by dynamic programming. R. J. **3**(2), 29–33 (2011)
58. Ward, G.: A contrast-based scalefactor for luminance display. Gr. Gems. **IV**, 415–421 (1994)
59. Warrant, E., Oskarsson, M., Malm, H.: The remarkable visual abilities of nocturnal insects: neural principles and bioinspired night-vision algorithms. Proc. IEEE **102**(10), 1411–1426 (2014)
60. Wilkie, K., Devlin, A., Chalmers, A., Purgathofer, W.: Tone reproduction and physically based spectral rendering. Eurographics 2002: State of the Art Reports, pp. 101–123 (2002)

Magnus Oskarsson received his M.Sc. degree in Engineering Physics in 1997 and Ph.D. in Mathematics in 2002 from the University of Lund, Sweden. His thesis work was devoted to computer vision with applications for autonomous vehicles. He is currently an associate professor at the Centre for Mathematical Sciences, Lund University, where his teachings include undergraduate and graduate courses in mathematics and image analysis. He is the author and co-author to a number of papers in international journals and conference proceedings within geometry, algebra and optimization with applications in computer vision, cognitive vision and image enhancement.