

Adaptive Camera-Based Color Mapping For Mixed-Reality Applications

Martin Knecht*

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Christoph Traxler†

VRVis - Center for Virtual Reality
and Visualization Research, Ltd.

Werner Purgathofer‡

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Michael Wimmer§

Institute of Computer Graphics and Algorithms
Vienna University of Technology

ABSTRACT

We present a novel adaptive color mapping method for virtual objects in mixed-reality environments. In several mixed-reality applications, added virtual objects should be visually indistinguishable from real objects. Recent mixed-reality methods use global-illumination algorithms to approach this goal. However, simulating the light distribution is not enough for visually plausible images. Since the observing camera has its very own transfer function from real-world radiance values to RGB colors, virtual objects look artificial just because their rendered colors do not match with those of the camera.

Our approach combines an on-line camera characterization method with a heuristic to map colors of virtual objects to colors as they would be seen by the observing camera. Previous tone-mapping functions were not designed for use in mixed-reality systems and thus did not take the camera-specific behavior into account. In contrast, our method takes the camera into account and thus can also handle changes of its parameters during runtime. The results show that virtual objects look visually more plausible than by just applying tone-mapping operators.

Keywords: Tone Mapping, Color Matching, Differential Rendering, Mixed Reality

Index Terms: I.1.3 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; I.4.10 [IMAGE PROCESSING AND COMPUTER VISION]: Enhancement—Filtering;

1 INTRODUCTION

Some application areas for mixed reality (MR) require a realistic rendering of virtual objects. Among them are architectural visualization, product design and marketing, and cultural heritage. In these applications, virtual objects should blend seamlessly into the real scene and provide a plausible illusion. Recent methods [8, 7] use advanced lighting simulations to make virtual objects look as if they were placed into the observed scene by taking shadows and indirect illumination between real and virtual objects into account.

Even though these methods can be quite accurate, virtual objects are still recognized easily, because their colors do not match with those captured by the observing camera. This problem is caused by the fact that the global illumination (GI) solution has a high dynamic range and tone mapping must be performed to get color values that are in the range of the observing camera. However, to

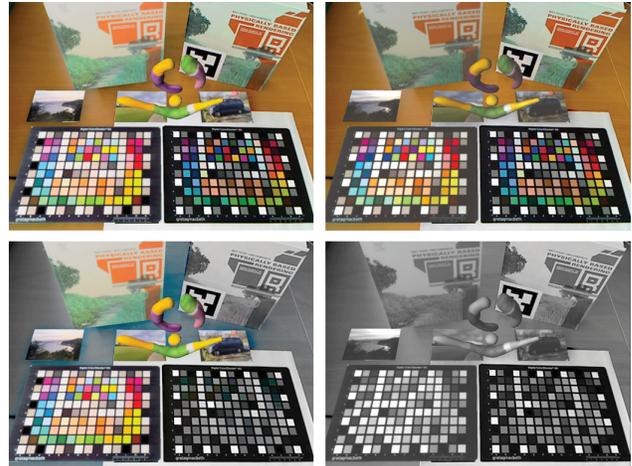


Figure 1: The left column shows renderings with the tone mapping operator from Reinhard et al. [9] without any adaption to the characteristics of the observing camera. On the right column our color mapping method is applied, resulting in better adapted colors as they were seen through the camera. In each image the color chart and the book on the right side are real objects.

our knowledge a lot of these tone mapping operators work independently of the internal camera settings and only focus on luminance mapping and ignore color mapping. Especially with upcoming mobile devices, where the camera automatically adapts to the light situation, our proposed method is a convenient way to enhance the visual quality of mobile augmented-reality applications.

In this paper we present a new adaptive color-mapping solution that considers the continuous image stream from the camera in real time. In this way, virtual objects appear as if also captured by the observing camera, thus removing a significant visual cue that destroys the plausibility of the illusion. We want to emphasize that our method focuses on color adjustment. Simulation of other camera artifacts, such as lens distortions for a better matching appearance of virtual objects has been addressed by Klein and Murray [6, 5].

Our method is based on Differential Instant Radiosity [7]. We derive a mapping to the camera color space RGB_{cam} by comparing the computed solution with the images delivered by the camera. We create color sample pairs of these buffers and derive a mapping function from them. A heuristic is used to also create color sample pairs for colors of virtual objects that are not available in these buffers. The main assumption behind this heuristic is that there is at least one dominant color channel.

Our contributions are:

- A tone-mapping method that adaptively maps colors of virtual objects to the colors as if they were seen through the camera.

*e-mail: knecht@cg.tuwien.ac.at

†e-mail: traxler@vrvis.at

‡e-mail: wp@cg.tuwien.ac.at

§e-mail: wimmer@cg.tuwien.ac.at

- A heuristic to generate samples of colors that are not represented in the real scene.
- A way to temporally smooth the color-mapping operator.

2 RELATED WORK

Klein and Murray [5] introduced a new compositing method for video-see-through AR that simulates most visible artifacts of small cameras. The considered effects are distortions, chromatic aberrations, blur, Bayer masking, noise, sharpening, and color-space compression. In this way, the appearance of virtual objects better matches those of real ones as captured by the camera. However, they do not attempt to achieve an accurate color matching, which requires at least an estimation of the real scene’s lighting conditions.

Color management by colorimetric characterization of cameras is another topic closely related to our work. Light that hits a sensor element in a camera is mapped to RGB values, forming the device-dependent color space. The transformation of this color space into a device-independent one, usually CIEXYZ (CIE tristimulus values), is called colorimetric characterization and is often described by an ICC profile.

Color management is so far only applied to single images, mainly in digital photography and print. Fairchild mentions that colors in video are purely device dependent (RGB to RGB) and presents some theoretical thoughts about how color management could work for it [3]. His observation is also true for the image stream of interactive MR applications. It was our intention to close this gap and find at least a convincing match between the RGB space of the camera and that of rendered virtual objects. In contrast to the classical methods of color management, our approach is adaptive and fulfills the real-time requirements of mixed reality systems. We avoid the painstaking process of defining an accurate colorimetric characterization by using many samples from a broad range of colors. This allows switching cameras at any time and promptly reacting to automatic camera adjustments.

Another related topic are tone-mapping operators. A very good survey was published by Čadík et al. [1]. They performed a perceptual evaluation of 14 tone mapping operators. Their analysis showed that photographic tone reproduction, proposed by Reinhard et al. [9], is one of the best in terms of image quality and performance. Therefore we used a simplified version of this operator in our work to compare the resulting images.

3 CAMERA-BASED COLOR MAPPING

We assume a mixed-reality setup where a real scene is captured by a camera and augmented by virtual objects before being presented to the user. Illumination is captured by a fisheye-camera to allow the scene to adapt to environment lighting.

When virtual objects are merged with the real video feed from the see-through camera, it is important that they appear in a visually pleasing way. Recent methods use tone mappers to calculate low dynamic range data from high dynamic range images. However, every time the camera parameters change (saturation, contrast, auto-shutter, ...) the tone-mapping operators must be adjusted manually (see Figure 1). What we would need is a color mapping from the virtual objects to the colors of the camera and this in an adaptive way so that changes over time can be handled adequately.

In our approach, we do not attempt to work in a device-independent color space. All computations are done in device-dependent RGB color spaces. The fisheye-camera captures the light that is defined through device-independent radiance values XYZ_{world} . This camera converts the incident light into RGB values, thus defining the color space that we will denote as reference color space RGB_{ref} . All elements that interact with light (materials and light sources) must be defined in that reference color space in order to get physically correct results.

The observing camera has a completely independent RGB color space denoted as RGB_{cam} . The method we present in this paper tries to find an appropriate mapping function τ from RGB_{ref} to RGB_{cam} as shown in Figure 2.

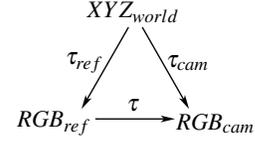


Figure 2: This figure illustrates the dependencies of the different color spaces. The proposed method tries to find a mapping function τ from RGB_{ref} to RGB_{cam} .

In order to find this mapping function, we need to gather information of the RGB_{ref} and RGB_{cam} color spaces that correlate with each other.

3.1 Methodology

Differential Rendering (DR) by Debevec [2] is a commonly used method to add virtual objects into a real scene. The method uses two global illumination solutions of the local scene (LS_{vr} and LS_r) which are defined in the RGB_{ref} color space. LS_{vr} stores the global illumination solution including real and virtual objects, whereas LS_r stores a global illumination solution taking only the real objects into account. Then the difference between both buffers is computed and added to a masked version of the video frame (LS'_{cam}). In DR these buffers need to be mapped to RGB_{cam} (by function τ) as follows:

$$\Delta L = \tau(LS_{vr}) - \tau(LS_r) \quad (1)$$

$$LS_{final} = LS'_{cam} + \Delta L \quad (2)$$

Here, ΔL is the difference image and LS_{final} represents the final composed image in RGB_{cam} color space.

Since we have the LS_r solution and the LS_{cam} image, the idea is to find a proper mapping function τ between these two buffers and then use this mapping for the LS_{vr} solution (where the virtual objects are visible) too. Our method can be summarized in four main steps. First color sample pairs are created. Second, to overcome issues with missing color samples a heuristic is applied. Third, a mapping function is computed based on the collected color sample pairs. The last step performs temporal smoothing on the computed mapping function to avoid flickering artifacts.

3.2 Color sample pair creation

We want good color matching for the virtual objects but we can only create color sample pairs (denoted as $\langle C_r, C_{cam} \rangle$) from the LS_r and L_{cam} buffers where real objects are visible. Therefore we need to find color samples in LS_r that are similar to the colors of the virtual objects.

In order to do this we first collect N color samples on virtual objects from the LS_{vr} buffer. In a second step, these samples are used to find similar color samples in the LS_r buffer. To compare the similarity between two colors we use the following equations:

$$\Delta h = |h(c_a) - h(c_b)| \quad (3)$$

$$h_m = 0.5 - \min(\Delta h, 1 - \Delta h) \quad (4)$$

$$match = \frac{h_m}{1 + |c_a - c_b|} \quad (5)$$

where c_a and c_b are the two colors to be compared, $h(\dots)$ converts the color to HSV color space and returns the hue channel, and $match$ is the final match of colors c_a and c_b .

The screen-space coordinates (x_i, y_i) of the pixel with the highest color match to the i th virtual object's color sample will be used to create a color sample pair $\langle C_r[i], C_{cam}[i] \rangle$ from the LS_{cam} and LS_r buffers:

$$C_r[i] = LS_r(x_i, y_i) \quad (6)$$

$$C_{cam}[i] = LS_{cam}(x_i, y_i) \quad (7)$$

3.3 Finding missing color samples

In the previous step, we assumed that the virtual objects have similar colors as the real objects. However, in typical AR applications, this will not always be the case, and then no suitable mapping function can be found for missing colors.

For these cases we propose a heuristic that tries to cover missing colors by creating new color samples before the mapping function τ is computed. The heuristic is based on the assumption that at least one dominant color (ranging from zero to full intensity) is available on the real objects. So for this dominant color a meaningful mapping can be found. By simply swapping color channels for each color sample pair, the mapping characteristics of one color channel can be transferred to another one.

In other words the volume covered by a mapping function that relies only on the dominant color is very small. If the dominant color is distributed into the other color channels, the mapping function is forced to cover a larger volume and thus other colors can be mapped to RGB_{cam} even though they are not directly available. Note that for these new colors, the exact mapping would be different. However, the main mapping properties are similar to the ones of the dominant color and thus, fewer visual artifacts appear than without any heuristic. For a given color sample pair the heuristic would randomly create a new one as follows (with $RGB \rightarrow GRB$):

$$c'_r.rgb = c_r.grb \quad (8)$$

$$c'_{cam}.rgb = c_{cam}.grb \quad (9)$$

3.4 Finding a mapping function

In the previous sections we created a set of N color sample pairs that represent corresponding colors in the RGB_{ref} and the RGB_{cam} color spaces. In order to calculate a mapping function, we use polynomial regression. Polynomial regression is commonly used in camera calibration, or, in a more general sense, for input/output device calibration. It is described in more detail in the book by Kang [4]. For our method we implemented three different types of polynomials:

- $C_{cam}(r) = a_0 + a_1r$, $C_{cam}(g) = a_0 + a_1g$, $C_{cam}(b) = a_0 + a_1b$
- $C_{cam}(r, g, b) = a_0r + a_1g + a_2b$
- $C_{cam}(r, g, b) = a_0 + a_1r + a_2g + a_3b$

Note that a_* represent the coefficients calculated by the regression method.

3.5 Temporal smoothing

The samples are created in screen-space and thus, changes in the view point or simply image noise may result in different color samples every frame. This causes temporal flickering artifacts between the adjacent frames. To dampen these sudden color changes, we temporally smooth the resulting mapping function on a per-component level. We use exponential smoothing [10] as follows:

$$a_i = ua_i + (1 - u)a'_i \quad (10)$$

where a are the component values after the regression, i is the index variable from 0 to the number of used coefficients and a' is

the value from the previous frame. The smoothing value u is used to steer the influence of every new result value. In our case a value of 0.3 led to pleasing results while still getting fast adaption to camera parameter changes.

4 RESULTS

The PC used for the test results has an Intel Core2 Quad CPU 9550 at 2.8GHz with 8GB of memory. The graphics card is an NVIDIA Geforce GTX 580 with 1.5GB of dedicated video memory. The operating system was Microsoft Windows 7 64 bits and the framework was developed in C#. All result images were rendered at a resolution of 1024x768 pixels using the DirectX10 API in conjunction with the SlimDX library. In all images the heuristic was activated.

We set up two different scenarios that should give an impression how our solution performs compared to simply using the tone mapping operator from Reinhard et al. [9]. These two scenarios should represent two extreme cases. Scenario A shows a real color checker board (Gretagmacth - ColorChecker Digital SG) in which our method can find a large number of different color samples. Scenario B is more challenging, since the only real object visible is a wooden desk with a more or less uniform brownish color.

The implementation supports three different regression modes, which are listed in Section 3.4. Table 1 shows a comparison between different polynomials for the regression, and the tone-mapping operator by Reinhard et al. [9]. Furthermore, we have adjusted the parameters of the cameras and show the adaption to it by the proposed method. In camera settings A, we reduced the saturation and increased the brightness. Setting B is the opposite, here the saturation was increased and the brightness decreased. Note that the implementation is not aware of any camera parameter settings. They were altered during runtime directly by the camera driver software. Furthermore we used similar real and virtual objects for comparison but there is no need to have equal objects in the scenes. The frame numbers in the brackets belong to Scenario A.

The results show that regression mode C leads to the best results in comparison to the other regression modes.

5 LIMITATIONS

Our approach currently requires a representation of the real scene, i.e., the geometry and BRDF estimation of real objects. This implies it does not work for conventional MR applications. Therefore, camera-based adaptive color mapping is tailored for MR systems that consider the real environment for rendering.

Another obvious limitation of our solution is the heuristic. It is based on the assumption that there is at least one dominating color on real objects and if that is not true, the heuristic will fail and visual artifacts may appear. There are some special cases where our solution will fail or produce undesired effects:

- When the tracking for real objects is too inaccurate, the see-through image will not exactly coincide with the GI solution for real objects (LS_r). Therefore wrong color pairs will be mapped to each other.
- In a situation where no real objects are visible because a virtual object completely occludes them, a color characterization cannot be calculated.
- Wrong color adjustment occurs when real objects are captured by the camera for which no representation for rendering exists. This usually happens when hands of interacting users are visible in the camera stream.
- If the discrepancy between real and virtual colors is too extreme, for example colorful virtual objects are placed into a gray-scale real environment, a mapping cannot be obtained and there would be no basis for our heuristic.

Mode	Scenario A	Setting A	Setting B	Scenario B
Reinhard (45 fps)				
A (30 fps)				
B (27 fps)				
C (26 fps)				

Table 1: In this table the different regression modes are compared to each other using the two scenarios as well as two different camera parameter settings. Setting A has low saturation and high brightness. Parameter setting B has high saturation and low brightness. The tone mapper of Reinhard et al. [9] does no approximation to camera colors at all, while the other modes do. Mode C yields the best approximation. The frame numbers in the brackets belong to Scenario A.

6 CONCLUSION AND FUTURE WORK

In this paper we presented a novel adaptive color-mapping method that can be used in mixed reality applications. Every frame we calculate a color mapping function that automatically adapts to the internal changes of the camera behavior. In this way, the virtual objects have colors as if seen by the camera. To enhance the quality of our mapping method, we proposed a heuristic that allows our method to function even if the needed colors are not available in the image. Furthermore, we exploit temporal coherence between adjacent frames to temporally smooth the mapping function and thus get better non-flickering results.

In the future, we want to find a more elaborate heuristic that adapts to a broader range of possible scenarios. Furthermore, we want to find solutions for some of the other limitations.

ACKNOWLEDGEMENTS

The authors wish to thank Raphael Grasset for the ISMAR logo model and Ralf Habel. This work was supported by a grant from the FFG-Austrian Research Promotion Agency under the program “FIT-IT Visual Computing” (project nr. 820916). Studierstube Tracker is kindly provided by Imagination Computer Services.

REFERENCES

- [1] M. Čadík, M. Wimmer, L. Neumann, and A. Artusi. Evaluation of hdr tone mapping methods using essential perceptual attributes. *Computers and Graphics*, 32(3):330–349, 2008.
- [2] P. Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high

dynamic range photography. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198, 1998.

- [3] M. D. Fairchild. A color scientist looks at video. *3rd International Workshop on Video Processing and Quality Metrics (VPQM)*, 2007.
- [4] H. R. Kang. *Computational Color Technology (SPIE Press Monograph Vol. PM159)*. SPIE- International Society for Optical Engineering, 2006.
- [5] G. Klein and D. Murray. Compositing for small cameras. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '08*, pages 57–60. IEEE Computer Society, 2008.
- [6] G. Klein and D. W. Murray. Simulating low-cost cameras for augmented reality compositing. *IEEE Transactions on Visualization and Computer Graphics*, 16:369–380, 2010.
- [7] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. In *Proceedings of the 9th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '10*, pages 99–108, 2010.
- [8] S. A. Pessoa, G. de S. Moura, V. Teichrieb, and J. Kelner. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In *Virtual Reality*, pages 3–10, 2010.
- [9] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Transactions on Graphics*, 21(3):267–276, 2002.
- [10] D. Scherzer, S. Jeschke, and M. Wimmer. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, pages 45–50. Eurographics Association, 2007.