Real-Time Pixel Luminance Optimization for Dynamic Multi-Projection Mapping

Christian Siegl[1]* Matteo Colaianni[1] Lucas Thies[1] Justus Thies[1]
Michael Zollhöfer[2] Shahram Izadi[3] Marc Stamminger[1] Frank Bauer[1]

1) Computer Graphics Group, University Erlangen-Nuremberg
2) Max Planck Institute for Informatics 3) Microsoft Research

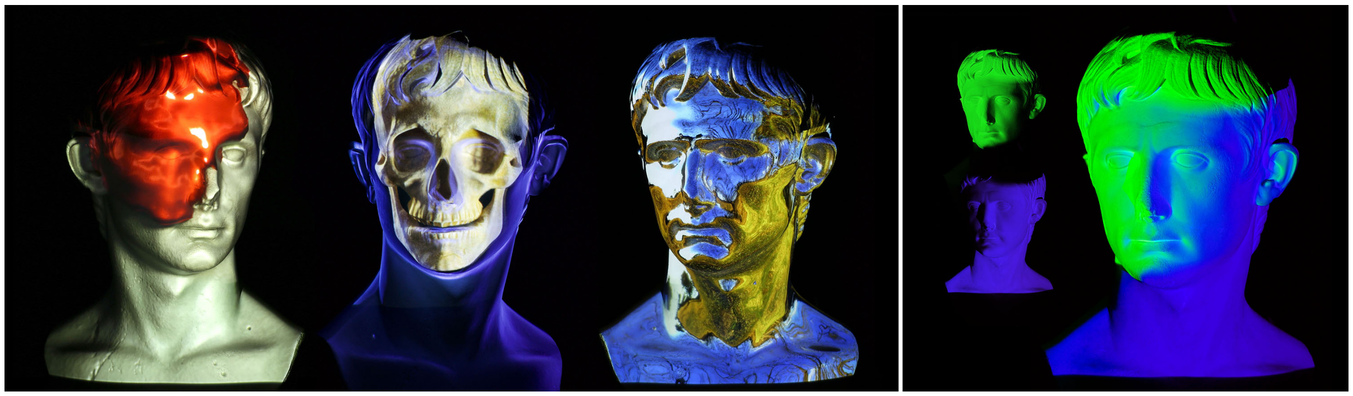


Figure 1: *Photos of augmenting our Augustus statue (see Figure 2) with multiple projectors. First, an animated texture is projected onto the object. Second, the statue appears to be made of glass with a skull behind the surface and third, the object appears as a perfect mirror using an environment map. On the right, the contribution of the two projectors is visualized. Projector 1 is green, projector 2 is blue (see Figure 2).*

Abstract

Using projection mapping enables us to bring virtual worlds into shared physical spaces. In this paper, we present a novel, adaptable and real-time projection mapping system, which supports multiple projectors and high quality rendering of dynamic content on surfaces of complex geometrical shape. Our system allows for smooth blending across multiple projectors using a new optimization framework that simulates the diffuse direct light transport of the physical world to continuously adapt the color output of each projector pixel. We present a real-time solution to this optimization problem using off-the-shelf graphics hardware, depth cameras and projectors. Our approach enables us to move projectors, depth camera or objects while maintaining the correct illumination, in real-time, without the need for markers on the object. It also allows for projectors to be removed or dynamically added, and provides compelling results with only commodity hardware.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

Keywords: mixed reality, projection mapping, multi-projector

*Christian.Siegl@cs.fau.de

To appear in ACM TOG 34(6)

1 Introduction

Augmenting physical objects like buildings or statues using projectors is frequently used in theme parks, museums and art installations. More recently, systems such as IllumiRoom and RoomAlive have brought *projection mapping* into the living room and the consumer domain [Jones et al. 2013; Jones et al. 2014]. This notion of spatial augmented reality (SAR) [Bimber and Raskar 2005], augments arbitrary real-world objects with virtual content through *projection mapping*, and provides an alternative experience, which is more shared than head-worn VR systems.

To increase the resolution and surface area covered by SAR systems, multiple overlapping projectors need to be used. However, illuminating an arbitrary surface with complex geometry using multiple projectors obviously introduces visible errors in areas where the projectors overlap.

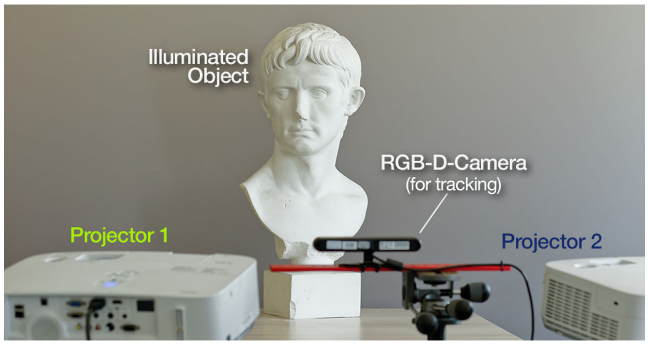


Figure 2: *Our primary setup: Projector* 1 *projects from the chin upwards, projector* 2 *from the temple downwards. In the overlapping regions both projectors can illuminate the statue.*

Furthermore, many existing projection systems require a *static* (i.e. non-moving) setup. This is constraining in many SAR scenarios where the projector, camera or target object may intentionally or unintentionally move. For example, imagine a user accidentally bumping into these objects, or novel interactive scenarios where rendered content needs to be projected either on a moving surface (e.g. [Bandyopadhyay et al. 2001]) or with a moving projector (e.g. [Molyneaux et al. 2012]).

In this paper we introduce a novel, adaptable, and real-time projection mapping system that supports multiple projectors and high quality rendering of dynamic content on white Lambertian surfaces of complex geometrical shape. Our system allows for smooth blending across multiple projectors using a new optimization framework that simulates the diffuse direct light transport of the physical world. This allows to continuously adapt the color output of each projector pixel such that the joint illumination on the target geometry equals the desired color. Our system considers the incident angle, distance and footprint size of all pixels from different projectors.

We solve this problem on the GPU, for each projected pixel in real-time, using only an additional off-the-shelf depth camera to observe the geometry and appearance. With one depth camera calibrated to the projector setup, the object can be moved freely while maintaining correct illumination in real-time, without the need for markers on the object. To gain the possibility to also move the projectors, every moving projector needs its own depth camera for tracking. It also allows for projectors to be removed or dynamically added, and provides compelling SAR results with only commodity hardware.

The main contributions of our paper are:

- A novel projection mapping system that uses commodity hardware, which for the first time allows multiple projections onto arbitrarily shaped surfaces, where the projectors and objects can all move, and no markers are required.
- A novel optimization approach that adapts the contribution of multiple projectors in real-time by modeling the diffuse direct light transport in the scene.
- A real-time demonstration of the capabilities of our system with compelling SAR scenarios.

2 Previous Work

There has been extensive work on enabling augmented reality scenarios using projection mapping (for an extensive review see [Bimber and Raskar 2005; Bimber et al. 2008]). Whilst recent work has captured much press attention [Jones et al. 2014; Jones et al. 2013], there has been much work over the last two decades, stemming from the seminal ‘Office of the Future’ project [Raskar et al. 1998].

Since this early work, which explored the use of large and curved projection surfaces that could be semi-automatically calibrated, there has been much work on projection systems, calibration techniques, and camera-projector systems. Systems have looked at the problem of projecting and interacting across multiple flat surfaces [Pinhanez 2001; Wilson and Benko 2010], even using a moving handheld projector [Cao and Balakrishnan 2006; Harrison et al. 2011]. One specific problem explored is the blending of multiple projections on flat or pseudo-flat surfaces [Brown et al. 2005; Majumder and Brown 2007]. These systems borrow from the large literature on image-based rendering and blending techniques (see [Shum et al. 2008] for a review). However, these systems have not explored complex geometry or moving surfaces.

More recently, with the advent of real-time depth cameras, projector systems have begun to deal with more complex geometries. IllumiRoom [Jones et al. 2013] explored projecting around the periphery of a television screen, and RoomAlive [Jones et al. 2014] explores multiple projections within an entire room. However, neither of these systems deal with moving projector/cameras or moving scenes. One example of a moving handheld projector system is [Molyneaux et al. 2012], which is used to render content onto real-world geometry of arbitrary shape, captured using the KinectFusion system [Izadi et al. 2011; Newcombe et al. 2011]. However, this system did not scale to moving objects, nor multiple projectors.

Other work has focused on the problem of correcting the projected image due to inherent color and visual artifacts using various radiometric compensation techniques [Grundhöfer and Bimber 2008; Grundhöfer 2013], or by modeling the inverse light transport of the projector [Wetzstein et al. 2007]. Some projector systems compensate in real-time for color changes on dynamic projection surfaces [Fujii et al. 2005]. Other systems compensate for environment lighting changes [Bimber et al. 2005a] or even the material properties of the scene [Bimber et al. 2005b; Law et al. 2011]. These systems however often do not deal with complex geometries or support movement of the surface or projector/camera system.

Shader Lamps [Raskar et al. 2001] demonstrated compelling results of using projection onto arbitrary shaped white Lambertian surfaces. This included various rendering and animation effects, and support for multiple projectors. However, a time-consuming *static* calibration technique was performed by manually moving a cross hair in the projector view to highlight pixels that illuminate known object points. In follow-up work, [Lee et al. 2004] used fiber optic sensors embedded inside the object to localize a moving object, but not in real-time. Bandyopadhyay et al. [2001] and Lee et al. [2008] demonstrated projecting on a moving object, but required infrared markers and/or magnetic sensors on the object, and only a small working volume. DisplayObjects [Akaoka et al. 2010] also used a marker-based tracking system, to track the display objects. In these systems the projector and sensor system are assumed to remain static. Resch et al. [2014] supports a moving projector-camera system, but does not scale to multiple projectors modeling the light transport to ensure correct per-pixel projection rendering.

Sueishi et al. [2015] present a projection mapping system that is able to dynamically project onto moving objects. They use a 1000fps camera and galvanometer mirrors to track and illuminate the object with a single projector. The resulting tracking is extremely fast and impressive results are achieved. Note that this tracking approach is orthogonal to our contribution and could be used to significantly reduce the latency of our system.

Another area utilizing projection mapping is CAD. Konieczny et al. [2006] use multiple projectors for material simulation. However, their multi-projector handling comes down to using different projectors for fore- and background. Sheng et al. [2009] use multiple projectors for simulating sunlight on architectural models. Here the scene is static as well and could greatly benefit from the dynamics that our system introduces.

Our work extends the growing body of GPU-based optimization literature, which has gained interest in the field of 3D reconstruction and geometry processing, since it is a crucial prerequisite for interactive real-time applications. In the KinectFusion approach [Newcombe et al. 2011; Izadi et al. 2011], the GPU is exploited to build the system required for real-time 6DoF camera tracking and reconstruction. In Weber et al. [2013], efficient matrix layouts and approaches for linear optimization on GPUs are evaluated in the context of finite element simulation. Zollhöfer et al. [2014] extends this to a non-linear Gauss-Newton optimization framework that is

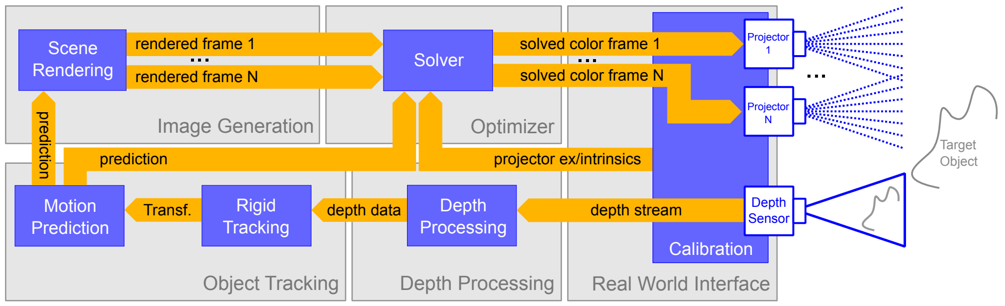


Figure 3: *Our pipeline for augmenting real-world objects with N projectors.*

used for real-time non-rigid surface registration. This framework has been adapted [Wu et al. 2014] to a GPU-based Gauss-Newton solver that exploits fast shared memory based on a *Schwarz alternating procedure* for real-time shading based refinement of commodity depth maps. Kaspar and Deng [2015] use a GPU optimizer to model constrained meshes in real-time. They use one thread per constraint and one kernel per constraint type. In contrast to these previous approaches, we propose a GPU framework for bounded non-linear optimization under given in-equality constraints.

3 System Overview

When illuminating real-world, non-convex objects using a single projector, self-shadowing will break the immersion for spectators. Therefore, we need a setup consisting of multiple projectors (Figure 2) where one can fill regions shadowed from the other. In Figure 4a two projectors illuminate an object with a pure white image. While shadows are filled in, it is evident that the surface still shows shading mainly due to Lambertian attenuation. Taking Lambertian and distance effects into account, we can produce a uniformly lit surface (see Figure 4b). This gives us a blank canvas to generate arbitrary appearances (see Figure 1) without interference from real-world shading. We describe this in more detail in Section 4.2.

For this to work, our system requires knowledge of the projectors' and the illuminated objects' positions. Since we support fully dynamic setups where projectors and objects are allowed to move, real-time tracking is needed. For this tracking and the following pixel luminance correction, we need a high quality 3D scan of the target geometry. Furthermore, for high quality projection mapping and the correct calculation of complex overlap scenarios between projectors, we need highly accurate intrinsic and extrinsic parameters of all hardware components in our setup described in the following paragraphs.

Our software-system is based on the pipeline depicted in Figure 3. The RGB-D camera provides our system with rigid tracking data. To compensate for latencies, motion prediction is applied before the transformation is sent to the rendering. The renderer then computes the desired surface colors of the object that will be used in our solver stage to compute the pixel colors for each projector.

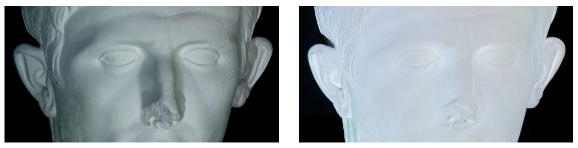


Figure 4: *(a) Our statue illuminated with plain white light from two projectors. (b) The statue illuminated from both projectors when compensating for Lambertian attenuation, distance and overlap.*

Hardware Setup Our hardware setup consists of a standard desktop workstation with an `Intel Core i7 4771 (3.5GHz)`, 32GB of RAM and two graphics cards. The first graphics card, an `NVidia GeForce 760`, performs the tracking detailed in the next paragraph. The second graphics card, an `NVidia GeForce 980`, is used for rendering and solving the non-linear multi-projection system described in Section 4.

As projectors we use two `NEC NP-P451WG` and for one demo two `Optoma ML750e LED` devices with a resolution of 1280 by 800 pixels. For object tracking `ASUS Xtion PRO Live` depth sensors are utilized. For the projectors' intrinsic calibration, we use a `Logitech C920 HD` camera.

An exemplary setup with a single depth camera, two stationary projectors and a target object (our Augustus statue) is depicted in Figure 2.

Camera Calibration In order to calibrate the intrinsics of our projectors we need a calibrated color camera. We rely on the widely used `OpenCV` implementation of the calibration procedure presented by Zhang [2000]. This algorithm delivers intrinsic parameters and distortion coefficients for our color camera. Intrinsics for the depth camera are provided by the sensors' manufacturers.

Projector Calibration The next step towards getting a completely calibrated system is the intrinsic calibration of the projectors. To get such a calibration, we project sequences of points onto different planes (tracked using multiple optical markers) that are captured by the color camera. The color camera is only used for this calibration step and is not a part of our on-line system.
An additional calibration step required for multi-projector setups is color calibration. All projectors we used, showed noticeable differences in color rendering. We use an `i1 Display Pro` colorimeter to obtain color profiles for all projectors.

Extrinsic Calibration The last step we need is an extrinsic calibration between the depth camera and the projectors. We use a 3D-Calibration object with a known pattern that can be tracked with our object tracker. To find correspondences between the depth image and a projector, we identify pixels illuminating known points on the surface of the calibration object. This is done once, subsequent tracking does not rely on markers.

Object Tracking We track the projectors 6DoF transform relative to the object based on the captured depth data of the input sensor

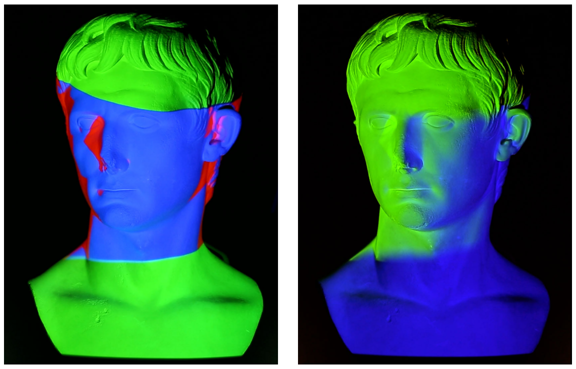

(a) (b)

Figure 5: *Left: The different types of rays contributing to coloring the surface. Green regions are illuminated by only one projector, blue regions by both. The red colored regions are occluded by the target geometry from one of the projectors. Right: The light contribution of a* 2*-projector case (projector* 1 *green, projector* 2 *blue, see Figure 2). The blending between projectors in overlap regions adapts to the surface shape.*

and our 3D scene model. To this end, we employ a projective iterated closest point strategy which is based on a point-to-plane metric. Similar to previous work on online 3D reconstruction [Izadi et al. 2011; Nießner et al. 2013; Newcombe et al. 2011], we iteratively linearize the problem around the last estimate and build the corresponding linear system on the GPU. The resulting small 6×6-system is efficiently solved on the CPU using Singular Value Decomposition.

Motion Prediction While our system runs at real-time framerates, the object tracking suffers from delays introduced by the on-chip processing of the RGB-D camera and the round trip time through our computation pipeline for tracking. This is noticeable as a lag between user interaction and corresponding changes in our tracking matrices. Since this effect impairs the immersion for the user, motion prediction is implemented in our system. We use a statistically based predictor to fit a spline into the past N frames that is extrapolated to predict the tracking trajectory. An additional weighted low-pass filter over new frames improves the robustness against outliers.

4 Dynamic Multi-Projection Optimization

Augmenting a dynamically moving object using multiple, potentially moving, projectors imposes a hard global optimization problem in the unknown per-pixel color values of the projectors that has to be solved under a tight real-time constraint. This global optimization problem is defined by finding the correct output color $\mathbf{p} = [p^k_{i,j}]^\top = [p_h]^\top$ for every projector k at each pixel position (i, j) that generates the desired color values $\mathbf{l} = [l^k_{i,j}]^\top = [l_h]^\top$ for every surface point x_h. To simplify notation, we integrate indices i, j, k to a single index h. Therefore, x_h ist the surface point hit by the central ray of pixel h.

4.1 Ray – Surface Interaction

In order to compute the light contribution, we need to take several physically motivated effects into account.

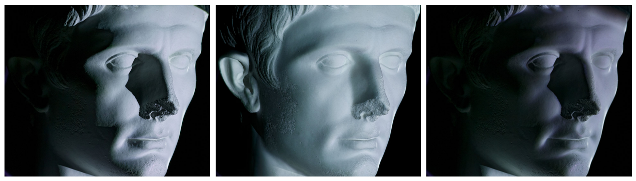

(a) (b) (c)

Figure 6: *(a) The result of a geometry aware illumination (Phong shading) from a single projector. (b) The shadow on the cheek is lit by a second projector. (c) shows the contribution of projector* 2 *to this multi-projection by manually occluding projector* 1*.*

Interaction of Projector Rays on the Surface The first, and maybe most important point is the interaction of projector rays with the surface of an object. Figure 5a shows different ray interaction types on the surface where ...

... only one projector hits (green, single-projector ray).

... at least one ray is occluded by the target geometry while others hit (red, shadowed multi-projector ray).

... rays from multiple projectors hit (blue, multi-projector ray).

An immediate advantage of multiple projectors is additionally covered space. As motivated in Section 3, we also illuminate regions not reachable by a single projector (red/green regions). In addition we are able to favor projector rays with a better incident angle (smaller footprint) in the blue regions.
Figure 6 shows a close up of the red area next to the nose (with Phong shading). Using only projector 2 (see Figure 2), the nose casts a shadow on the cheek (Figure 6a). With the addition of projector 1 and our algorithm, this shadow can be filled seamlessly (see Figure 6b). In Figure 6c, the contribution of projector 2 to Figure 6b is captured. Note the dimming of the projector on the statue's right cheek. This is due to projector 1 illuminating these parts at a better incident angle. The blending (visible in Figure 6b) is constantly calculated, allowing us to move objects while still keeping a uniform lighting.

The multi-projector rays (blue in Figure 5a) induce the globality into our system. This is illustrated in Figure 7 for an exemplary setup with 2 projectors. In the first case, both projectors have a similar location relative to a surface patch and every pixel of the projectors illuminates approximately the same spatial surface extent. Therefore, depending on the shift in the pixel grid, each pixel of the green projector interacts with at most 4 other pixels of the blue projector (see Figure 7a). In contrast, if the orientation of both projectors differs, the locality in the pixel-to-pixel mapping is lost and the problem has a global nature (see Figure 7b). While the projection of the pixel grid from the green projector has an equal spacing on the surface, the pixels of the blue projector are stretched out due to perspective foreshortening. As a result, the projected area of certain pixels is much larger, thus associating the corresponding pixel with more pixels of the green projector.

Quality of Rays As another important contribution to our system, a balancing between rays (based on their quality) is introduced. In general, rays hitting the surface at a more perpendicular angle will result in a smaller footprint on the surface. This is preferable over rays hitting at smaller angles. The smaller the footprint, the more local the influence of each ray on the actual surface is and the more detailled the projected texture can be. Figure 5b depicts the contribution to the final surface color for each projector. To our knowledge, all other approaches blend projectors based on the location of a pixel in projector space. Our method blends using properties of

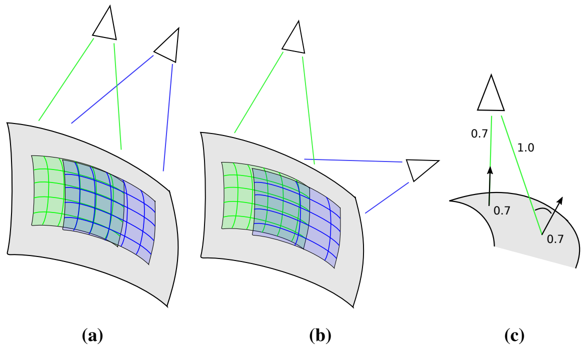


Figure 7: *Two configurations of projectors. (a) Both projectors generate similar pixel footprints on the surface. (b) The blue projector projects from a more level point, generating larger pixel footprints. (c) Shows why a reserve for correcting Lambertian and distance effects is required.*

the actual surface geometry, dynamically taking the orientation of the projector to the geometry into account.

Physical Properties of Light The most influential physical property to our system is the Lambertian cosine law. As described in Section 3, we aim to light the surface in a way that viewers will not notice the shading generated by the projectors. To achieve this we need to set a maximum white intensity that is below 1 (the maximum projector intensity). This gives us a reserve to increase a ray's intensity to compensate for the previously mentioned Lambertian attenuation. Figure 7c shows this effect for two exemplary rays. Here the maximum white intensity is set to 0.7. We also use this reserve to compensate the attenuation due to increasing distance to the object.

4.2 Objective Function

In this Section, we present an objective function that transparently handles dynamic scenes, multiple projectors and optimization at real-time rates. The problem of finding ideal per-pixel color values $\mathbf{p} = [p_{i,j}^k]^\top = [p_h]^\top$ for all projectors is posed as a variational non-linear bounded optimization problem in the unknown per-pixel color values:

$$\mathbf{p}^* = \underset{\substack{\mathbf{p} \\ s.t.\ p_h \in [0,1]}}{\operatorname{argmin}} \; E_{total}(\mathbf{p})$$

Our re-coloring objective consists of four main parts:

$$\begin{aligned} E_{total}(\mathbf{p}) = \quad & w_{light} E_{light}(\mathbf{p}) & + w_{reg} E_{reg}(\mathbf{p}) \qquad + \\ & + w_{balance} E_{balance}(\mathbf{p}) & + w_{bound} E_{bound}(\mathbf{p}) \end{aligned}$$

The fitting term E_{light} models the diffuse direct light transport in the scene and allows us to specify the re-coloring for an object, E_{reg} enforces a locally coherent brightness, $E_{balance}$ balances the projectors against each other and $E_{bound}(\mathbf{p})$ is a soft box-constraint that keeps the parameters in the $[0, 1]$ range of real-world projectors. Note, while we use a parameter re-projection strategy for the optimization of the energy, these soft constraints help to achieve plausible results. To balance different parts of our objective function we use weights w_*. Our method turned out to be quite robust to the choice of parameters. Therefore, we set all weights w to 1. In the following, we explain the individual parts of our novel objective function and show how to compute the best solution of the associated optimization problem at real-time rates.

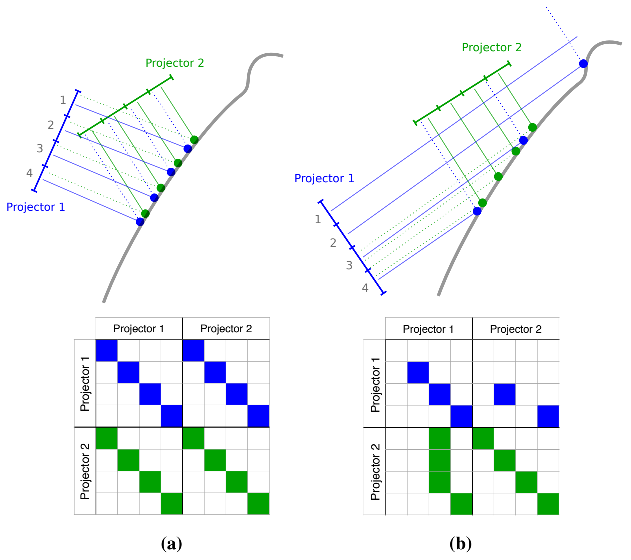


Figure 8: *Example configurations of two projectors with their corresponding light-transport matrices. (a) The projected pixels have a similar spacing on the surface. Every re-projection hits another pixel in the other projector. (b) Projector 1 is more level. The first pixel of projector 1 no longer hits the surface, the row in the matrix remains empty. The second ray hits the surface, but the re-projection misses projector 2. The corresponding row only has a diagonal entry. Multiple surface points of projector 2 are re-projected into the same pixel of projector 1, resulting in three green vertical matrix entries.*

Diffuse Direct Light-Transport The first part of our objective function allows to specify an arbitrary re-coloring of the object. To this end, it takes the physical transport of direct light in the real-world scene into account:

$$E_{light}(\mathbf{p}) = ||\mathbf{Tp} - \mathbf{l}||_2^2$$

Here, $\mathbf{T}$ is the scene's direct light transport operator and $\mathbf{l}$ is a vector of desired surface color values. $||\ .\ ||_2$ denotes the ℓ_2-Norm. Therefore, $\mathbf{T}$ encodes the physical properties of the surface and takes the relative configuration of the projectors as well as of the object into account, mapping pixel values p_i to surface color values l_i. Our operator $\mathbf{T}$ only considers direct illumination from the projectors, but no indirect light transport on the object. Note that $\mathbf{T}$ could easily be extended in this direction, however the matrix structure would become more complicated and solving the system more expensive. Fig. 8 (top row) illustrates the structure of $\mathbf{T}$ for two simple example configurations. The corresponding matrix structure is shown in the bottom row. Every pixel p_i is projected onto the surface and corresponds to a row in the matrix $\mathbf{T}$. If the projection misses the target object, the row is filled with zeros. Otherwise, the hit surface point is re-projected into the image planes of all projectors. Those re-projections p_j are added as a non-zero value into their corresponding column j. I.e. the diagonal entries encode, that a pixel always sees its own reprojection. Every re-projection can only hit one pixel p_j of every other projector, thus the number of entries per row is limited to the number of projectors in the setup. In the second example, see Fig. 8b, the blue projector misses the object leading to a zero row. Multiple pixels of the green projector map to the same pixel of the blue projector resulting in the three vertical green entries in the matrix. Due to this re-projection strategy our system is also indifferent to projectors with different resolutions. To achieve real-time performance, we have to exploit the sparsity of $\mathbf{T}$ in the optimization.

The entries of $\mathbf{T}$, map the brightness p_i of a projected pixel to the corresponding reflected intensity at a surface point. Our mapping is based on a *Lambertian* reflectance assumption coupled with a distance based attenuation term:

$$t_{ij} = \frac{\langle \mathbf{n}_i, \mathbf{i}_i \rangle}{d_i^2} \cdot v_{ij}$$

Here, $\mathbf{n}_i$ is the normalized surface normal at the corresponding surface point and $\mathbf{i}_i$ is the normalized negative incident light direction. d_i denotes the distance of the surface point to the projector. In cases where the hit point of ray i is within the surface footprint of pixel j's projection, the visibility v_{ij} is 1. In all other cases it is 0.

Local Coherence We enforce local coherence of the per-pixel brightness values of the projectors to regularize the problem. To this end, we add a *Laplacian* smoothness constraint to our objective function:

$$E_{reg}(\mathbf{p}) = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} ||p_i - p_j||^2$$

$\mathcal{N}_i$ denotes the 1-neighborhood of the i-th pixel and N the total number of pixels. This constraint is especially important in regions where the ray configuration changes (i.e. red to blue in Figure 5a). In this area, sub-pixel calibration errors would become visible. We take special care that smoothing is not applied across depth discontinuities.

Brightness Balance Each surface point can potentially be illuminated by multiple projectors (see Figure 5a). $E_{balance}$ tries to control the contribution of the different projectors such that the best re-coloring result is obtained (see Figure 5b). To this end, we assess the quality of the footprint a certain pixel will generate:

$$E_{balance}(\mathbf{p}) = \sum_{i=1}^{N} \sum_{r \in \mathcal{R}_i} \Lambda(p_i, p_r)^2$$

Where $\mathcal{R}_i$ is the set of all re-projections of pixel p_i.

The ray quality decider Λ is defined by:

$$\Lambda(p_i, p_r) = \lambda_{smooth}(p_i) \cdot p_i - \lambda_{smooth}(p_r) \cdot p_r$$

Since a linear blending function is numerically unstable at the boundaries, our blending is based on the following continuous smooth step function:

$$\lambda_{smooth}(p_i) = 3 \cdot \lambda_{lin}(p_i)^2 - 2 \cdot \lambda_{lin}(p_i)^3$$

The decider takes into account that rays with a smaller footprint lead to sharper projections and therefore better re-coloring results. In our setup, we define $\lambda_{lin}(p_i)$ based on the angle between the surface normal and the negative incident ray direction:

$$\lambda_{lin}(p_i) = \frac{1}{|\mathcal{R}_i| - 1} \cdot \frac{\left(\sum_{r \in \mathcal{R}_i} \langle \mathbf{n}_r, \mathbf{i}_r \rangle\right) - \langle \mathbf{n}_i, \mathbf{i}_i \rangle}{\sum_{r \in \mathcal{R}_i} \langle \mathbf{n}_r, \mathbf{i}_r \rangle}$$

For two projectors this comes down to:

$$\lambda_{lin}(p_i) = \frac{\langle \mathbf{n}_j, \mathbf{i}_j \rangle}{\langle \mathbf{n}_i, \mathbf{i}_i \rangle + \langle \mathbf{n}_j, \mathbf{i}_j \rangle}$$

Where j is the re-projection index of pixel p_i into the other projector.

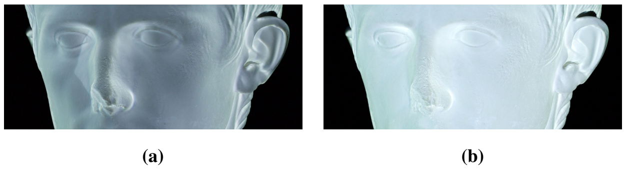


Figure 9: *(a) Shows a multi-projection ignoring gamma correction. The sum of projected light appears too dark. (b) This effect disappears if correct gamma correction is included.*

Boundary Constraints In theory we would need an infinite amount of light to compensate distance and Lambertian effects for extreme cases. Since E_{light} reflects this fact, the solver will output such extreme values. The controllable brightness range of real world projectors, however is limited to a certain range. Thus we need to constrain our per-pixel color values to $[0, 1]$:

$$E_{bound}(\mathbf{p}) = \sum_i \Phi(p_i)^2$$

To this end, we implement a soft box-constraint on the variables based on the following function:

$$\Phi(p_i) = \alpha \cdot (p_i - 0.5)^p$$

Experiments showed that $\alpha = 1.2$ and $p = 10$ leads to the best results, using a steeper function leads to numerical instabilities due to high derivatives at 0 and 1. On top of this soft box-constraint, we also use a parameter re-projection strategy in the optimization.

4.3 Linear Color Space

The objective of our algorithm is to generate correct (in our case rendered) colors on the surface of a target object. This is achieved by optimizing the contribution of pixel color intensities of the projected images using our objective function. Our solver assumes that colors behave linear when calculating the blending of two projectors (compare to Figure 5b). Since colors generated by our rendering are gamma corrected by the system and the projectors before they are projected, we need to compensate by applying an inverse gamma correction.

Figure 9a shows the result of an uncorrected setup. The desired surface color is set to a gray value of 0.7. Due to the non-linearity of the color space, the regions that are illuminated by both projectors appear too dark. In Figure 9b the gamma corrections are correctly applied and the surface appears uniformly lit.

4.4 Luminance Optimization

Solving our optimization problem for all color channels independently produces a large problem size. However, changes are only noticeable in the luminance of the optimized colors. We therefore adapt our objective function to compute a luminance such that the surface is lit with a constant brightness. In Section 4.1 we discussed the reserve we leave for compensation of Lambertian and distance effects. We also apply this to the luminance, setting the maximum value to 0.7, leaving a reserve of 0.3. This results in a new Energy E_{light} for our objective function:

$$E_{light}(\mathbf{p}) = ||\mathbf{T}\mathbf{p} - (0.7, 0.7, ...)^{\top})||_2^2$$

Due to this change, $\mathbf{p}$ now no longer denotes the pixel color, but the pixel's luminance weight. The actual pixel color is computed by multiplying the luminance weights p_i with the rendered target colors l_i.

5 Fast and Robust Bounded Optimization

Our proposed seamless re-projection objective gives rise to a non-linear bounded least squares problem in the unknown projected luminance weights $\mathbf{p}$. Interactive re-projection applications, e.g. re-texturing of rigidly moving objects, require the optimization to operate at real-time frame-rates. To this end, we propose a data-parallel optimization framework that leverages the horsepower of modern GPUs for the fast and robust solution of bounded least-squares problems. Our approach (see Algorithm 1) is a hybrid between a parallel Gauss-Newton (GN) solver and a fast and efficient parameter projection strategy. In contrast to the approach presented by Zollhöfer et al. [2014] for non-rigid surface tracking, our core solver directly supports *inequality* constraints. This is highly important in the context of our projector setup, since each luminance value has to be constrained to a physically re-producible subspace of $\mathbb{R}^N$.

5.1 Gauss-Newton and PCG Solver

Algorithm 1 Non-Linear Bounded Least-Squares Optimizer

```
Compute_Foreground_Masks();
Project_To_Other_Views();
Check_For_Occlusions();

$\mathbf{p}^0$ = Initialize_From_Last_Frame();
for $k = 1 \ldots K$ do
    Compute_RHS_And_Preconditioner($\mathbf{p}^{k-1}$);

    $\Delta\mathbf{p} = \mathbf{0}$;
    for $m = 1 \ldots M$ do
        $\Delta\mathbf{p}$ += PCG_Step($\mathbf{p}^{k-1}$);
    end for

    $\mathbf{p}^k$ = Reproject_To_Range($\mathbf{p}^{k-1} + \Delta\mathbf{p}$);
end for
```

The previously proposed re-projection objective is a highly non-linear bounded least squares problem. Without the *inequality* constraints, the problem is a non-linear least squares problem in the unknown vector of luminance values $\mathbf{p}$:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}}\, E(\mathbf{p})$$

Therefore, it can be rewritten in the following canonical form by re-naming all R scalar residuals consistently:

$$E(\mathbf{p}) = \sum_{i=0}^{R} r_i(\mathbf{p})^2$$

The notation can be further simplified by reformulating the objective in terms of its associated residual vector field $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^R$ that stacks all scalar residuals:

$$E(\mathbf{p}) = ||\mathbf{F}(\mathbf{p})||_2^2,\ \mathbf{F}(\mathbf{p}) = [\ \ldots,\ r_i(\mathbf{p}),\ \ldots\ ]^T$$

This minimization problem can be readily approached using the Gauss-Newton algorithm. To this end, we linearize the vector field around the last solution $\mathbf{p}^{k-1}$ using a first order Taylor expansion:

$$\mathbf{F}(\mathbf{p}^k) = \mathbf{F}(\mathbf{p}^{k-1}) + \mathbf{J}(\mathbf{p}^{k-1}) \cdot \Delta\mathbf{p}$$

with $\Delta\mathbf{p} = \mathbf{p}^k - \mathbf{p}^{k-1}$ being the linear parameter update and $\mathbf{J}$ denoting the Jacobian of $\mathbf{F}$. This gives rise to the following over-constrained linear least squares problem:

$$\Delta\mathbf{p}^* = \underset{\Delta\mathbf{p}}{\operatorname{argmin}}\, ||\mathbf{F}(\mathbf{p}^{k-1}) + \mathbf{J}(\mathbf{p}^{k-1}) \cdot \Delta\mathbf{p}||_2^2$$

The optimal linear updates $\Delta\mathbf{p}^*$ are computed by solving the associated normal equations using an iterative and fast GPU-based Preconditioned Conjugate Gradient (PCG) solver:

$$\mathbf{J}(\mathbf{p}^{k-1})^T\mathbf{J}(\mathbf{p}^{k-1}) \cdot \Delta\mathbf{p} = -\mathbf{J}(\mathbf{p}^{k-1})^T\mathbf{F}(\mathbf{p}^{k-1})$$

Note, that an iterative solution strategy is especially advantageous for the GN algorithm, since the system matrix $\mathbf{J}(\mathbf{p}^{k-1})^T\mathbf{J}(\mathbf{p}^{k-1})$ changes in each non-linear iteration due to its dependence on the old parameter estimate $\mathbf{p}^{k-1}$, see Algorithm 1. We solve for a sequence of solutions $\mathbf{p}^k$, starting from a good initial estimate $\mathbf{p}_0$, until convergence.

The Gauss-Newton approach is a derivative of Newton's method for minimizing non-linear least squares problems that does not require explicit second order derivatives, but still exhibits a super-linear convergence rate. Therefore, $\mathbf{J}^T\mathbf{J}$ can be seen as a first order approximation to the Hessian $\mathbf{H}$ of $\mathbf{F}$. To further speed up convergence, we use Jacobi/diagonal preconditioning.

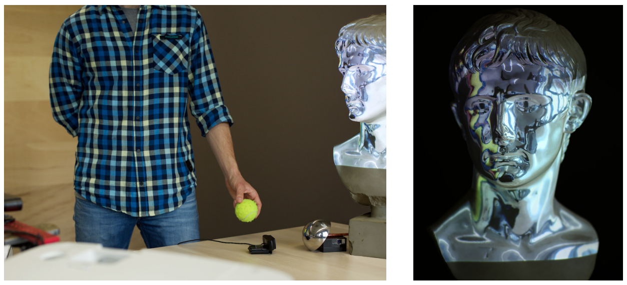
Structure of the Jacobian Matrix As we discussed in Section 4.2, the induced Jacobian $\mathbf{J}$ turns out to be sparse. To allow for real-time performance of the solver, we have to exploit this inherent sparsity in all computation steps. The sparsity of the Jacobian enables on-the-fly computation of the non-zero entries in the matrix-vector products of the PCG step. Parameters are initialized based on the previous frame.

Parameter Projection Since we are dealing with a non-linear bounded optimization problem, we have to keep the parameters in a reasonable range, such that they fulfill the given inequality constraints. To this end, we augment the GN procedure with a fast and efficient parameter projection strategy, see Algorithm 1. Therefore, we interleave each non-linear GN step with a projection step, bringing the parameters to their valid range.

Implementation Details Since the re-projection pattern is highly non-regular, we can not easily exploit shared memory as done in Wu et al. [2014]. We implemented our data-parallel bounded optimization strategy using `CUDA 7.0`. The cachable parts of the non-zero entries of the Jacobian are stored in a quasi *Compressed Row Storage* (CRS) matrix format with a fixed maximal row length. For this, we know the maximum of N_{row} non-zero entries per row, which equals the number of projectors. To aid a faster evaluation of the normal equations, we precompute some data that remains constant over all iterations. The multiplication with the system matrix in the `PCG_Step` is performed using two kernel calls: one for $\mathbf{J}^T$ and one for $\mathbf{J}$, respectively. This reduces fill-in and prevents the expansive $O(n^3)$ system matrix computation step. We use a parallel-prefix sum on a foreground mask to compute linearized thread indices. One thread is allocated per variable. In contrast to Zollhöfer et al. [2014] we use 3 scans: 1 scan on block level 512 blocks and two scans on warp level to compute optimal step-sizes. This allows us to support up to $2M$ variables which is enough for the two projected HD images, i.e. $2 \times (1280 \times 800) \approx 2M$ variables. It is easily possible to extend this to a larger number of projectors. Compared to Weber et al. [2013] and Zollhöfer et al. [2014], we also do not redundantly compute the step-sizes, this turned out to be beneficial on the newest graphics hardware.

6 Results

In the following section we will detail some example applications based on our real-time light transport optimizer. Please refer to the accompanying video to see the setup in action.



(a) (b)

Figure 10: *(a) The setup used for live environment mapping. A hemispherical mirror is captured by a color camera. (b) A photograph of the resulting projection onto the surface of the Augustus statue. The reflection of the tennis ball appears on the right cheek.*

Live Environment Map Having the possibility to completely reverse real-world shading on an illuminated object, we can change the appearance of the plaster statue seen in Figure 2. Our *Live Environment Mapping* captures a mirror half-sphere with a color camera. This live image is used as an environment map in our rendering, creating the illusion of a metallic surface (see Figure 10). The video shows that our system is capable of providing real-time updates without having an adverse effect on the frame-rate.

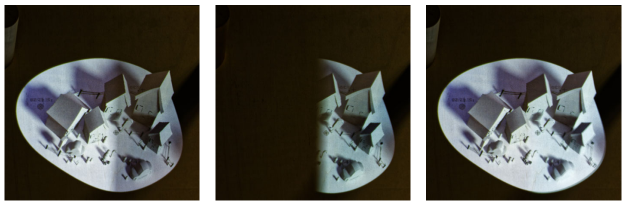
Viewer Tracking We integrated viewer tracking into our system to enable effects that are based on the spectator's position. This allows us to show correct specular lighting and proper parallax effects. The result of this can be seen in Figure 14. A virtual skull is rendered inside the statue that appears to be made of glass. The rendering adapts to the position of the spectator.

So far we have only showed our Augustus statue to demonstrate our system. This is due to the very high quality 3D-scan we have of this object. However, our algorithm works for arbitrary objects. Figures 14c and 11 show two additional demos of different objects.

N-Dynamic Shader-Lamps Our online system for solving multi-projection setups enables us to move the target object. Beyond this we also have the possibility to move the projection systems independently. Therefore, every projector is assigned to its own RGB-D camera for tracking purposes (see Figure 14d). Now, the user is able to take a projector, move it around and augment the regions he is interested in. As the system is solved globally, every device as well as the target object can be moved independently at the same time. Whenever a surface region is seen by more than one projector, our system takes care of the correct blending.

Since our tracking relies on active stereo RGB-D cameras, the use of two cameras in the same setup has a negative impact on tracking accuracy. However, only in very parallel orientations of cameras the tracking was lost in our experiments. In general the tracking showed only a slight degradation.

Performance We measured the performance of our system using the Augustus statue with about 300k faces. The tracking is performed using `DirectX` compute shaders. The rendering is based on the `NVidia Optix 3.8` framework, which gives us more flexibility for shooting shadow rays and the glass effects in our demos. Our system needs 14ms to compute the output colors used to illuminate the objects. It would easily be possible to switch to a forward rendering pipeline like `DirectX` or `OpenGL` in the future. This would give a substantial performance improvement. The solver runs on `CUDA 7.0` and is set to 5 GN and 5 PCG iterations



(a) (b) (c)

Figure 11: *This scene shows holographic rendering on a physical plane. (a) and (b) show the contribution of each projector to produce the final image (c).*

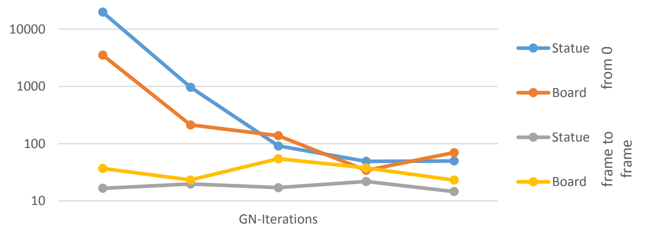
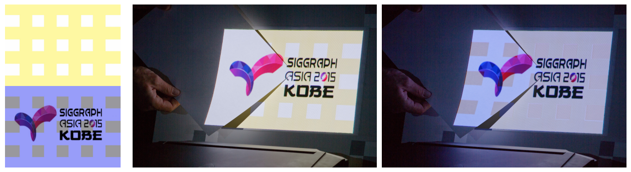


Figure 12: *Convergence (squared residuals) of our solver for the statue and board example on a logarithmic scale. The first two lines show convergence starting from scratch, the second two lines convergence frame to frame without moving the object.*

per frame. For the Augustus statue the solver finishes in 26ms. While this does not lead to complete convergence, we can use the result from the previous frame and the algorithm continues to converge. After about 5-8 frames, convergence is reached.

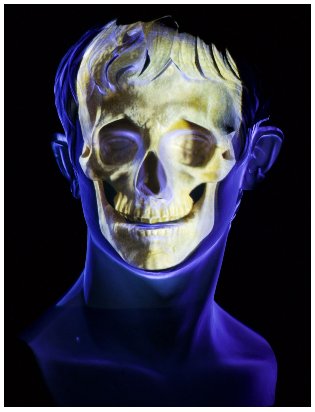
In our videos, a latency is visible while moving the object, or moving the projectors. Note that this latency is not due to our solver, but introduced by the tracking. Our tracking algorithm takes 4ms to find a solution. While our algorithm is responsible for some of the delay, the processing on the RGB-D camera itself contributes a major part. Using a faster tracking (see [Sueishi et al. 2015]) the user immersion of our system could be improved, at the cost of no longer using off-the-shelf hardware.

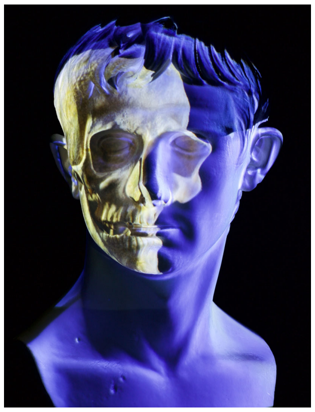
Convergence of the Solver Figure 12 shows the convergence of the solver for the statue and the board example. The solver reached a stationary and correct solution in all our experiments within a few GN iterations. In the first two lines, the solver starts from scratch, the second two lines show convergence frame to frame without moving the object. The occuring slight degradations in the residuum are easily explained by the linear approximation of the PCG and continuous slight changes from the object tracking.

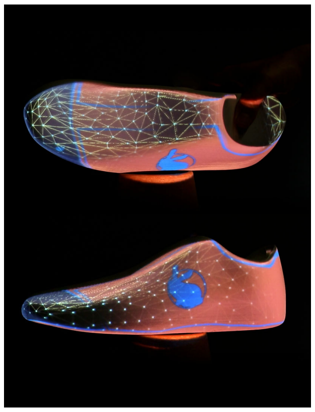


(a) (b) (c)

Figure 13: *(a) The real world surface texture and the projected image to compensate. (b) Projection of the Siggraph logo on the textured surface. Note the sheet of paper on the left, showing the projected image. (c) Projecting the corrective image onto the textured surface. Note the corrective pattern on the sheet of paper on the left.*


(a)


(b)


(c)

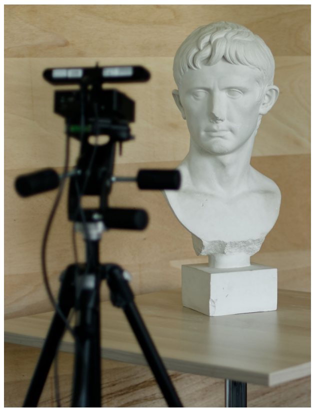

(d)

Figure 14: *(a) A skull rendered within the bust from the viewer's perspective. (b) The same rendering from a viewer position not aligned with the tracked viewer. (c) Our method applied to the last of a shoe showing a potential product design application. (d) A shader-lamp (projector with a rigidly mounted depth sensor).*

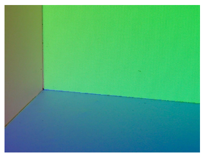
These fluctuations however are not visible in the final projection.

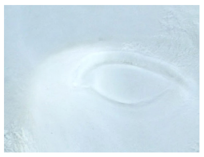
Limitations The primary focus of our work is calculating the light contribution of multiple projectors on arbitrary, known surfaces. However, there are surface effects that can impair the quality of projection mapping systems which are currently not handled. We are confident that our implementation is extensible enough to integrate published solutions to these problems. One example is compensating surface textures. We implemented a simple color correction approach to show this extensibility of our system (see Figure 13). The Siggraph logo is projected onto an object with the texture shown in Figure 13a (top). This results in the projection depicted in Figure 13b. Note the white sheet of paper we added to show the projected image on top of the textured surface. When compensating the surface color, we project the pattern shown in Figure 13a (bottom). The resulting effect is depicted in Figure 13c. In general, color compensation of textured surfaces is challenging. The original colors of the pattern cannot be determined easily and the internal color processing behaviour of the projectors is unknown as well. However, a multitude of literature on color compensation for projection mapping exists [Bimber et al. 2005a; Law et al. 2011; Grundhöfer 2013].

We currently do not correct for interreflection introduced by light from the projectors. This leads to color bleeding (see Figure 15a) and overexposed areas in concavities of the target object (see Figure 15b). One simple approach to alleviate this, is a less reflective coating for the target object. However, we plan to use our system for museum applications, where this is not possible. Hence, we need to incorporate handling interreflections into our solver, which introduces an enormous additional computational effort. We leave this for a future work.

The third limitation regarding the target objects' surfaces is their reflectance. Currently we assume Lambertian surfaces. For conductive and mirrorlike materials it is nearly impossible to perform projection mapping. For rougher, non-conductive materials however, we are confident that a compensation of specular highlights can be integrated into our system. Since this imposes a hard challenge, we also leave this for future work.

Another limitation of our system can be seen in Figure 15c. If the 3D model of the target object is not accurate enough, artifacts will appear. This is especially problematic for objects with larger occlusions or shadow casters, as these amplify the effect. In Figure 15c the pole on the board is at slightly different positions in the 3D model and the real world. This leads to visible artifacts along the shadow borders (red arrows), as the second projector assumes the shadow at a slightly different position. Note, the pole's soft shadow (blue arrows) belongs to the rendering and is not an artifact.


(a)


(b)

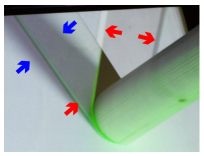

(c)

Figure 15: *(a) Since we do not correct for interreflection, color bleeding is visible. (b) Some areas appear too bright. (c) If the scan quality of the target object is not sufficient, the missalignment becomes apparent (red arrows), especially for an object casting a large shadow, as shown in. Note, that the soft shadow (blue arrows) belongs to the rendering and is not an artifact.*

7 Discussion

In the previous section and the accompanying video we have shown that with our implemented complete augmented reality system we are able to create immersive user experiences. The seams between projectors are invisible and shadowed regions are filled in real-time. This enables the user to manipulate the complete setup while the projection always shows correctly blended results. It would not be possible to create this depth of immersion with only a single projector as the resulting self-shadowing and limited field of view breaks the illusion.

While moving the objects in the video small latency artifacts, especially around the nose (the red region in Figure 5a) occur. Due to this and the convergence in *only* 5-8 frames, the reader might argue that our system is not real-time capable. However, these small artifacts are only visible during movement in case of a very uniform projected texture (the Phong example in our video) and even in these cases they immediately disappear when the object is stationary again. For projected textures with even a little bit of detail, the effect is imperceptible. Also the biggest steps towards convergence take place in the first 1-2 frames.

Another advantage of our system is the solver's independence from the surface color (see Section 4.3). By this means, we can project animations onto the object that run arbitrarily fast, without the multi-projection optimization becoming a limiting factor. Also, a smoothing across the optimized luminance weights will not affect the sharpness of the final projection.

Note that for our demonstration setup we chose a rather extreme configuration of projectors to also show that our system handles corner cases well. However, this introduces some small artifacts. The projectors illuminate the object from two sides with a vertical offset (see Figure 2 and Figure 1, right). This results in visible edges on the object (e.g. the left temple of our Augustus statue) as the upper projector can not reach this region due to self-shadowing (see Figure 5). For a real-world setup a more conservative configuration would be chosen to prevent such cases.

A limiting factor of our system proved to be the tracking. This is visible in the *ghosting* while moving the objects or shader-lamps in our video. The projections no longer look like they are *glued* to the object. This however is not an artifact of the multi-projection solver.

Our dynamic multi-projection mapping system can be adapted to handle most projection mapping scenarios. However, under some circumstances (e.g. scenes with known object movements or static scenes) our more complex system is not needed. Also for scenarios, which require an extremely fast tracking, better solutions exist. In general however, to create the level of immersion presented throughout this paper, multiple projectors are required to illuminate the complex geometry. While other works support multiple projectors for static geometry, blending mostly is performed in the space of the projector. To achieve the presented quality, our per-pixel blending of projector light on the target geometry is crucial. With the additional real-time capabilities of our system, we are able to handle complex dynamic scenes with an unpreceded quality. The possiblity for projecting on dynamic scenes becomes beneficial even in seemingly static setups. These setups' calibrations often tend to deteriorate due to accidental movement of the objects. Our system is unsusceptible with respect to such movements.

8 Conclusion and Outlook

Our system leaves a lot of potential for the future. Not only the previously described limitations could be remedied, but also completely new applications come to mind. For example, the inherently dynamic setting of the *Augmented Reality Sandbox* presented by Reed et al. [2014] could be extended to multiple projectors using our system. Another very interesting field of future research could be bringing works of lighting estimation and relighting in images (see [Cossairt et al. 2008; Knecht et al. 2010; Nowrouzezahrai et al. 2011; Uday Mehta et al. 2015]) to projection mapping onto real world objects. Also the light coming from the projector indirectly illuminating the environment could be counteracted with such techniques.

In this work we have presented a novel method for dynamically augmenting physical scenes with multiple projectors. Our objective function incorporates the physical effects of light, the balancing of rays based on their expected projection quality and regularization terms. We solve this function in a non-linear least squares manner using a parallel Gauss-Newton solver, reaching real-time frame-rates. Using this system we can freely move the illuminated objects and projectors while maintaining the correct illumination at all times, without the need for markers on the objects. This allows us to create immersive experiences which can be seen best in our video.

Acknowledgements

We want to thank . . .
. . . Philip Saa and Tobias Kaiser for the voiceover.
. . . Martin Boss from the Collection of Antiquities of the University Erlangen-Nuremberg for the Augustus Bust.
. . . Ben for all the great advice on low level GPU stuff.
This research is funded by the ERC Starting Grant 335545 CapReal and German Research Foundation (DFG) grant GRK-1773.

References

AKAOKA, E., GINN, T., AND VERTEGAAL, R. 2010. Displayobjects: Prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction*, ACM, New York, NY, USA, TEI '10, 49–56.

BANDYOPADHYAY, D., RASKAR, R., AND FUCHS, H. 2001. Dynamic shader lamps: Painting on movable objects. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, IEEE, 207–216.

BIMBER, O., AND RASKAR, R. 2005. *Spatial augmented reality: merging real and virtual worlds*. AK Peters Wellesley, MA.

BIMBER, O., EMMERLING, A., AND KLEMMER, T. 2005. Embedded entertainment with smart projectors. *Computer 38*, 1.

BIMBER, O., WETZSTEIN, G., EMMERLING, A., AND NITSCHKE, C. 2005. Enabling view-dependent stereoscopic projection in real environments. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, 14–23.

BIMBER, O., IWAI, D., WETZSTEIN, G., AND GRUNDHÖFER, A. 2008. The visual computing of projector-camera systems. In *Computer Graphics Forum*, vol. 27, Wiley Online Library.

BROWN, M., MAJUMDER, A., AND YANG, R. 2005. Camera-based calibration techniques for seamless multiprojector displays. *Visualization and Computer Graphics, IEEE Transactions on 11*, 2, 193–206.

CAO, X., AND BALAKRISHNAN, R. 2006. Interacting with dynamically defined information spaces using a handheld projector and a pen. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM, 225–234.

COSSAIRT, O., NAYAR, S. K., AND RAMAMOORTHI, R. 2008. Light field transfer: Global illumination between real and synthetic objects. *ACM Trans. on Graphics* (Aug).

FUJII, K., GROSSBERG, M. D., AND NAYAR, S. K. 2005. A projector-camera system with real-time photometric adaptation for dynamic environments. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, IEEE, 814–821.

GRUNDHÖFER, A., AND BIMBER, O. 2008. Real-time adaptive radiometric compensation. *Visualization and Computer Graphics, IEEE Transactions on 14*, 1, 97–108.

GRUNDHÖFER, A. 2013. Practical non-linear photometric projector compensation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, IEEE.

HARRISON, C., BENKO, H., AND WILSON, A. D. 2011. Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM, 441–450.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., ET AL. 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, ACM, 559–568.

JONES, B. R., BENKO, H., OFEK, E., AND WILSON, A. D. 2013. Illumiroom: Peripheral projected illusions for interactive experiences. In *ACM SIGGRAPH 2013 Emerging Technologies*, ACM, New York, NY, USA, SIGGRAPH '13, 7:1–7:1.

JONES, B., SODHI, R., MURDOCK, M., MEHRA, R., BENKO, H., WILSON, A., OFEK, E., MACINTYRE, B., RAGHUVANSHI, N., AND SHAPIRA, L. 2014. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ACM, NY, USA, UIST '14.

KASPAR, A., AND DENG, B. 2015. Real-time deformation of constrained meshes using gpu. In *GPU Computing and Applications*, Y. Cai and S. See, Eds. Springer Singapore, 15–34.

KNECHT, M., TRAXLER, C., MATTAUSCH, O., PURGATHOFER, W., AND WIMMER, M. 2010. Differential instant radiosity for mixed reality. In *Proceedings of the 2010 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2010)*, 99–107. Best Paper Award!

KONIECZNY, J., AND MEYER, G. W. 2006. Material and color design using projectors. In *CGIV'06*, 438–442.

LAW, A. J., ALIAGA, D. G., SAJADI, B., MAJUMDER, A., AND PIZLO, Z. 2011. Perceptually based appearance modification for compliant appearance editing. In *Computer Graphics Forum*, vol. 30, Wiley Online Library, 2288–2300.

LEE, J. C., DIETZ, P. H., MAYNES-AMINZADE, D., RASKAR, R., AND HUDSON, S. E. 2004. Automatic projector calibration with embedded light sensors. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, ACM, 123–126.

LEE, J. C., HUDSON, S. E., AND TSE, E. 2008. Foldable interactive displays. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, 287–290.

MAJUMDER, A., AND BROWN, M. S. 2007. *Practical multi-projector display design*. AK Peters USA.

MOLYNEAUX, D., IZADI, S., KIM, D., HILLIGES, O., HODGES, S., CAO, X., BUTLER, A., AND GELLERSEN, H. 2012. Interactive environment-aware handheld projectors for pervasive computing spaces. In *Pervasive Computing*. Springer, 197–215.

NEWCOMBE, R. A., DAVISON, A. J., IZADI, S., KOHLI, P., HILLIGES, O., SHOTTON, J., MOLYNEAUX, D., HODGES, S., KIM, D., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, IEEE.

NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG 32*, 6, 169.

NOWROUZEZAHRAI, D., GEIGER, S., MITCHELL, K., SUMNER, R., JAROSZ, W., AND GROSS, M. 2011. Light factorization for mixed-frequency shadows in augmented reality. In *10th IEEE International Symposium on Mixed and Augmented Reality (Proceedings of ISMAR 2011)*.

PINHANEZ, C. 2001. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Ubicomp 2001: Ubiquitous Computing*, Springer, 315–331.

RASKAR, R., WELCH, G., CUTTS, M., LAKE, A., STESIN, L., AND FUCHS, H. 1998. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 179–188.

RASKAR, R., WELCH, G., LOW, K.-L., AND BANDYOPADHYAY, D. 2001. Shader lamps: Animating real objects with image-based illumination. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, Springer-Verlag, London.

REED, S. E., KREYLOS, O., HSI, S., KELLOGG, L. H., SCHLADOW, G., YIKILMAZ, M. B., SEGALE, H., SILVERMAN, J., YALOWITZ, S., AND SATO, E. 2014. Shaping Watersheds Exhibit: An Interactive, Augmented Reality Sandbox for Advancing Earth Science Education. *AGU Fall Meeting Abstracts* (Dec.), A1.

RESCH, C., KEITLER, P., AND KLINKER, G. 2014. Sticky projections—a new approach to interactive shader lamp tracking. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, IEEE, 151–156.

SHENG, Y., YAPO, T., YOUNG, C., AND CUTLER, B. 2009. Virtual heliodon: Spatially augmented reality for architectural daylighting design. In *Virtual Reality Conference, 2009. IEEE*.

SHUM, H.-Y., CHAN, S.-C., AND KANG, S. B. 2008. *Image-based rendering*. Springer Science & Business Media.

SUEISHI, T., OKU, H., AND ISHIKAWA, M. 2015. Robust high-speed tracking against illumination changes for dynamic projection mapping. In *IEEE Virtual Reality Conference (VR2015)*.

UDAY MEHTA, S., KIM, K., PAJAK, D., PULLI, K., KAUTZ, J., AND RAMAMOORTHI, R. 2015. Filtering environment illumination for interactive physically-based rendering in mixed reality. Tech. Rep. UCB/EECS-2015-164, EECS Department, University of California, Berkeley, Jun.

WEBER, D., BENDER, J., SCHNOES, M., STORK, A., AND FELLNER, D. 2013. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum 32*, 1, 16–26.

WETZSTEIN, G., BIMBER, O., ET AL. 2007. Radiometric compensation through inverse light transport. In *Pacific conference on computer graphics and applications*, 391–399.

WILSON, A. D., AND BENKO, H. 2010. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM.

WU, C., ZOLLHÖFER, M., NIESSNER, M., STAMMINGER, M., IZADI, S., AND THEOBALT, C. 2014. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics (TOG) 33*, 6.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 11 (Nov), 1330–1334.

ZOLLHÖFER, M., NIESSNER, M., IZADI, S., REHMANN, C., ZACH, C., FISHER, M., WU, C., FITZGIBBON, A., LOOP, C., THEOBALT, C., AND STAMMINGER, M. 2014. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG) 33*, 4.