applied sciences

MDPI

Article

Per-Pixel Manifold-Based Color Calibration Technique

Stanisław Gardecki [1], Krzysztof Wegner [2], Tomasz Grajek [3,*] and Krzysztof Klimaszewski [3]

[1] Institute of Robotics and Machine Intelligence, Poznan University of Technology, pl. M. Skłodowskiej-Curie 5, 60-965 Poznan, Poland; stanislaw.gardecki@put.poznan.pl
[2] Mucha sp. z o.o., ul. Szelągowska 17, 61-626 Poznan, Poland; kwegner@mucha.be
[3] Institute of Multimedia Telecommunications, Poznan University of Technology, pl. M. Skłodowskiej-Curie 5, 60-965 Poznan, Poland; krzysztof.klimaszewski@put.poznan.pl
* Correspondence: tomasz.grajek@put.poznan.pl

Abstract: In this paper, we present a method for obtaining a manifold color correction transform for multiview images. The method can be applied in various scenarios, for correcting the colors of stitched images, adjusting the colors of images obtained in different lighting conditions, and performing virtual view synthesis based on images taken by different cameras or in different conditions. The provided derivation allows us to use the method to correct regular RGB images. The provided solution is specified as a transform matrix that provides the pixel-specific color transformation for each pixel and therefore is more general than the methods described in the literature, which only provide the transformed images without explicitly providing the transform. By providing the transform for each pixel separately, we can introduce a smoothness constraint based on the transformation similarity for neighboring pixels, a feature that is not present in the available literature.

Keywords: color correction; multiview; multi-camera; virtual views

check for updates

Academic Editor: Andrea Prati

Received: 6 February 2025
Revised: 10 March 2025
Accepted: 10 March 2025
Published: 13 March 2025

Citation: Gardecki, S.; Wegner, K.; Grajek, T.; Klimaszewski, K. Per-Pixel Manifold-Based Color Calibration Technique. *Appl. Sci.* **2025**, *15*, 3128. https://doi.org/10.3390/app15063128

1. Introduction

In many applications, we use multiple acquisition devices to record images of the same object. This results in multiple images of the same object with different gamut/color characteristics. Even a single camera can capture images with different colorings due to changing weather conditions. This variation affects image or object recognition algorithms, panorama creation, image stitching, virtual view creation, or Neural Radiance Field (NeRF) [1,2] estimation. To address these differences, color calibration is necessary. It allows us to find a mapping of colors from one image to another. Various methods exist to achieve this mapping or transformation. Nearly all of them assume constant color transformation across the whole image. In this paper, we are presenting a novel manifold color calibration method that allows per-pixel color transformation estimation between pairs of images, even when in partially occluded image pairs.

For example, in low-light conditions, we can take a dark color image and a near-infrared image of the same object [3], or take a shot with and without a flash [4] and then try to combine both of the images together.

Another example would be applications in view synthesis [5,6], where we can create a non-existing view of the scene from the viewpoint of so-called virtual cameras based on two (or more) images captured by real cameras recording the given scene. Even if both cameras are identical (which is not always possible), they can produce images with different color characteristics. This could happen even due to the different viewpoints from which the images were taken.

Although the problem is the most obvious for multi-camera scenarios, using a single camera can also lead to the different colorings of the captured images, for example, due to the changing weather conditions. In applications such as image/object recognition, this could significantly influence the performance of the algorithms used.

Yet another example is panorama creation or/and image stitching [7]. Where we can stitch or combine multiple images captured either by the same image sensor or by multiple sensors into one big continuous image.

In all the abovementioned situations, we can color calibrate the images in order to minimize the artifacts occurring due to the color mismatch. This can be performed by finding the mapping of colors from one image to the other.

There already exist many methods of finding such mappings or transformations, which map the color of all pixels from one image to the other.

The significance of color correction manifests itself in many applications, where different methods were demonstrated to effectively improve the performance of the whole system. One such example is multiview video compression, as evidenced in [8–11]. In those papers, the authors demonstrate an improvement in the coding of the multiview video after applying a preprocessing step of color matching of the sequences due to the improved accuracy of motion compensation and inter-view prediction.

The methods used to perform the color correction that is described in the literature can be based on histogram equalization [11–13]. A popular method is also using a single scaling factor, correction coefficient, or a polynomial to correct for the color differences [14–17]. A low-dimensional matrix correction can also be used [18]. Another way to color correct the images is to use the color transfer technique [19]. In recent years, neural networks have been used to perform similar tasks and can be used to perform color transfer as well; an example of this color transfer method can be found in [20].

In practice, it is important to not only correct the colors of the images but also maintain sufficient quality of the corrected images without degrading the contrast of the processed images [21]. Thus, the correction algorithm should also consider the texture/contrast information in the processed images.

Another way of performing the color correction is to solve a manifold optimization problem defined globally for the entire image. Such an approach can be found in [8,22,23]. The authors of [8,23] define color correction as an optimization problem with priors both from the source image and from the target image. From the source image, the priors are color characteristics and color distribution, while from the target image, the priors tend to preserve the spatial and temporal structure of the image, after color correction. Similarly, the paper [22] introduces a method for global optimization of pixel values for large baseline multiview sequences, preserving the structure and spatio-temporal consistency.

The paper [24] introduces a parametric linear method and a nonparametric nonlinear method to deal with different color changes between pairs of images. The method proposed is a modified Laplacian Eigenmaps, which is a nonlinear manifold learning approach. The used goal function for color transfer is a quadratic cost function with a quadratic regularizer. The described method only searches for image sample values of a target image, matching the color characteristics of a source image.

In many applications, only a fraction of the pixels from one image directly correspond to any other image. For example, in image stitching, images only partially overlap. Similarly, in view synthesis, synthesized images from a virtual camera contain holes caused by disocclusions, where some parts of the scene are not visible due to perspective changes between the source view and the virtual view created with the source view data. Thus, only part of the real image has corresponding pixels in the virtual image. Yet we can obtain color transformation for all of the pixels in the images in order to seamlessly combine them together.

Color characteristics can vary across images due to effects like ambient occlusion, nonuniform lighting, shadows, and non-Lambertian reflections. Most color calibration techniques do not work reliably for these cases, only averaging color distribution and color characteristics across entire images. But in many applications, we can retain that non-uniformity during the color transformation from one image to the other, since they are inherent for some objects and materials. We can preserve these features for a color-corrected view.

To obtain a proper transformation that preserves the aforementioned effects, one needs to estimate per-pixel color transformation from one image to the other. An important factor of novelty of the work presented in this paper is the fact that the optimization algorithm does not work on the pixel values themselves, but rather on the transformations applied to those pixels. In this way, our method can use optimization constraints that do not refer to pixel values, but rather to the transformations that are applied to the pixel values. In this way, it is possible to obtain a smooth transformation field that can better capture the specific properties of a given scene. Additionally, the presented method includes parameters that can be used to adjust the smoothness of the transformation manifold. We also provide values of the parameters that were found to provide good results for all the tested sequences.

In this paper, we describe such a method and provide experimental results obtained with the use of the method.

In this paper, we first present the formal description of the optimization problem in a simple abstract space and then provide an analytical solution for this space.

From there, we move on to a more complex case, where the problem is defined in a 2D grid, normally used in image and video processing. We present the problem formulation in this 2D grid space and provide a solution for such a space.

The final part of derivation considers the occlusions and non-overlapping parts of the scene, the phenomena usually encountered in practical applications. The solution provided for the 2D grid is therefore modified to deal with occlusions and provide accurate transformation for those parts of the image, even in case of the absence of correspondence data.

The rest of the paper is organized as follows: Section 2 defines formally the actual problem, Section 3 provides an analytical solution, Section 4 describes the same problem in a 2D grid that is commonly used in digital images, Section 5 provides the solution for a 2D grid, Section 6 provides information about the application for occluded parts of the virtual image, Section 7 provides experimental results performed with the use of the described method, and Section 8 summarizes the entire paper with conclusions.

2. Problem Definition

Let us assume we have two images, left and right, that represent the same viewpoint of a scene. Let us assume that $L^p = \begin{bmatrix} l_1^p & l_2^p & \cdots & l_i^p & \cdots & l_n^p \end{bmatrix}^T$ and $R^p = \begin{bmatrix} r_1^p & r_2^p & \cdots & r_i^p & \cdots & r_n^p \end{bmatrix}^T$ are n-component vectors of colors of pixel p in the left and right images. The n denotes the number of components in an image increased by 1 since the operations are performed in homogenous coordinates, which offer more flexibility over regular cartesian coordinates. Therefore, commonly n will be 4 for homogenous coordinates of RGB color space.

We would like to estimate a manifold of per-pixel transformation A^p such that for all pixels p

$$R^p = A^p \cdot L^p \tag{1}$$

where

$$A^p = \begin{bmatrix} a_{11}^p & a_{12}^p & \cdots & a_{1j}^p & \cdots & a_{1n}^p \\ a_{21}^p & a_{22}^p & \cdots & a_{2j}^p & \cdots & a_{2n}^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1}^p & a_{i2}^p & \cdots & a_{ij}^p & \cdots & a_{in}^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^p & a_{n2}^p & \cdots & a_{nj}^p & \cdots & a_{nn}^p \end{bmatrix} \tag{2}$$

For an RGB image, the matrix A^p will be a 4×4 (since we are operating in homogenous coordinates, as explained before), and there is a p such matrix, one for every pixel in the image (p is the number of pixels in the image). Please note that at this stage, the matrix is agnostic to the way the pixels are organized in the image since p is merely an index.

In the further part of the derivation, we will continue to use n as the size of the matrix A^p since the method can be applied not only to RGB images but potentially also to multispectral and hyperspectral images.

The problem considered in this paper is to find the transformation A^p for a given pair of images, e.g., coming from a pair of cameras in stereoscopic applications.

3. Derivation of the Solution

Because the system of Equation (1) for every pixel p is under-determined, we can define the minimization problem with a goal function S. We can transform color L^p of left image pixel p through A^p as close as possible to the color R^p of the right image pixel with the same pixel index p

$$S_1 = \sum_p \|A^p \cdot L^p - R^p\|^2 \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm.

Moreover, we can have a smooth manifold A^p, which means that the two transformations A^p and A^q for two neighboring pixels p or q should be similar. It means that both transformations applied to one (any) of those pixels should give the same result (or at least as close as possible). We denote o as any of the two pixels p or q

$$S_2 = \sum_{p,q} \|A^p \cdot L^o - A^q \cdot L^o\|^2 \tag{4}$$

So, finally, we can minimize the difference (3). We compute the derivative of (3) with respect to every element a_{ij}^p of manifold A^p

$$\frac{\partial S_1}{\partial a_{ij}^p} = \frac{\partial \left(\sum_{p'} \|A^{p'} \cdot L^{p'} - R^{p'}\|^2 \right)}{\partial a_{ij}^p} \tag{5}$$

All derivatives with respect to a_{ij}^p from the sum in (5) except $p' = p$ are equal to 0 from which follows that

$$\frac{\partial S_1}{\partial a_{ij}^p} = \frac{\partial \left(\|A^p \cdot L^p - R^p\|^2 \right)}{\partial a_{ij}^p} \tag{6}$$

Lemma 1. *Assume some vector* $B = \begin{bmatrix} b_1 & b_2 & \cdots & b_i & \cdots & b_n \end{bmatrix}^T$ *then the square of Euclidean norm is:*

$$\|B\|^2 = \left(\sqrt{\sum_{i=1}^n b_i^2} \right)^2 = \sum_{i=1}^n b_i^2 = B^T \cdot B \tag{7}$$

So, any derivative of a square of Euclidean norm of this vector can be expressed as:

$$\frac{\partial\|B\|^2}{\partial\cdot} = \frac{\partial\left(B^T\cdot B\right)}{\partial\cdot} = \frac{\partial\left(\sum_{i=1}^{n} b_i^2\right)}{\partial\cdot} = \frac{\sum_{i=1}^{n} 2\cdot b_i\cdot\partial b_i}{\partial\cdot} = 2\cdot B^T\cdot\frac{\partial B}{\partial\cdot} \tag{8}$$

Using Lemma 1 in (6), we have

$$\begin{aligned}\frac{\partial S_1}{\partial a_{ij}^p} &= \frac{\partial\left(\|A^p\cdot L^p - R^p\|^2\right)}{\partial a_{ij}^p} = 2\cdot\left(A^p\cdot L^p - R^p\right)^T\cdot\frac{\partial\left(A^p\cdot L^p - R^p\right)}{\partial a_{ij}^p} \\ &= 2\cdot\left(A^p\cdot L^p - R^p\right)^T\cdot\frac{\partial A^p}{\partial a_{ij}^p}\cdot L^p\end{aligned} \tag{9}$$

Lemma 2. *Let us define a vector $M_i = \begin{bmatrix} 0 & 0 & \cdots & 1 & \cdots & 0\end{bmatrix}^T$ in which all components are 0 except for i-th, which is 1. Now, we can calculate the derivative of matrix A with respect to its element a_{ij}*

$$\begin{aligned}\frac{\partial A}{\partial a_{ij}} &= \frac{\partial\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn}\end{bmatrix}}{\partial a_{ij}} \\ &= \begin{bmatrix} \frac{\partial a_{11}}{\partial a_{ij}} & \frac{\partial a_{12}}{\partial a_{ij}} & \cdots & \frac{\partial a_{1j}}{\partial a_{ij}} & \cdots & \frac{\partial a_{1n}}{\partial a_{ij}} \\ \frac{\partial a_{21}}{\partial a_{ij}} & \frac{\partial a_{22}}{\partial a_{ij}} & \cdots & \frac{\partial a_{2j}}{\partial a_{ij}} & \cdots & \frac{\partial a_{2n}}{\partial a_{ij}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_{i1}}{\partial a_{ij}} & \frac{\partial a_{i2}}{\partial a_{ij}} & \cdots & \frac{\partial a_{ij}}{\partial a_{ij}} & \cdots & \frac{\partial a_{in}}{\partial a_{ij}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_{n1}}{\partial a_{ij}} & \frac{\partial a_{n2}}{\partial a_{ij}} & \cdots & \frac{\partial a_{nj}}{\partial a_{ij}} & \cdots & \frac{\partial a_{nn}}{\partial a_{ij}}\end{bmatrix}\end{aligned} \tag{10a}$$

Because all matrix components are independent of a_{ij} except a_{ij} element, so

$$\begin{aligned}\frac{\partial A}{\partial a_{ij}} &= \begin{bmatrix} \frac{\partial a_{11}}{\partial a_{ij}} & \frac{\partial a_{12}}{\partial a_{ij}} & \cdots & \frac{\partial a_{1j}}{\partial a_{ij}} & \cdots & \frac{\partial a_{1n}}{\partial a_{ij}} \\ \frac{\partial a_{21}}{\partial a_{ij}} & \frac{\partial a_{22}}{\partial a_{ij}} & \cdots & \frac{\partial a_{2j}}{\partial a_{ij}} & \cdots & \frac{\partial a_{2n}}{\partial a_{ij}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_{i1}}{\partial a_{ij}} & \frac{\partial a_{i2}}{\partial a_{ij}} & \cdots & \frac{\partial a_{ij}}{\partial a_{ij}} & \cdots & \frac{\partial a_{in}}{\partial a_{ij}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_{n1}}{\partial a_{ij}} & \frac{\partial a_{n2}}{\partial a_{ij}} & \cdots & \frac{\partial a_{nj}}{\partial a_{ij}} & \cdots & \frac{\partial a_{nn}}{\partial a_{ij}}\end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0\end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0\end{bmatrix}\cdot\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0\end{bmatrix}^T = M_i\cdot M_j^T\end{aligned} \tag{10b}$$

So, using Lemma 2 in (9), we have

$$\begin{aligned}\frac{\partial S_1}{\partial a_{ij}^p} &= 2\cdot(A^p\cdot L^p - R^p)^T\cdot\frac{\partial A^p}{\partial a_{ij}^p}\cdot L^p\\ &= 2\cdot(A^p\cdot L^p - R^p)^T\cdot M_i\cdot M_j^T\cdot L^p\\ &= 2\cdot M_i^T\cdot(A^p\cdot L^p - R^p)\cdot M_j^T\cdot L^p\\ &= 2\cdot\left(M_i^T\cdot A^p\cdot L^p - M_i^T\cdot R^p\right)\cdot M_j^T\cdot L^p\end{aligned} \tag{11}$$

Lemma 3. *Let us define a vector $M_i = \begin{bmatrix}0 & 0 & \cdots & 1 & \cdots & 0\end{bmatrix}^T$ in which all components are 0 except for i-th, which is 1. Now, we can define the selection of i-th component as:*

$$M_i^T\cdot L = \begin{bmatrix}0 & 0 & \cdots & 1 & \cdots & 0\end{bmatrix}\cdot\begin{bmatrix}l_1 & l_2 & \cdots & l_i & \cdots & l_n\end{bmatrix}^T = l_i \tag{12}$$

where L is an n component vector.

$$M_i^T\cdot A = \begin{bmatrix}0 & 0 & \cdots & 1 & \cdots & 0\end{bmatrix}\cdot\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn}\end{bmatrix} = a_{i:} \tag{13}$$

where A is n by n matrix and $a_{i:}$ mean i-th row of a matrix A.

Using Lemma 3 on (11), we have

$$\begin{aligned}\frac{\partial S_1}{\partial a_{ij}^p} &= 2\cdot\left(M_i^T\cdot A^p\cdot L^p - M_i^T\cdot R^p\right)\cdot M_j^T\cdot L^p = 2\cdot\left(a_{i:}^p\cdot L^p - r_i^p\right)\cdot l_j^p\\ &= 2\cdot\left(\sum_k a_{ik}^p\cdot l_k^p - r_i^p\right)\cdot l_j^p = 2\cdot\sum_k a_{ik}^p\cdot l_k^p\cdot l_j^p - 2\cdot r_i^p\cdot l_j^p\end{aligned} \tag{14}$$

Similarly, we would like to minimize (4). So, we compute the derivative of (4) with respect to every element a_{ij}^p of manifold A^p

$$\frac{\partial S_2}{\partial a_{ij}^p} = \frac{\partial\left(\sum_{p',q'}\|A^{p'}\cdot L^o - A^{q'}\cdot L^o\|^2\right)}{\partial a_{ij}^p} \tag{15}$$

All derivatives with respect to a_{ij}^p from the sum in (15) except for the pair $p' = p$ and $q' = q$ are equal to 0, so

$$\frac{\partial S_2}{\partial a_{ij}^p} = \frac{\partial\left(\|A^p\cdot L^o - A^q\cdot L^o\|^2\right)}{\partial a_{ij}^p} \tag{16}$$

Using Lemma 1 in (16), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a_{ij}^p} &= \frac{\partial\left(\|A^p\cdot L^o - A^q\cdot L^o\|^2\right)}{\partial a_{ij}^p} = 2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial(A^p\cdot L^o - A^q\cdot L^o)}{\partial a_{ij}^p}\\ &= 2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial A^p}{\partial a_{ij}^p}\cdot L^o\end{aligned} \tag{17}$$

Using Lemma 2 in (17), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a^p_{ij}} &= 2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial A^p}{\partial a^p_{ij}}\cdot L^o = 2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot M_i\cdot M_j^T\cdot L^o\\ &= 2\cdot M_i^T\cdot(A^p\cdot L^o - A^q\cdot L^o)\cdot M_j^T\cdot L^o = 2\cdot\left(M_i^T\cdot A^p\cdot L^o - M_i^T\cdot A^q\cdot L^o\right)\cdot M_j^T\cdot L^o\end{aligned} \tag{18}$$

Using Lemma 3 on (18), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a^p_{ij}} &= 2\cdot\left(M_i^T\cdot A^p\cdot L^o - M_i^T\cdot A^q\cdot L^o\right)\cdot M_j^T\cdot L^o\\ &= 2\cdot\left(a^p_{i:}\cdot L^o - a^q_{i:}\cdot L^o\right)\cdot l^o_j\\ &= 2\cdot\left(\sum_k a^p_{ik}\cdot l^o_k - \sum_k a^q_{ik}\cdot l^o_k\right)\cdot l^o_j\\ &= 2\cdot\sum_k a^p_{ik}\cdot l^o_k\cdot l^o_j - 2\cdot\sum_k a^q_{ik}\cdot l^o_k\cdot l^o_j\end{aligned} \tag{19}$$

Similarly, we compute the derivative of (4) with respect to every element a^q_{ij} of manifold A^p

$$\frac{\partial S_2}{\partial a^q_{ij}} = \frac{\partial\left(\sum_{p',q'}\|A^{p'}\cdot L^o - A^{q'}\cdot L^o\|^2\right)}{\partial a^q_{ij}} \tag{20}$$

All derivatives with respect to a^q_{ij} from the sum in (20) except for the pair $p' = p$ and $q' = q$ are equal to 0, so

$$\frac{\partial S_2}{\partial a^q_{ij}} = \frac{\partial\left(\|A^p\cdot L^o - A^q\cdot L^o\|^2\right)}{\partial a^q_{ij}} \tag{21}$$

Using Lemma 1 in (21), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a^q_{ij}} &= \frac{\partial\left(\|A^p\cdot L^o - A^q\cdot L^o\|^2\right)}{\partial a^q_{ij}} = 2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial(A^p\cdot L^o - A^q\cdot L^o)}{\partial a^q_{ij}}\\ &= -2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial A^q}{\partial a^q_{ij}}\cdot L^o\end{aligned} \tag{22}$$

Using Lemma 2 in (22), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a^q_{ij}} &= -2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot\frac{\partial A^q}{\partial a^q_{ij}}\cdot L^o = -2\cdot(A^p\cdot L^o - A^q\cdot L^o)^T\cdot M_i\cdot M_j^T\cdot L^o\\ &= -2\cdot M_i^T\cdot(A^p\cdot L^o - A^q\cdot L^o)\cdot M_j^T\cdot L^o = -2\cdot\left(M_i^T\cdot A^p\cdot L^o - M_i^T\cdot A^q\cdot L^o\right)\cdot M_j^T\cdot L^o\end{aligned} \tag{23}$$

Using Lemma 3 on (23), we have

$$\begin{aligned}\frac{\partial S_2}{\partial a^q_{ij}} &= -2\cdot\left(M_i^T\cdot A^p\cdot L^o - M_i^T\cdot A^q\cdot L^o\right)\cdot M_j^T\cdot L^o = -2\cdot\left(a^p_{i:}\cdot L^o - a^q_{i:}\cdot L^o\right)\cdot l^o_j\\ &= -2\cdot\left(\sum_k a^p_{ik}\cdot l^o_k - \sum_k a^q_{ik}\cdot l^o_k\right)\cdot l^o_j\\ &= -2\cdot\sum_k a^p_{ik}\cdot l^o_k\cdot l^o_j + 2\cdot\sum_k a^q_{ik}\cdot l^o_k\cdot l^o_j\end{aligned} \tag{24}$$

In order to estimate manifold A^p, we create a system of equations for pixel p

$$\begin{aligned}
\frac{\partial S_1}{\partial a_{11}^p} &= 2\cdot\sum_k a_{1k}^p\cdot l_k^p\cdot l_1^p - 2\cdot r_1^p\cdot l_1^p = 0\\
\frac{\partial S_1}{\partial a_{12}^p} &= 2\cdot\sum_k a_{1k}^p\cdot l_k^p\cdot l_2^p - 2\cdot r_1^p\cdot l_2^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{1j}^p} &= 2\cdot\sum_k a_{1k}^p\cdot l_k^p\cdot l_j^p - 2\cdot r_1^p\cdot l_j^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{1n}^p} &= 2\cdot\sum_k a_{1k}^p\cdot l_k^p\cdot l_n^p - 2\cdot r_1^p\cdot l_n^p = 0\\
\frac{\partial S_1}{\partial a_{21}^p} &= 2\cdot\sum_k a_{2k}^p\cdot l_k^p\cdot l_1^p - 2\cdot r_2^p\cdot l_1^p = 0\\
\frac{\partial S_1}{\partial a_{22}^p} &= 2\cdot\sum_k a_{2k}^p\cdot l_k^p\cdot l_2^p - 2\cdot r_2^p\cdot l_2^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{2j}^p} &= 2\cdot\sum_k a_{2k}^p\cdot l_k^p\cdot l_j^p - 2\cdot r_2^p\cdot l_j^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{2n}^p} &= 2\cdot\sum_k a_{2k}^p\cdot l_k^p\cdot l_n^p - 2\cdot r_2^p\cdot l_n^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{i1}^p} &= 2\cdot\sum_k a_{ik}^p\cdot l_k^p\cdot l_1^p - 2\cdot r_i^p\cdot l_1^p = 0\\
\frac{\partial S_1}{\partial a_{i2}^p} &= 2\cdot\sum_k a_{ik}^p\cdot l_k^p\cdot l_2^p - 2\cdot r_i^p\cdot l_2^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{ij}^p} &= 2\cdot\sum_k a_{ik}^p\cdot l_k^p\cdot l_j^p - 2\cdot r_i^p\cdot l_j^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{in}^p} &= 2\cdot\sum_k a_{ik}^p\cdot l_k^p\cdot l_n^p - 2\cdot r_i^p\cdot l_n^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{n1}^p} &= 2\cdot\sum_k a_{nk}^p\cdot l_k^p\cdot l_1^p - 2\cdot r_n^p\cdot l_1^p = 0\\
\frac{\partial S_1}{\partial a_{n2}^p} &= 2\cdot\sum_k a_{nk}^p\cdot l_k^p\cdot l_2^p - 2\cdot r_n^p\cdot l_2^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{nj}^p} &= 2\cdot\sum_k a_{nk}^p\cdot l_k^p\cdot l_j^p - 2\cdot r_n^p\cdot l_j^p = 0\\
&\vdots\\
\frac{\partial S_1}{\partial a_{nn}^p} &= 2\cdot\sum_k a_{nk}^p\cdot l_k^p\cdot l_n^p - 2\cdot r_n^p\cdot l_n^p = 0
\end{aligned} \tag{25}$$

We can express (25) as

$$LL^p\cdot AA^p = RL^p \tag{26}$$

where AA^p is a vector composed of all coefficients of the matrix A^p, and vector RL^p is defined as follows

$$AA^p = \begin{bmatrix} a^p_{11} \\ a^p_{12} \\ \vdots \\ a^p_{1j} \\ \vdots \\ a^p_{1n} \\ a^p_{21} \\ a^p_{22} \\ \vdots \\ a^p_{2j} \\ \vdots \\ a^p_{2n} \\ \vdots \\ a^p_{i1} \\ a^p_{i2} \\ \vdots \\ a^p_{ij} \\ \vdots \\ a^p_{in} \\ \vdots \\ a^p_{n1} \\ a^p_{n2} \\ \vdots \\ a^p_{nj} \\ \vdots \\ a^p_{nn} \end{bmatrix} \quad RL^p = \begin{bmatrix} 2\cdot r^p_1\cdot l^p_1 \\ 2\cdot r^p_1\cdot l^p_2 \\ \vdots \\ 2\cdot r^p_1\cdot l^p_j \\ \vdots \\ 2\cdot r^p_1\cdot l^p_n \\ 2\cdot r^p_2\cdot l^p_1 \\ 2\cdot r^p_2\cdot l^p_2 \\ \vdots \\ 2\cdot r^p_2\cdot l^p_j \\ \vdots \\ 2\cdot r^p_2\cdot l^p_n \\ \vdots \\ 2\cdot r^p_i\cdot l^p_1 \\ 2\cdot r^p_i\cdot l^p_2 \\ \vdots \\ 2\cdot r^p_i\cdot l^p_j \\ \vdots \\ 2\cdot r^p_i\cdot l^p_n \\ \vdots \\ 2\cdot r^p_n\cdot l^p_1 \\ 2\cdot r^p_n\cdot l^p_2 \\ \vdots \\ 2\cdot r^p_n\cdot l^p_j \\ \vdots \\ 2\cdot r^p_n\cdot l^p_n \end{bmatrix} \tag{27}$$

And the matrix LL^p is a block-based matrix composed of ll^p defined as follows

$$ll^p = \begin{bmatrix} 2\cdot l^p_1\cdot l^p_1 & 2\cdot l^p_2\cdot l^p_1 & \cdots & 2\cdot l^p_k\cdot l^p_1 & \cdots & 2\cdot l^p_n\cdot l^p_1 \\ 2\cdot l^p_1\cdot l^p_2 & 2\cdot l^p_2\cdot l^p_2 & \cdots & 2\cdot l^p_k\cdot l^p_2 & \cdots & 2\cdot l^p_n\cdot l^p_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2\cdot l^p_1\cdot l^p_j & 2\cdot l^p_2\cdot l^p_j & \cdots & 2\cdot l^p_k\cdot l^p_j & \cdots & 2\cdot l^p_n\cdot l^p_j \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2\cdot l^p_1\cdot l^p_n & 2\cdot l^p_2\cdot l^p_n & \cdots & 2\cdot l^p_k\cdot l^p_n & \cdots & 2\cdot l^p_n\cdot l^p_n \end{bmatrix} \tag{28}$$

$$LL^p = \begin{bmatrix} ll^p & 0 & \cdots & 0 & \cdots & 0 \\ 0 & ll^p & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & ll^p & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & ll^p \end{bmatrix} \tag{29}$$

For a manifold composed of P pixels, we can define the system of Equation (25) using the Formula (26) as

$$\mathcal{LL} \cdot \mathring{A}\mathring{A} = \mathcal{RL} \tag{30}$$

where ÅÅ and $\mathcal{RL}$ are a concatenation of AA^p and RL^p for all pixels in the image

$$\mathring{A}\mathring{A} = \begin{bmatrix} AA^1 \\ AA^2 \\ \vdots \\ AA^p \\ \vdots \\ AA^P \end{bmatrix} \quad \mathcal{RL} = \begin{bmatrix} RL^1 \\ RL^2 \\ \vdots \\ RL^p \\ \vdots \\ RL^P \end{bmatrix} \tag{31}$$

And $\mathcal{LL}$ can be defined as a block matrix composed of LL^p

$$\mathcal{LL} = \begin{bmatrix} LL^1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & LL^2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & LL^p & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & LL^P \end{bmatrix} \tag{32}$$

Similarly, in order to estimate manifold A^p based on S_2, we create a system of Equations (19) and (24) for pixel p and q

$$\begin{aligned} \frac{\partial S_2}{\partial a^p_{ij}} &= 2 \cdot \sum_k a^p_{ik} \cdot l^o_k \cdot l^o_j - 2 \cdot \sum_k a^q_{ik} \cdot l^o_k \cdot l^o_j = 0 \\ \frac{\partial S_2}{\partial a^q_{ij}} &= 2 \cdot \sum_k a^q_{ik} \cdot l^o_k \cdot l^o_j - 2 \cdot \sum_k a^p_{ik} \cdot l^o_k \cdot l^o_j = 0 \end{aligned} \tag{33}$$

Using a definition of LL^p given in (29), we can rewrite Equation (33) as

$$\begin{bmatrix} LL^o & -LL^o \\ -LL^o & LL^o \end{bmatrix} \cdot \begin{bmatrix} AA^p \\ AA^q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{34}$$

The exact formulation of $\mathcal{LL}$ matrix for all of the P pixels depends strongly on the relationship between the position of pixels p and q.

4. Formulation in 2D Pixel Grid

Suppose we have an image of size W by H, and we can indicate the pixel p position by its u, v coordinates in the image. Moreover, suppose that we introduce 2 smoothness constraints: one vertical and one horizontal. It means that pixel p_v and q_v have two neighboring pixels in which the v coordinate differs only by 1.

$$p_v = u, v \quad q_v = u, v+1 \tag{35}$$

Similarly, pixel p_h and q_h have two neighboring pixels whose u coordinate differs only by 1.

$$p_h = u, v \quad q_h = u+1, v \tag{36}$$

Moreover, we will assume that condition (34) is reflective and must hold for both pixels p and q, so:

$$o = p \quad o = q \tag{37}$$

will lead to

$$\begin{bmatrix} LL^p & -LL^p \\ -LL^p & LL^p \end{bmatrix} \cdot \begin{bmatrix} AA^p \\ AA^q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} LL^q & -LL^q \\ -LL^q & LL^q \end{bmatrix} \cdot \begin{bmatrix} AA^p \\ AA^q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{39}$$

Combining both (38) and (39) together results in

$$\begin{bmatrix} LL^p + LL^q & -(LL^p + LL^q) \\ -(LL^p + LL^q) & LL^p + LL^q \end{bmatrix} \cdot \begin{bmatrix} AA^p \\ AA^q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{40}$$

After making such assumptions, we can provide an explicit form of ÅÅ, $\mathcal{RL}$ vectors, and $\mathcal{LL}$ matrix for any given pixel $p = u, v$ based on (31), (32), and (40). As the conditions S_1 and S_2 need to be satisfied simultaneously, we introduce two coefficients α and β to control the strength of the smoothness prior to S_2 in both directions on the 2D grid. The values of coefficients α and β were experimentally adjusted for universally good results among the tested sequences. The value of 0.1 was found to work well in our experiments, independent of the sequence contents and scene complexity. It needs to be stressed, however, that different resolutions of the sequences may require a slight modification of the values, but we found during the experiments that the method is not sensitive to the actual value of the coefficients α and β.

$$\text{ÅÅ}^{p=u,v} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ AA^{p=u,v} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \mathcal{RL}^{p=u,v} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ RL^{p=u,v} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{41}$$

$$
\mathcal{LL}^{p=u,v} = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & LL^{p=u,v} + \alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) + \beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & -\alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & 0 & \cdots & 0 & -\beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & -\alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & \alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & -\beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & 0 & \cdots & 0 & \beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{bmatrix} \tag{42}
$$

The final form of Equation (43) for all of the pixels is just a sum of (41) and (42)

$$
\mathcal{LL}\cdot \mathring{A}\mathring{A} = \mathcal{RL} \tag{43}
$$

$$
\begin{aligned}
\mathcal{LL} &= \sum_{p}^{P} \mathcal{LL}^{p} \\
\mathcal{RL} &= \sum_{p}^{P} \mathcal{RL}^{p} \\
\mathring{A}\mathring{A} &= \sum_{p}^{P} \mathring{A}\mathring{A}^{p}
\end{aligned} \tag{44}
$$

5. Solution on a 2D Grid

The solution of obtained Equation (43) is simply given by

$$
\mathring{A}\mathring{A} = \mathcal{LL}^{-1}\cdot\mathcal{RL} \tag{45}
$$

However, solving for $\mathring{A}\mathring{A}$ on an average image is not an easy task. For a typical RGB image of resolution 1920×1080 vectors $\mathcal{RL}$ have the size of $4\cdot4\cdot1920\cdot1080 = 33,177,600$ and the square matrix $\mathcal{LL}$ is of that same size. The main problem is to inverse the $\mathcal{LL}$ matrix, mainly because it is so enormous.

Fortunately, matrix $\mathcal{LL}$ is also vastly sparse with most of its coefficients located along the first, second, and W-diagonal (where W stands for the width of the image under consideration). There are many algorithms for efficient inversing (or pseudo-inversing) such huge sparse matrices.

In the experiments further on we have used Transpose-Free Quasi-Minimal Residual Algorithm [25–27].

After solving Equation (43) for $\mathring{A}\mathring{A}$, we obtained an estimation of the manifold of per-pixel transformation A^{p} allowing per-pixel color transformation between both of the images.

This allows the transformation of one image into a color space of the other, retaining its local color characteristics. After transformation, depending on the use case, one can directly output the transformed image as the final result, as can be performed in the "flash–no flash" application, merge them via some kind of blending or averaging, or inpaint one of the images by the content of the other or stitch them together to create a larger panorama.

6. Notes on Partially Overlapping Images

In the case of partially overlapping images like in the case of image stitching or view synthesis, a form of the $\mathcal{LL}^{p=u,v}$ matrix and the $\mathcal{RL}^{p=u,v}$ vector needs to be adjusted. Without loss of generality, let us assume that the left image is more complete, i.e., it has pixels in places where the right image does not. So, there are two possibilities: either left and right images do not have a given pixel or just the right pixel is not available. In both cases, for these pixels' positions, condition S_1 cannot be used due to a lack of explicit pixel correspondence. In the second case, depending on the surroundings, we have pixel color values in the left image that can be used in condition S_2 for smooth manifold estimation.

For the first case, $\mathcal{LL}^{p=u,v}$ is as shown in (46)

$$\mathcal{LL}^{p=u,v} = [0] \tag{46}$$

In the second case, the $\mathcal{LL}^{p=u,v}$ is given as (47)

$$\mathcal{LL}^{p=u,v} = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & \alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) + \beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & -\alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & 0 & \cdots & 0 & -\beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & -\alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & \alpha\cdot\left(LL^{p=u,v} + LL^{q_h=u+1,v}\right) & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & -\beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & 0 & \cdots & 0 & \beta\cdot\left(LL^{p=u,v} + LL^{q_v=u,v+1}\right) & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{bmatrix} \tag{47}$$

7. Experiments

The foundation of our experiments was the reference virtual view synthesis software VSRS [28]. It should be noted that VSRS does not include any color correction algorithm. Our algorithm was implemented within this software package.

We modified the post-processing only after the forward depth map projection step and the backward texture warping step had been carried out, and before the inpainting step.

For our testing, we employed several well-known and recognizable multiview test sequences equipped with high-quality depth maps. This set includes linear and arc multiview content. Used multiview test sequences exhibit a whole range of effects like ambient occlusions, nonuniform lighting, shadows, and non-Lambertian reflections. Table 1 lists the sequences used in the experiment.

We rendered virtual views using the original VSRS and VSRS implementing our method. The virtual views were generated from data from the two views specified as the right and left views in Table 1. The set of data consists of the images, depth maps, and camera parameters. The virtual view matched the position of the third camera used during sequence acquisition.

The quality of the resulting virtual view has been measured as a luminance PSNR value of the virtual view generated with respect to the view recorded by the real camera at the same spatial position. The numerical results are presented in Table 2. A demonstration of the improvement provided by our method over the original method is shown in Figure 1, where details of the virtual views are compared.

Table 1. Test sequences used in the experiment.

Sequence	Resolution	Left Reference View	Right Reference View	Virtual View
Poznań Street [29]	1920 × 1080	3	5	4
Poznan Fencing 2 [30]	1920 × 1080	4	6	5
Poznan Blocks [30]	1920 × 1080	4	6	5
Poznan Carpark [29]	1920 × 1080	3	5	4
Ballet [31]	1024 × 768	3	5	4
Breakdancer [31]	1024 × 768	3	5	4
Soccer [32]	1392 × 1136	3	5	4

Table 2. Quality (PSNR) of the virtual views.

	Quality (PSNR) Without Color Correction [dB]	Quality (PSNR) with Proposed Color Correction [dB]	Quality (PSNR) Improvement [dB]
Poznan Street	35.48	36.23	0.75
Poznan Fencing	29.62	31.23	1.61
Poznan Blocks	29.60	32.43	2.83
Poznan Carpark	36.87	37.02	0.15
Ballet	31.05	31.94	0.89
Breakdancer	31.47	31.83	0.46
Soccer	35.00	36.03	1.03

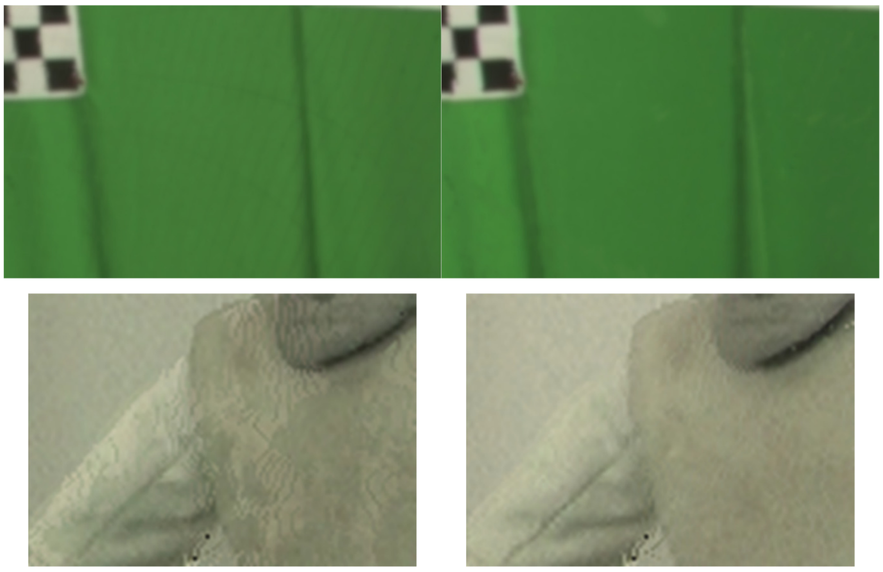

Figure 1. Details of the virtual views: Top row: Poznan Blocks, Bottom row: Poznan Fencing 2. Original VSRS on the **left**, modified VSRS with proposed color correction on the **right**. As can be seen on the right-hand side images, most of the artifacts due to color mismatch are eliminated.

It can clearly be seen that the presented method significantly improves the quality of a virtual view by providing consistent colors of the pixels from different views. The artifacts stemming from slight differences between the color characteristics of the cameras and/or lighting conditions are reduced significantly.

The PSNR value measured against a real view is a commonly used measure of quality for virtual view synthesis research. For color correction algorithms, however, a more significant figure of merit would be the comparison between the virtual view synthesized only from the left and only from the right view. Table 3 shows the values of the error (measured as a mean value of SSD for pixel values) for the virtual view generated from the left view calculated against the virtual view generated from the right view for three cases: no color correction applied, the global color correction applied according to the method described in [33], and the proposed method. A visual comparison of the differences between the virtual view generated from the left view and the virtual view generated from the right view is presented in Figures 2 and 3. The proposed method outperforms significantly the global color correction method.

Table 3. Mean values of SSD for the virtual view generated from the left view compared to the virtual view generated from the right view.

Sequence	Without Color Correction	Global Color Transformation From [33]	Proposed Per-Pixel Color Transformation
Poznań Street	1.6496	1.4576	0.0304
Poznan Fencing 2	4.4137	4.0283	0.0504
Poznan Blocks	2.0739	1.9387	0.0977
Poznan Carpark	1.7684	1.5984	0.0779
Ballet	1.3975	1.1293	0.0600
Breakdancer	1.4839	1.3938	0.2880
Soccer	1.1830	1.0283	0.2735

In order to test our method in a flash/no-flash scenario, we use the images from [4], as supplied in [34]. The proposed method of estimating per-pixel color transfer manifold can be used to correct images captured with flash to its low-light ambient counterpart, to match the no-flash colors (Figure 4).

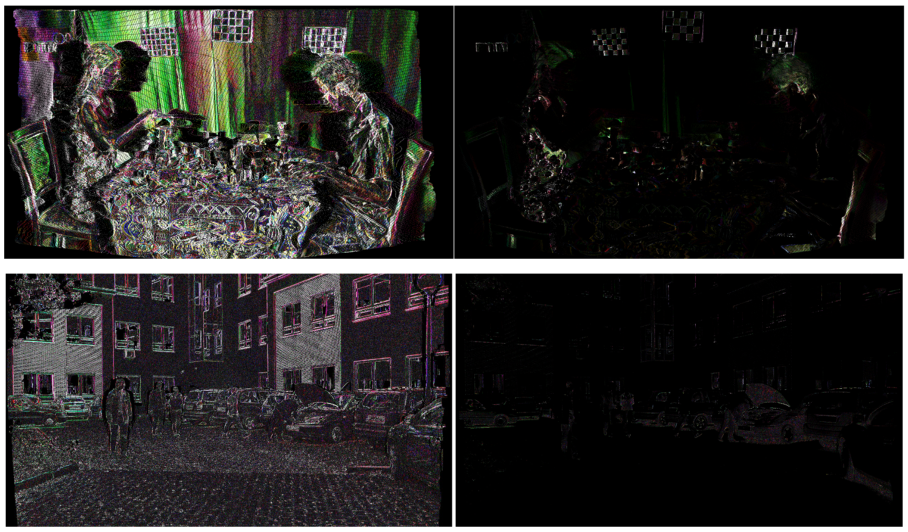

Figure 2. *Cont.*

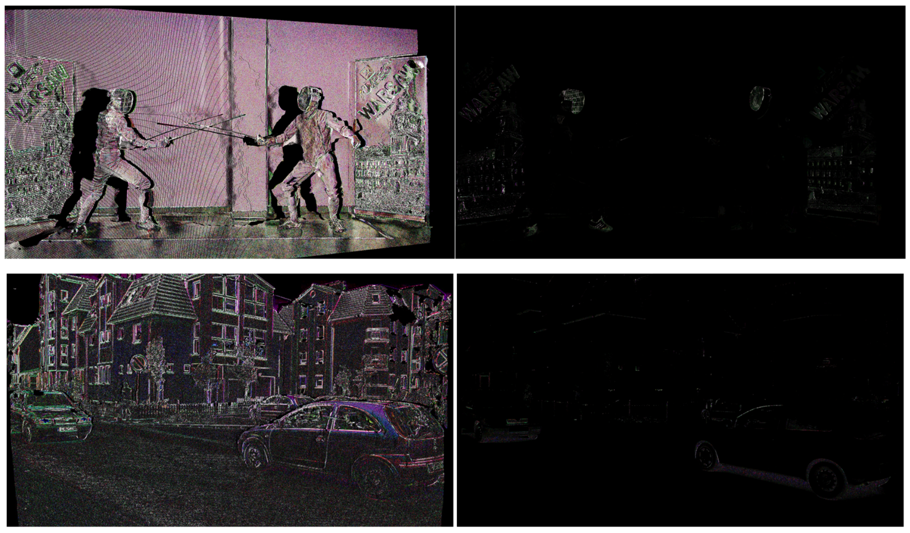

Figure 2. Comparison of the differences between the virtual view generated from the left view and the virtual view generated from the right view: **Left column**—original VSRS, **right column**—modified VSRS. From the top: Poznan Blocks, Poznan Street, Poznan Fencing 2, Poznan Carpark.

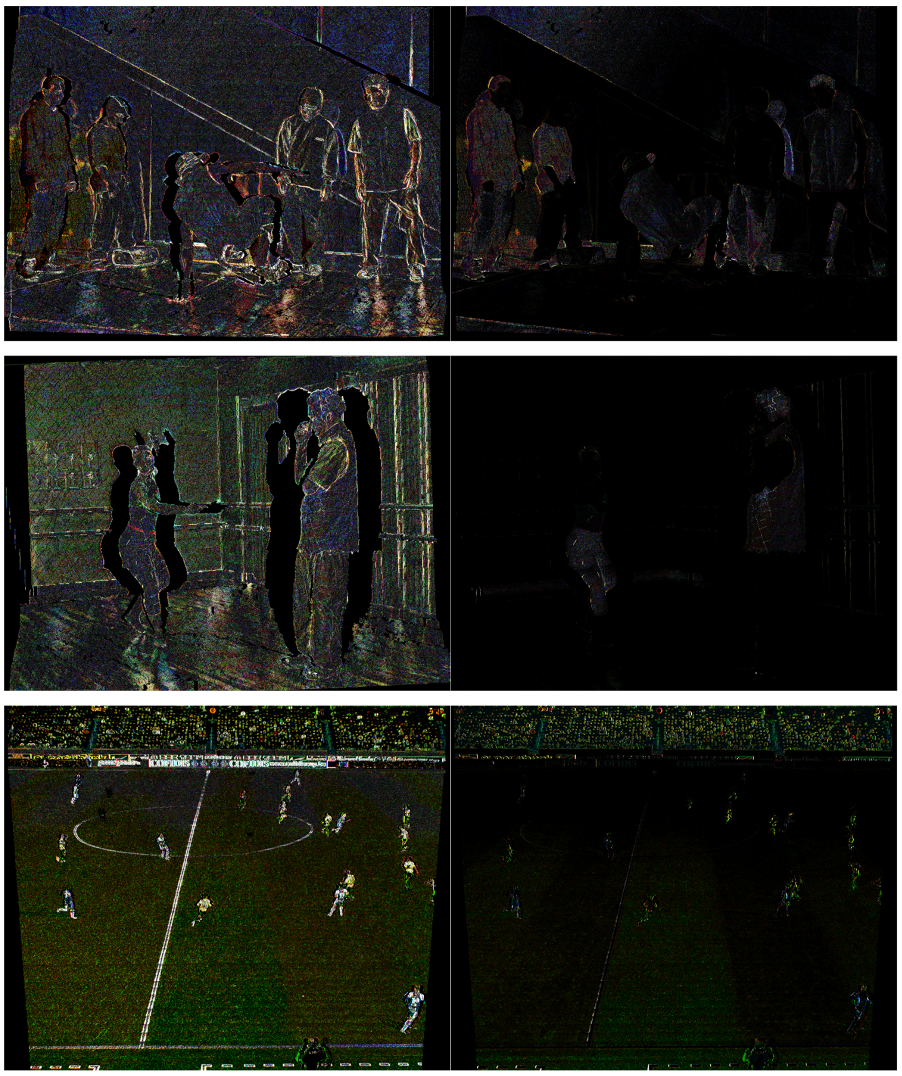

Figure 3. Comparison of the differences between the virtual view generated from the left view and the virtual view generated from the right view: **Left column**—original VSRS, **right column**—modified VSRS. From the top: Breakdancer, Ballet, Soccer.

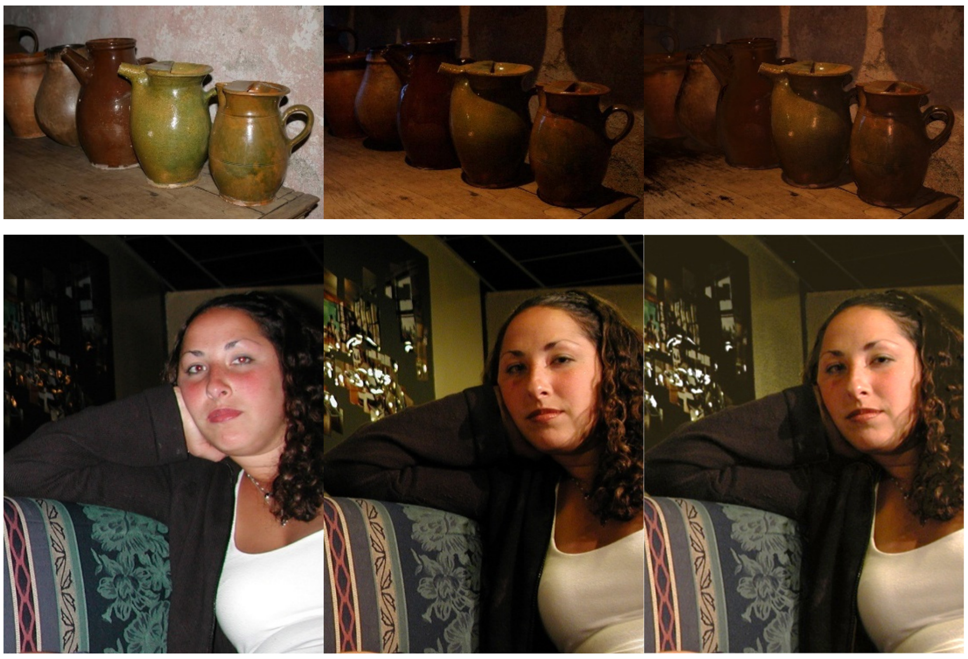
Figure 4. Results of the proposed method for images with different lighting conditions. From the left: image with flash, no-flash image, result of our color-correction of the image with flash.

As can be seen, the proposed method is able to perform the color correction of the flash image so that it matches the colors of the image taken without flash. It can be seen that the artifacts common to images taken with flash, as non-Lambertian reflections are correctly processed.

8. Summary

In the paper, we have presented a new formulation of the per-pixel color calibration between images. The presented method can be applied to images with an arbitrary number of channels due to the flexibility and generalization provided by our derivation. The proposed algorithm is based on per-pixel manifold color transformation estimation. The proposed algorithm estimated a smooth manifold of color transformations across the whole image, even in the case when not all pixels have correspondence. The method allows for color profile mapping in difficult areas, like occluded regions of images, in shadows, and in the presence of reflections and a non-uniform lighting. The proposed method provides a gain in objective quality measure PSNR of 1.10 dB on average over seven different test sequences. The proposed method provided improvement for all tested sequences; the quality increase was in the range from 2.83 dB to 0.15 dB. The explanation of the differences may be attributed qualitatively to the different lighting conditions. Smaller improvements were observed for outdoor sequences recorded during a cloudy day, with no bright sun, and for sequences with uniform ambient lighting. For sequences with artificial light, the improvement was significantly larger. The subjective quality increase is also significant, as presented by the images in Figures 1 and 4.

Much more improvement due to the use of the proposed method is observed when SSD between the virtual view synthesized from the left view is compared to the view synthesized from the right view. Here, when compared to the no color correction scenario, the reduction was more than 20-fold, with a maximum of 87-fold SSD reduction. When compared to a scenario where a competing method [33] is used, the SSD reduction is from 19.8-fold to 79.9-fold.

The presented method is envisaged to be applied as a preprocessing step in a post-production phase, since the computational complexity is significant, the most time-consuming part being the large sparse matrix inversion. While real-time image processing was beyond the scope of this work, it is worth noting that faster matrix inversion methods exist, which would reduce the computational complexity of our method.

The method significantly improves the quality of the virtual view synthesized from the multi-camera setups by applying color correction to all original views. It can be used in immersive video applications, where it can greatly reduce the color mismatch between camera views. It is also useful for single images in the scenario, flash–no flash images, as evidenced in the experiments.

Author Contributions: Conceptualization, K.W. and T.G.; methodology, S.G.; software, K.W.; validation, S.G., T.G. and K.K.; formal analysis, S.G.; investigation, T.G. and K.K.; writing—original draft preparation, K.W.; writing—review and editing, T.G. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: Project funded by The National Centre for Research and Development in the LIDER Programme (LIDER/34/0177/L-8/16/NCBR/2017). The work was supported by the Ministry of Science and Higher Education, Poland.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Krzysztof Wegner was employed by the company Mucha sp. z o.o. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Mildenhall, B.; Srinivasan, P.P.; Tancik, M.; Barron, J.T.; Ramamoorthi, R.; Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **2021**, *65*, 99–106. [CrossRef]
2. Wang, P.; Liu, Y.; Chen, Z.; Liu, L.; Liu, Z.; Komura, T.; Theobalt, C.; Wang, W. F2-nerf: Fast neural radiance field training with free camera trajectories. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 4150–4159.
3. Nawaz, A.; Kanwal, I.; Idrees, S.; Imtiaz, R.; Jalil, A.; Ali, A.; Ahmed, J. Fusion of color and infrared images using gradient transfer and total variation minimization. In Proceedings of the 2nd International Conference on Computer and Communication Systems (ICCCS), Krakow, Poland, 11–14 July 2017; pp. 82–85. [CrossRef]
4. Petschnigg, G.; Szeliski, R.; Agrawala, M.; Cohen, M.; Hoppe, H.; Toyama, K. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.* **2004**, *23*, 664–672. [CrossRef]
5. Furukawa, Y.; Hernández, C. Multi-View Stereo: A Tutorial. *Found. Trends® Comput. Graph. Vis.* **2015**, *9*, 1–148. [CrossRef]
6. Li, Y.; Li, Y.; Yao, J.; Gong, Y.; Li, L. Global Color Consistency Correction for Large-Scale Images in 3-D Reconstruction. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 3074–3308. [CrossRef]
7. Chung, K.-L.; Lee, C.-Y. Fusion-Based Multiview Color Correction via Minimum Weight-First-Then-Larger Outdegree Target Image Selection. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 4707716. [CrossRef]
8. Lu, S.-P.; Ceulemans, B.; Munteanu, A.; Schelkens, P. Spatio-Temporally Consistent Color and Structure Optimization for Multiview Video Color Correction. *IEEE Trans. Multimed.* **2015**, *17*, 577–590. [CrossRef]
9. Chen, Y.; Cai, C.; Liu, J. YUV Correction for Multi-View Video Compression. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20–24 August 2006; pp. 734–737. [CrossRef]
10. Fezza, S.A.; Larabi, M.-C.; Faraoun, K.M. Coding and rendering enhancement of multi-view video by color correction. In Proceedings of the European Workshop on Visual Information Processing (EUVIP), Paris, France, 10–12 June 2013; pp. 166–171.
11. Fezza, S.A.; Larabi, M.-C. Color calibration of multi-view video plus depth for advanced 3D video. *Signal Image Video Process.* **2015**, *9*, 177–191. [CrossRef]

12. Ganelin, I.; Nasiopoulos, P. Virtual View Color Estimation for Free Viewpoint TV Applications Using Gaussian Mixture Model. In Proceedings of the 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 3898–3902. [CrossRef]
13. Ding, C.; Ma, Z. Multi-Camera Color Correction via Hybrid Histogram Matching. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 3327–3337. [CrossRef]
14. Joshi, N. *Color Calibration for Arrays of Inexpensive Image Sensors, Technoly Report*; CSTR 2004-02 3/31/04 4/4/04; Stanford University: Stanford, CA, USA, 2004.
15. Suominen, J.; Egiazarian, K. Camera Color Correction Using Splines. *Electron. Imaging* **2024**, *36*, 165-1–165-6. [CrossRef]
16. Qiao, Y.; Jiao, L.; Yang, S.; Hou, B.; Feng, J. Color Correction and Depth-Based Hierarchical Hole Filling in Free Viewpoint Generation. *IEEE Trans. Broadcast.* **2019**, *65*, 294–307. [CrossRef]
17. Dziembowski, A.; Mieloch, D.; Różek, S.; Domański, M. Color Correction for Immersive Video Applications. *IEEE Access* **2021**, *9*, 75626–75640. [CrossRef]
18. Ilie, A.; Welch, G. Ensuring color consistency across multiple cameras. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China, 17–21 October 2005; Volume 2, pp. 1268–1275. [CrossRef]
19. Reinhard, E.; Adhikhmin, M.; Gooch, B.; Shirley, P. Color transfer between images. *IEEE Comput. Graph. Appl.* **2001**, *21*, 34–41. [CrossRef]
20. Flynn, J.; Neulander, I.; Philbin, J.; Snavely, N. DeepStereo: Learning to Predict New Views From the World's Imagery. *arXiv* **2016**, arXiv:1506.06825. [CrossRef]
21. Li, Y.; Li, L.; Yao, J.; Xia, M.; Wang, H. Contrast-Aware Color Consistency Correction for Multiple Images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 4941–4955. [CrossRef]
22. Ye, S.; Lu, S.-P.; Munteanu, A. Color correction for large-baseline multiview video. *Signal Process. Image Commun.* **2017**, *53*, 40–50. [CrossRef]
23. Ceulemans, B.; Lu, S.-P.; Schelkens, P.; Munteanu, A. Globally optimized multiview video color correction using dense spatio-temporal matching. In Proceedings of the 2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), Lisbon, Portugal, 8–10 July 2015; pp. 1–4. [CrossRef]
24. Liao, D.; Qian, Y.; Li, Z.-N. Semisupervised manifold learning for color transfer between multiview images. In Proceedings of the 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 259–264. [CrossRef]
25. Freund, R.W. A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems. *SIAM J. Sci. Comput.* **1993**, *14*, 470–482. [CrossRef]
26. Saad, Y. *Iterative Methods for Sparse Linear Systems: Second Edition*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2003. [CrossRef]
27. Kelley, C.T. *Iterative Methods for Linear and Nonlinear Equations*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1995. [CrossRef]
28. Senoh, T.; Kenji, J.; Nobuji, T.; Hiroshi, Y.; Wegner, K. View Synthesis Reference Software (VSRS) 4.2 with improved inpainting and hole filing. In Proceedings of the ISO/IEC JTC1/SC29/WG11 MPEG2017, Hobart, Australia, 1–7 April 2017.
29. Domański, M.; Grajek, T.; Klimaszewski, K.; Kurc, M.; Stankiewicz, O.; Stankowski, J.; Wegner, K. Poznań Multiview Video Test Sequences and Camera Parameters M17050. In Proceedings of the ISO/IEC JTC1/SC29/WG11, MPEG2009, Xian, China, 26–30 October 2009.
30. Domański, M.; Dziembowski, A.; Kuehn, A.; Kurc, M.; Łuczak, A.; Mieloch, D.; Siast, J.; Stankiewicz, O.; Wegner, K. Poznan Blocks—A Multiview Video Test Sequence and Camera Parameters for Free Viewpoint Television M32243. In Proceedings of the ISO/IEC JTC1/SC29/WG11 MPEG2014, San Jose, CA, USA, 13–17 January 2014.
31. Zitnick, C.; Kang, S.B.; Uyttendaele, M.; Winder, S.; Szeliski, R. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.* **2004**, *23*, 600–608. [CrossRef]
32. Goorts, P. The Genk Dataset, Real-Time Adaptive Plane Sweeping for Free Viewpoint Navigation in Soccer Scenes. Ph.D. Thesis, Hasselt University, Hasselt, Belgium, 2014; pp. 175–180.
33. Senoh, T.; Tetsutani, N.; Yasuda, H. [MPEG-I Visual] Proposal of Trimming and Color Matching of Multi-View Sequences M47170. In Proceedings of the ISO/IEC JTC1/SC29/WG11 MPEG2019, Geneva, Switzerland, 25–29 March 2019.
34. Afifi, M.; Digital Photography with Flash and No-Flash Image Pairs. MATLAB Central File Exchange. Available online: https://www.mathworks.com/matlabcentral/fileexchange/62625-digital-photography-with-flash-and-no-flash-image-pairs (accessed on 7 March 2025).