Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Arithmetically fast position transformation for view synthesis and depth estimation

KRZYSZTOF WEGNER[1] (Senior Member, IEEE), TOMASZ GRAJEK[2] (Senior Member, IEEE), KRZYSZTOF KLIMASZEWSKI [2]
[1]Mucha sp. z o.o., ul. Szelagowska 17, 61-626 Poznan
[2]Poznan University of Technology, pl. M. Sklodowskiej-Curie 5, 60-965 Poznan, Poland

Corresponding author: Krzysztof Klimaszewski (e-mail: krzysztof.klimaszewski@put.poznan.pl).

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 Education and Science, Poland.

ABSTRACT In this paper we present a method of fast computation of matrix transformation in the process of position transformation of objects of the scene between different, virtual or real, camera positions. The process finds extensive use in virtual view generation in Free Viewpoint Video (FVV) and virtual reality applications as well as in depth estimation algorithms. The proposed method relies on the reformulation of the matrix equation used in the process. As a result, the number of necessary arithmetic operations is reduced and some of the calculations can be reused for consecutive pixel position transformations. The presented algorithm produces identical output as an unoptimized algorithm with approximately 22% reduction of the processing time averaged over the examined representative test sequences.

INDEX TERMS Depth estimation, DIBR, Multiview and depth, Position transformation, View synthesis

I. INTRODUCTION

IN many contemporary applications that focus on the processing of 3D scenes, it is necessary to convey information about the 3D structure of the scene. Such information may be later used for displaying the correct view for a given viewpoint in immersive video [1] applications, for a user wearing a head-mounted display or even for a regular display [2]. Usually, the information about the structure of the scene is stored in a depth map - a grayscale image that accompanies a regular RGB image of a scene recorded by a regular camera. The resolution of a depth map is usually the same as the resolution of the RGB image. Each pixel of the depth map for an RGB image from a given camera has a value dependent on the distance between the camera and the object of the scene depicted in the co-located RGB image pixel. Usually, the dependence is non-linear. A set of images from different cameras, recording a given scene from different points, together with the corresponding depth maps for each image, can be used to generate another, virtual view of the scene, from a point that was not occupied by any camera during the recording, in a process called depth image based rendering (DIBR) [3] [4] [5].

The are two major applications of DIBR: view synthesis as the most obvious one and the less obvious one: depth estimation. In both of them, the projection of the point position from one view to the other plays a crucial role. The process of reprojection is the most important one used in DIBR and in order to obtain a single frame of a virtual view, it has to be performed for every pixel of all of the considered real views together with the corresponding depth maps. Therefore, the speed of the reprojection process significantly influences the overall performance of the respective algorithms. The usual method of performing a reprojection in a multiview system is based on the relationships stemming from the projective geometry. The process involves matrix calculations that, in a general case, can not be avoided. Although for some systems, a much simpler version of the reprojection algorithm can be used, but that is only true if the images from cameras are rectified. The accurate rectification process is possible only for up to three cameras. An approximate method, when applied to systems with more cameras, introduces additional errors. For most contemporary systems the rectification process for all cameras at once is neither possible nor desirable.

The first significant application of reprojection algorithms is view synthesis. In the process of view synthesis, many problems are usually encountered that can be overcome by additional reprojection operations. Among the problems is the one in which the image may be produced using inconsistent depth [6], so further processing is necessary, including a repeated reprojection operation. Another problem that some-times poses an important source of errors is a finite resolution of the depth map and image, along with the quantized nature of depth values. The ways to overcome those problems, again, by using the reprojection algorithm are described in the literature, like in [7].

Since in practice, this reprojection process does not generate a full image, there usually are some areas of the image that were not supplied with any information. Regardless of the mitigation methods cited above, there usually are some such areas of the image. Those areas need to be artificially filled with information that is likely to be perceived as correct information for the given part of the image. Therefore, after point transformation, other processing like image inpainting [8] completes the view synthesis process.

The view synthesis process has been studied in the literature for many years now. Starting from the idea of free viewpoint television and 3D television [9] [10], different algorithms and use cases have been studied [11] [12]. The study of view synthesis algorithms and systems is ongoing, newly released papers deal with different shortcomings of existing algorithms, like the existence of gaps in the virtual view and the need to fill them [13] as well as prevent the occurrence of artifacts of different kind [14]. Some work is also done in order to speed up the process of view synthesis. In [15] authors implement a method for virtual view generation, based on patches and the use of a GPU. The use of the patches is justified by the fact that it is faster than the pixel-wise view synthesis. Other papers consider different improvements in the view synthesis algorithm by preprocessing the input data in order to improve the final output of the view synthesis [16]. The authors of the paper [17] propose a method of virtual view synthesis based on the interpolation of data from different real views of the scene. The view synthesis process is also useful for view matching, in a similar way as described in [18]. At the same time the estimation is performed for the location of cameras and the structure of the scene.

The CPU-based algorithm for view synthesis using a reprojection based on matrix calculations is a View Synthesis Reference Software (VSRS). View Synthesis Reference Software is an implementation of a view synthesis algorithm developed by MPEG of ISO in the course of immersive media standardization effort with a significant contribution of the author of this paper [19].

A second area of application of the reprojection algorithms is depth map estimation. Depth estimation is a process of generating a depth map for a given image. A depth map includes information about the distances of individual points of the image from the camera that was used to record that image. The said estimation is a process of checking the similarity of parts of one image to the parts in the second image. For each part of the first image, the reprojection algorithm is used for all considered depth values. The part of the image gets reprojected onto the second image. For each reprojection, the so-called reprojection error (the difference between the part of the second image and the reprojected to that image part of the first image) is calculated. Finally, the error values are compared and the lowest is found. The depth value for which this reprojection was performed is then considered to be the proper one for the given part of the first image.

In the simplest arrangement of a stereo pair of cameras, both images are rectified [20] and the correspondence search is performed along a horizontal direction only. It is the simplest scenario, for which the reprojection procedure is very simple and does not require any involved calculations. In more general cases, however, the correspondence search must be performed along so called epipolar lines [21]. In such cases, the position of a given point of one of the images is projected many times to the other image in order to calculate and check reprojection error. The general case of reprojection involves a significant amount of computations and transformations and therefore can consume significant time.

There are two distinct groups of algorithms for depth estimation. The first group consists of traditional algorithms that seek to minimize a certain, usually global, cost function in the process of the depth estimation [22] [23] [24] [25]. In the depth estimation algorithm, some additional data may be used, like some low-resolution active depth sensors [26] [27]. In this group of algorithms, the reprojection process is extensively used at many stages of the processing. One of the most important properties of those traditional algorithms is the relatively low quality of the generated depth maps, especially when no additional information is used, and significant processing time that is required to obtain depth maps of reasonable quality.

Recently a lot of researchers have focused on deep learning algorithms for obtaining depth maps. This kind of algorithms constitutes the second group of depth estimation algorithms. The reprojection process is commonly used during the learning process of such neural networks. In the paper [28] the authors use the process they call warping, a reprojection algorithm that is used to obtain the error metric for the depth map generated by the neural network. The results demonstrate the superior performance of deep learning depth map generation algorithms over the traditional ones. The advantage of such algorithms is the ability to perform depth estimation much faster and of much higher quality. The recently developed algorithms, like the ones presented in [29] [30] [31] [32] show the big potential of the deep neural networks in applications of depth map generation. The use of reprojection in this group of depth estimation algorithms can be found in the process of network training and in providing initial data for key points or as initial data in the absence of the depth data from another source.

The use of deep learning algorithms to generate depth maps does have some significant issues that can be overcome by the use of a more traditional reprojection-based depth estimation. The mentioned deep learning-based algorithms are trained on very specific sets of training data, that are usually preprocessed to a significant extent. Rectification is applied to stereo data used for training - a step that is not necessary for the reprojection-based algorithms. Also, reprojection-based algorithms can provide accurate depth values for scene objects

- a feature that can not be guaranteed for some deep learning-based algorithms. In the recent paper [33], however, the authors demonstrate an entire system for free-viewpoint video, starting from the acquisition setup, depth map generation, and virtual view synthesis. The authors demonstrate a real-time performance of the whole system with good quality of virtual views, which is an outstanding achievement. The process of image reprojection is used in the virtual view synthesis, which would be able to utilize the optimization presented in our paper.

The deep learning algorithms are also used to post-process depth maps, as for example in [34], where they are used to produce a reduced size depth maps for thumbnail generation. Depth map optimization can also be performed using an analytic approach, as the same group of researchers demonstrate in a more recent paper [35].

Regardless the application, the use of the pixel-wise reprojection algorithm, in its general form, with matrix calculations, is suited for cases of arbitrary camera positioning and for cases where cameras used in the system have different intrinsic parameters. The usual, direct form of applying the algorithm can be, however, as will be shown in this paper, significantly optimized for speed while providing exactly the same numerical results. In this paper we are presenting a way of projection operation organization that is mathematically equivalent to commonly used implementation [19] yet resulting in vast performance improvement. The work is an extension of the findings of the paper [36], where authors pursue the aim of increasing the performance of the virtual view synthesis by applying vector calculations provided by modern general purpose processors. It is also shown that some improvement of the performance can be obtained without the use of any specialized hardware extensions. This improvement can therefore be applied regardless the hardware used to perform the reprojection algorithm. In this paper, we show a detailed derivation of the necessary modifications to the algorithm, that has been lacking in the literature. We also study the details of the performance improvement and give experimental results for some representative test sequences.

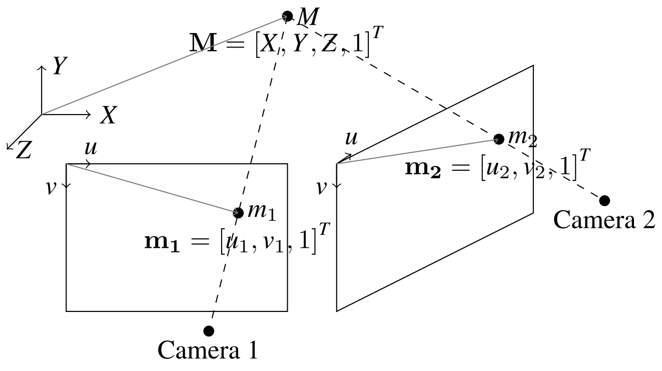
The remaining parts of the paper are organized as follows: first, we show how the position transformation operation is derived, then we discuss commonly used implementation of this operation, and then we present a new formulation of the transformation operation. Finally, we present our algorithm for fast position transformation and show how it compares to the algorithm used in the reference software.

II. MULTIVIEW GEOMETRY - REVIEW

Let us consider a point M in 3D space observed by two cameras simultaneously (Fig 1). Position of point M in world 3D space is given by homogeneous coordinates [21]

$$\mathbf{M} = [X, Y, Z, 1]^T, \tag{1}$$

where X, Y, Z are coordinates in the world 3D space.



M
$\mathbf{M} = [X, Y, Z, 1]^T$
Y
X
Z
u
v
m_1
$\mathbf{m_1} = [u_1, v_1, 1]^T$
Camera 1
u
v
m_2
$\mathbf{m_2} = [u_2, v_2, 1]^T$
Camera 2

FIGURE 1. Projections m_1 and m_2 of point M onto image planes of Camera 1 and Camera 2

Positions m_1 and m_2 of projections of point M onto the image planes of two considered cameras are expressed by homogenous coordinates as follows:

$$\begin{aligned} \mathbf{m_1} &= [u_1, v_1, 1]^T, \\ \mathbf{m_2} &= [u_2, v_2, 1]^T, \end{aligned} \tag{2}$$

where u_1, v_1, u_2, v_2 are coordinates on a Camera 1 and Camera 2 image planes.

When reprojection is attempted, one must find the relationship between the coordinates of projections m_1 and m_2. Both $\mathbf{m_1}$ and $\mathbf{m_2}$ are defined in coordinate systems of their respective camera image planes. The only common point for these projections is the point M.

Positions of points m_1, m_2 are projections of a point M seen by Camera 1 and Camera 2. They can be derived by a projection of the 3D position of point M onto the image plane of Camera 1 and Camera 2 through projection matrices $\mathbf{P_1}$ and $\mathbf{P_2}$ that are defined for both cameras with the following formulas:

$$\begin{aligned} \mathbf{m_1'} &= \mathbf{P_1} \cdot \mathbf{M}, \\ \mathbf{m_2'} &= \mathbf{P_2} \cdot \mathbf{M}, \end{aligned} \tag{3}$$

where, for simplification, vectors $\mathbf{m_1'}$ and $\mathbf{m_2'}$ have been introduced. Vectors $\mathbf{m_1'}$ and $\mathbf{m_2'}$ define position of m_1 and m_2 on image plane of Camera 1 and Camera 2:

$$\begin{aligned} \mathbf{m_1'} &= \left[z_1 \cdot \mathbf{m_1}^T, 1\right]^T &= [z_1 \cdot u_1, z_1 \cdot v_1, z_1, 1]^T, \\ \mathbf{m_2'} &= \left[z_2 \cdot \mathbf{m_2}^T, 1\right]^T &= [z_2 \cdot u_2, z_2 \cdot v_2, z_2, 1]^T. \end{aligned} \tag{4}$$

This definition of vector $\mathbf{m_1'}$ (and by analogy $\mathbf{m_2'}$) shows that vector $\mathbf{m_1'}$ is homogeneous extension of position vector $\mathbf{m_1}$ multiplied by scalar z_1.

By inversion of equation (3) one can obtain position of point M in global 3D coordinate system based on position of point m_1 or m_2 seen by camera 1 or 2:

$$\begin{aligned} \mathbf{m_1'} &= \mathbf{P_1} \cdot \mathbf{M} &\Rightarrow \mathbf{M} = \mathbf{P_1}^{-1} \cdot \mathbf{m_1'}, \\ \mathbf{m_2'} &= \mathbf{P_2} \cdot \mathbf{M} &\Rightarrow \mathbf{M} = \mathbf{P_2}^{-1} \cdot \mathbf{m_2'}. \end{aligned} \tag{5}$$

Substituting the position of point M obtained from the image of point M seen by Camera 1 (m_1) into the projection equation of Camera 2, the relationship between the position of both projections of point M onto the image planes of both cameras can be obtained.

$$\underbrace{\begin{array}{l}\mathbf{m_1'} = \mathbf{P_1}\cdot\mathbf{M} \quad \Rightarrow \mathbf{M} = \mathbf{P_1}^{-1}\cdot\mathbf{m_1'} \\ \mathbf{m_2'} = \mathbf{P_2}\cdot\mathbf{M}\end{array}}_{\Downarrow} \tag{6}$$
$$\mathbf{m_2'} = \mathbf{P_2}\cdot\mathbf{P_1}^{-1}\cdot\mathbf{m_1'} = \mathbf{H_{1\to2}}\cdot\mathbf{m_1'}$$

where $\mathbf{H_{1\to2}}$ is transformation matrix (homography matrix) [21]:

$$\mathbf{H_{1\to2}} = \mathbf{P_2}\cdot\mathbf{P_1^{-1}} \tag{7}$$

So in order to obtain position $\mathbf{m_2}$ of projection of point M on the image plane of Camera 2 based on known position of projection of the same point M onto the image plane of Camera 1 we need to calculate $\mathbf{m_2'}$ vector from equation (6) and then divide first and second component of that vector by the third component (see definition 4) which is just normalization of the $\mathbf{m_2'}$ vector by its third component:

$$\begin{aligned}\mathbf{m_{2h}} &= \frac{\mathbf{m_2'}}{m_2'(3)} = \frac{\mathbf{m_2'}}{z_2} = \frac{\mathbf{H_{1\to2}}\cdot\mathbf{m_1'}}{z_2} = \\ &= \frac{\mathbf{H_{1\to2}}\cdot\left[z_1\cdot\mathbf{m_1}^T, 1\right]^T}{z_2} = \\ &= \frac{\mathbf{H_{1\to2}}\cdot\left[z_1\cdot u_1, z_1\cdot v_1, z_1, 1\right]^T}{z_2}, \\ \mathbf{m_2} &= \left[m_{2h}(1), m_{2h}(2), 1\right]^T,\end{aligned} \tag{8}$$

where $m_2'(3)$ means value of the third component of the $\mathbf{m_2'}$ vector, $m_{2h}(1)$ means value of the first component of the $\mathbf{m_{2h}}$ vector, $m_{2h}(2)$ means value of the second component of the $\mathbf{m_{2h}}$ vector.

The equation above leads to known from literature [21] [37] equation for coordinate transformation from one image space to the other.

III. CLASSICAL CALCULATION ORGANIZATION

Direct implementation of equation (8) for point transformation from one image space to the other leads to the algorithm presented as algorithm 1.

In the algorithm shown above, a, b and c are coefficients that are specific for a given depth format and will be explained further on.

The algorithm 1, in the form shown, requires the following number of operations per source image pixel:

- 1 multiplication, 1 addition, and 1 division to denormalize depth value into a distance z_1 in source image space,
- 2 multiplications during $\mathbf{m_1}$ vector construction,
- 16 multiplications and 12 additions during multiplication of $\mathbf{H_{1\to2}}$ matrix by $\mathbf{m_1}$ vector,
- 3 divisions in $\mathbf{m_2}$ and depth normalization,
- 1 division, 1 addition, and 1 multiplication for distance to depth normalization,

so in total 20 multiplications, 14 additions, and 3 divisions, for each and every pixel in the source image. Summary of necessary operations for an image of resolution W by H in provided in table 1.

IV. NEW PROJECTION FORMULATION

A significant reduction of the computation complexity can be obtained by modifying the order of calculations, as shown below.

First let us rewrite $\mathbf{m_1'}$ vector

$$\begin{aligned}\mathbf{m_1'} &= \left[z_1\cdot u_1, z_1\cdot v_1, z_1, 1\right]^T = z_1\cdot\left[u_1, v_1, 1, \frac{1}{z_1}\right]^T = z_1\cdot\mathbf{q_1}, \\ \mathbf{m_2'} &= \left[z_2\cdot u_2, z_2\cdot v_2, z_2, 1\right]^T = z_2\cdot\left[u_2, v_2, 1, \frac{1}{z_2}\right]^T = z_2\cdot\mathbf{q_2},\end{aligned} \tag{9}$$

where introduced vectors $\mathbf{q_1}$ and $\mathbf{q_2}$ have the form:

$$\begin{aligned}\mathbf{q_1} &= \left[u_1, v_1, 1, \frac{1}{z_1}\right]^T, \\ \mathbf{q_2} &= \left[u_2, v_2, 1, \frac{1}{z_2}\right]^T.\end{aligned} \tag{10}$$

With the help of the introduced vectors $\mathbf{q_1}$ and $\mathbf{q_2}$, the equation (8) can be rewritten as:

$$\begin{aligned}\mathbf{q_2} &= \frac{\mathbf{m_2'}}{m_2'(3)} = \frac{\mathbf{m_2'}}{z_2} = \frac{\mathbf{H_{1\to2}}\cdot\mathbf{m_1'}}{z_2} = \frac{\mathbf{H_{1\to2}}\cdot z_1\cdot\mathbf{q_1}}{z_2} = \\ &= \frac{z_1}{z_2}\cdot\mathbf{H_{1\to2}}\cdot\mathbf{q_1} = \frac{z_1}{z_2}\cdot\mathbf{H_{1\to2}}\cdot\left[u_1, v_1, 1, \frac{1}{z_1}\right]^T.\end{aligned} \tag{11}$$

Such formulation allows to save two multiplications during $\mathbf{m_1'}$ creation, at a price of one division during $\mathbf{q_1}$ vector creation. Next, we will show how to eliminate this division as well.

As has been said earlier, the distance z necessary for projection is commonly stored as a normalized disparity in the form of a depth map. Relation of distance z and depth sample δ is given by:

$$z = \frac{1}{\frac{\delta}{2^{bps}-1}\cdot\left(\frac{1}{z_{near}}+\frac{1}{z_{far}}\right)+\frac{1}{z_{far}}}, \tag{12}$$

where bps is the number of bits per sample that are used to store depth samples commonly 8 or 16, z_{near} and z_{far} are the nearest and the farthest distance stored in the given depth map and are used for normalized disparity denormalization into a distance z. Turning the equation we can get the inverse of distance z:

$$\frac{1}{z} = \frac{\delta}{2^{bps}-1}\cdot\left(\frac{1}{z_{near}}+\frac{1}{z_{far}}\right)+\frac{1}{z_{far}}. \tag{13}$$

This allows us to obtain the necessary in equation (11) inverse of distance $\frac{1}{z}$ directly from depth sample δ. This eliminates division operation during creation of $\mathbf{q_1}$ vector. Additionally, it eliminates the necessity of division in depth to distance conversion (12) as direct distance z is never needed as it will be shown next.

Additionally, after obtaining $\mathbf{q_2}$ vector it is easy to extract depth sample δ_2 (normalized disparity) in the second image space. Vector's $\mathbf{q_2}$ (10) fourth component contains inverse

Algorithm 1 Direct implementation of the point position transformation

```
1:  procedure Transform
2:      for $v_1 \in \{0,1,2,\ldots,height-1\}$ do
3:          for $u_1 \in \{0,1,2,\ldots,width-1\}$ do
4:              $\delta_1 \leftarrow DepthMap(u_1, v_1)$                 ▷ Read depth map value
5:              $z_1 \leftarrow \frac{1}{\delta_1 * a + b}$                 ▷ Distance calculation
6:              $m_1' \leftarrow [z_1 \cdot u_1, z_1 \cdot v_1, z_1, 1]^T$   ▷ Position vector
7:              $m_2' \leftarrow H_{1\to2} \cdot m_1'$                   ▷ Transformation of position
8:              $m_2 \leftarrow \frac{m_2'}{m_2'(3)}$                    ▷ Normalization of position
9:              $u_2 \leftarrow m_2(1)$                                ▷ Read out point coordinates
10:             $v_2 \leftarrow m_2(2)$
11:             $z_2 \leftarrow \frac{m_2'(3)}{m_2'(4)}$                 ▷ Read out distance in target image space
12:             $\delta_2 \leftarrow \left(\frac{1}{z_2} - b\right) \cdot c$
13:             $virtualDepthMap(u_2, v_2) \leftarrow \delta_2$          ▷ Store depth value
```

TABLE 1. Summary of the number of operations for an image of resolution *W* by *H* in direct implementation of the equation of point transformation from source to target image space

Description	Multiplications	Addition	Division
Distance denormalization z_1	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$
$\mathbf{m_1'}$ vector construction	$2 \cdot W \cdot H$	0	0
Multiplication by transormation matrix	$16 \cdot W \cdot H$	$12 \cdot W \cdot H$	0
$\mathbf{m_2}$ vector normalization	0	0	$3 \cdot W \cdot H$
distance z_2 normalization	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$
Summary	$20 \cdot W \cdot H$	$14 \cdot W \cdot H$	$5 \cdot W \cdot H$

distance $\frac{1}{z_2}$ which can be directly planted into the equation for depth sample δ_2, as derived from (13):

$$\delta_2 = \left(\frac{1}{z_2} - \frac{1}{z_{far}}\right) \cdot \frac{2^{bps}-1}{\left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right)}, \quad (14)$$

$$\delta_2 = \left(q_2(4) - \frac{1}{z_{far}}\right) \cdot \frac{2^{bps}-1}{\left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right)}. \quad (15)$$

In implementation, constants in equations (13)(14) can be precalculated which simplifies those equations to:

$$\begin{aligned}\frac{1}{z} &= \frac{\delta}{2^{bps}-1} \cdot \left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right) + \frac{1}{z_{far}} = \\ &= \delta \cdot a + b,\end{aligned} \quad (16)$$

$$\begin{aligned}\delta &= \left(\frac{1}{z} - \frac{1}{z_{far}}\right) \cdot \frac{2^{bps}-1}{\left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right)} = \\ &= \left(\frac{1}{z} + b\right) \cdot c,\end{aligned} \quad (17)$$

where a,b,c:

$$\begin{aligned}a &= \frac{1}{2^{bps}-1} \cdot \left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right), \\ b &= -\frac{1}{z_{far}}, \\ c &= \frac{2^{bps}-1}{\left(\frac{1}{z_{near}} + \frac{1}{z_{far}}\right)}.\end{aligned} \quad (18)$$

Up to now, we have shown that the distance z_1 is not needed to create position vector $\mathbf{q_1}$ but it still remains in equation (11) in front of the square bracket. But this multiplication also can be eliminated. Let us express the transformation matrix row-wise:

$$\mathbf{H_{1\to2}} = \left[\mathbf{g_1}^T, \mathbf{g_2}^T, \mathbf{g_3}^T, \mathbf{g_4}^T\right]^T, \quad (19)$$

where $\mathbf{g_1} \cdots \mathbf{g_4}$ are rows of transformation matrix $\mathbf{H_{1\to2}}$. Transformation operation (equation 11) can be expressed as:

$$\begin{aligned}\mathbf{q_2} =& \frac{z_1}{z_2} \cdot \mathbf{H_{1\to2}} \cdot \mathbf{q_1} = \frac{z_1}{z_2} \cdot \left[\mathbf{g_1}^T, \mathbf{g_2}^T, \mathbf{g_3}^T, \mathbf{g_4}^T\right]^T \cdot \mathbf{q_1} = \\ =& \frac{z_1}{z_2} \cdot \left[\mathbf{q_1}^T \cdot \mathbf{g_1}^T, \mathbf{q_1}^T \cdot \mathbf{g_2}^T, \mathbf{q_1}^T \cdot \mathbf{g_3}^T, \mathbf{q_1}^T \cdot \mathbf{g_4}^T\right]^T = \\ =& \frac{\left[z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_1}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_2}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_3}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_4}^T\right]^T}{z_2} = \\ =& \frac{1}{z_2} \cdot \mathbf{m_2'}.\end{aligned} \quad (20)$$

but z_2 is just the third component of $\mathbf{m_2'}$ vector, so:

$$z_2 = z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_3}^T, \quad (21)$$

which leads to:

$$\begin{aligned}\mathbf{q_2} =& \frac{\left[z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_1}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_2}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_3}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_4}^T\right]^T}{z_2} = \\ =& \frac{\left[z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_1}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_2}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_3}^T, z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_4}^T\right]^T}{z_1 \cdot \mathbf{q_1}^T \cdot \mathbf{g_3}^T} = \\ =& \left[\frac{\mathbf{q_1}^T \cdot \mathbf{g_1}^T}{\mathbf{q_1}^T \cdot \mathbf{g_3}^T}, \frac{\mathbf{q_1}^T \cdot \mathbf{g_2}^T}{\mathbf{q_1}^T \cdot \mathbf{g_3}^T}, 1, \frac{\mathbf{q_1}^T \cdot \mathbf{g_4}^T}{\mathbf{q_1}^T \cdot \mathbf{g_3}^T}\right]^T.\end{aligned} \quad (22)$$

This clearly shows that scalars z_1 cancel out due to the normalization of $\mathbf{q_2}$ vector.

Derivation (22) shows that there is no need for multiplying by distance z_1 in equation (11) since it will cancel out during $\mathbf{m_2}$ normalization.

As a consequence, there is no need for direct calculation of distance z_1.

The proposed formulation allows to elimination of two multiplications during the construction of $\mathbf{m_1'}$ vector, and one division during the calculation of distance z from normalized disparity δ (taken from depth map).

V. FAST POSITION TRANSFORMATION BY GRADUAL PROJECTION BUILD UP

Thanks to the new formulation of equation (22) for point projection from one image space to the other it is possible to further vastly speed up the transformation process. Instead of independently applying the transformation (22) for every pixel of the image, it is possible to gradually build target position vector $\mathbf{m_2'}$.

In the proposed formulation, the source positions u_1 and v_1 do not need to be multiplied by the distance z_1 to construct the source position vector $\mathbf{q_1}$ (10).

Let us express the transformation matrix $\mathbf{H_{1\to2}}$ column-wise:

$$\mathbf{H_{1\to2}} = [\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}, \mathbf{h_4}]\,, \tag{23}$$

where $\mathbf{h_1}$, $\mathbf{h_2}$, $\mathbf{h_3}$, $\mathbf{h_4}$ are column vectors of $\mathbf{H_{1\to2}}$ matrix. Then transformation equation (11) can be expressed as:

$$\begin{aligned}\mathbf{q_2} &= \frac{z_1}{z_2}\cdot\mathbf{H_{1\to2}}\cdot\mathbf{q_1} = \\ &= \frac{z_1}{z_2}\cdot[\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}, \mathbf{h_4}]\cdot\left[u_1, v_1, 1, \frac{1}{z_1}\right]^T = \\ &= \frac{z_1}{z_2}\left(\mathbf{h_1}\cdot u_1 + \mathbf{h_2}\cdot v_1 + \mathbf{h_3} + \mathbf{h_4}\cdot\frac{1}{z_1}\right)\end{aligned} \tag{24}$$

Equation (24) indicates that the result of the transformation can be built gradually by accumulation of the column-wise multiplication of transformation matrix $\mathbf{H_{1\to2}}$ by the components of $\mathbf{q_1}$ vector. This allows us to share some of the computation among pixels of the image. Because usually during whole image transformation v_1 position of the pixel changes less often than the u_1 position, we can reduce the total number of computations required to transform the entire image.

This observation leads to the Algorithm 2 for projecting pixels from one image to the other.

Proposed Algorithm 2 for pixel position transformation requires:

- 4 multiplications and 4 additions for every row of pixels - accumulation of term dependent on v,
- 4 multiplications and 4 additions for every pixel in a row - accumulation of term dependent on u,
- 1 multiplication and 1 addition for every pixel - calculation of inverse distance *invz* from depth sample,
- 4 multiplications and 4 additions for every pixel - accumulation of term dependent on *invz*,
- 3 divisions during $\mathbf{m_2'}$ vector normalization,
- 1 addition and 1 multiplication during normalization of target distance to depth sample δ_2.

A summary of the number of required operations necessary to project all pixels of the image of W by H resolution is presented in table 2.

Comparing the number of necessary operations of the proposed algorithm to the direct implementation of equation (8), which is common in the literature, summarized in table 1, we can calculate the approximate reduction rate of necessary multiplication operations:

$$\frac{(6\cdot W+4)\cdot H}{20\cdot W\cdot H} = \frac{6}{20} + \frac{4}{20\cdot W}. \tag{25}$$

where W is width and H is height of the image considered. Commonly nowadays resolution of the processed image is high $W \gg 1$, in the order of several thousand (1980 for a typical HD video image), so the second fraction is small and can be omitted:

$$\frac{(6\cdot W+4)\cdot H}{20\cdot W\cdot H} = \frac{6}{20} + \frac{4}{20\cdot W} \approx \frac{3}{10}. \tag{26}$$

Similarly, we can obtain the reduction rate of necessary additions and divisions:

$$\frac{(6\cdot W+4)\cdot H}{14\cdot W\cdot H} = \frac{6}{14} + \frac{4}{14\cdot W} \approx \frac{3}{7}, \tag{27}$$

$$\frac{3\cdot W\cdot H}{5\cdot W\cdot H} = \frac{3}{5}. \tag{28}$$

To summarize, the proposed organization of the computation requires barely 30% of multiplications, 47% additions, and 60% divisions. It is worth to point out that in the case of many implementation platforms such as signal processor or FPGA, the division operation is the most computationally expensive one, so the reduction of this kind of operation is the most important one.

VI. EXPERIMENTAL VERIFICATION

The base for our experiments was the already mentioned reference virtual view synthesis software VSRS [19]. VSRS is a CPU-only implementation that does not utilize any vector instruction speedups like SIMD SSE or AVX. We have implemented our algorithm in the above-mentioned software package also without any vector instruction optimization, so any gains in performance would be only due to the improved processing algorithm and not due to a hardware-specific implementation.

We have only modified the position projection in the forward depth map projection step and backward texture warping step to our proposed fast algorithm. We did not modify any other part of the software. Therefore the results - virtual view images - obtained with the reference VSRS implementation and the results obtained with our modified version are identical.

Algorithm 2 The proposed procedure for transformation of point location

```
1:  procedure Transform
2:      $[\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}, \mathbf{h_4}] \leftarrow \mathbf{H_{1\rightarrow 2}}$        ▷ Column-wise transformation matix
3:      $\mathbf{m'_{2c}} \leftarrow \mathbf{h_3}$        ▷ Accumulation of the first term independent of pixel position
4:      for $v_1 \in \{0, 1, 2, \ldots, height - 1\}$ do
5:          $\mathbf{m'_{2b}} \leftarrow \mathbf{m'_{2c}} + \mathbf{h_2} \cdot v_1$        ▷ Accumulation of the term dependent on $v$
6:          for $u_1 \in \{0, 1, 2, \ldots, width - 1\}$ do
7:              $\mathbf{m'_{2a}} \leftarrow \mathbf{m'_{2b}} + \mathbf{h_1} \cdot u_1$        ▷ Accumulation of the term dependent on $u$
8:              $\delta_1 \leftarrow DepthMap(u_1, v_1)$        ▷ Read depth sample
9:              $invz_1 \leftarrow \delta_1 * a + b$        ▷ Distance denormalization
10:             $\mathbf{m'_2} \leftarrow \mathbf{m'_{2a}} + \mathbf{h_4} \cdot invz_1$        ▷ Accumulation of the term dependent on $z$
11:             $\mathbf{m_2} \leftarrow \frac{\mathbf{m'_2}}{m'_2(3)}$        ▷ Target image position normalization
12:             $u_2 \leftarrow m_2(1)$        ▷ Pixel coordinate readout
13:             $v_2 \leftarrow m_2(2)$
14:             $invz_2 \leftarrow m_2(4)$        ▷ Distance readout and normalization
15:             $\delta_2 \leftarrow (invz_2 - b) \cdot c$
16:             $virtualDepthMap(u_2, v_2) \leftarrow \delta_2$        ▷ Target depth sample storage
```

TABLE 2. Summary of the number of required operations necessary to project all pixels of the image of ***W*** by ***H*** resolution

Description	Multiplications	Additions	Divisions
Accumulation of term dependent on v	$4 \cdot H$	$4 \cdot H$	0
Accumulation of term dependent on u	$4 \cdot W \cdot H$	$4 \cdot W \cdot H$	0
Calculation of $invz$ from depth sample	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$	0
Accumulation of term dependent on $invz$	$4 \cdot W \cdot H$	$4 \cdot W \cdot H$	0
m_2 vector normalization	0	0	$3 \cdot W \cdot H$
Distance z_2 to depth sample conversion	$1 \cdot W \cdot H$	$1 \cdot W \cdot H$	0
Total	$(6 \cdot W + 4) \cdot H$	$(6 \cdot W + 4) \cdot H$	$3 \cdot W \cdot H$

We have used several well-known and recognizable multiview test sequences which are provided with high-quality depth maps. This set includes linear [38], arc multiview content [39] [40] [41] [41] [42] as well as lightfield (supermultiview) [43] content.

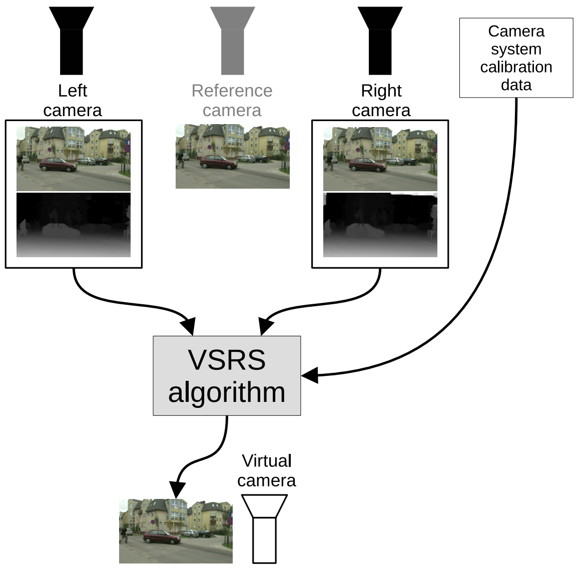


FIGURE 2. The schematic depicting the test scenario

For all test sequences, we have rendered virtual views with the use of the original unmodified VSRS as well as with the use of VSRS with our calculation method. The virtual views were generated from the data for two views, specified as right and left view. The data included images, depth maps, and camera parameters for two views. No other data was provided for the algorithm. The virtual view was situated in the same exact place as the third camera was situated during the acquisition of the sequence. The schematic depicting the test scenario is shown in Fig 2. Data from left and right camera (RGB image and depth image) together with the calibration data of the camera setup are the input data for the VSRS algorithm, generating a virtual view. For the purpose of the tests, the position of the virtual camera is the same as the position of the middle camera. The middle camera data (RGB image) is used exclusively for reference and those data are not used by the VSRS algorithm.

The quality of the resulting virtual view, 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 was exactly the same as for the reference method. This is due to the fact that the presented method gives exactly the same numerical results for a virtual view image as the reference method. In the Fig 3 a sample virtual view generated by VSRS is shown for Poznan Street and BBB Flowers sequence. For reference also the view from a real camera situated in the same spot is shown. The virtual view errors can be noticed mainly on some object borders, like the

TABLE 3. Summary of the multiview test sequences used in the experiments

Sequence	Resolution	Left reference view	Right reference view	Virtual view
Poznan Street [38]	1920x1080	3	5	4
Poznan Fencing [42]	1920x1080	0	2	1
Poznan Blocks [39]	1920x1080	0	2	1
BBB Butterfly [43]	1280x768	6	32	19
BBB Flowers [43]	1280x768	6	32	19
Soccer Arc [40]	1920x1080	25	32	28
Ballet [41]	1024x768	3	5	4
Breakdancers [41]	1024x768	2	4	3

stems and petals of the flowers for BBB Flowers. For Poznan Street please note the errors on pedestrian in the background and on the car on the left of the image.

In order to compare the performance of both implementations we have measured the time it takes for both versions of the software to create respective virtual views.

All experiments have been performed on a single PC without any GPU enhancement as the VSRS implementation is purely a CPU implementation. All essential parameters have been summarized in the Table 4.

TABLE 4. Parameters of the computer used to measure the algorithm efficiency

Processor	Intel i7-8700K 3.7GHz
Memory	32 GB
GPU	Built-in in CPU
HDD	WD 5TB
Operating system	Windows 10

VII. RESULTS AND FUTURE WORK

The results of the experiment are shown in Table 5. Each reported time is expressed in milliseconds and is the time of synthesis of a virtual view for the whole test sequence. It can be seen that the time of the virtual view generation is significantly shorter for our implementation. The virtual view generation time is shorter by up to 29%. For most of the sequences, the time was shorter by more than 20%. The average value of the percentage of speedup for the eight tested sequences is 22.58%. The smallest gain in performance was observed for the Ballet and Breakdancers sequences. These are sequences with low resolution, therefore the accumulation of the calculations does not provide that much of a gain as for the sequences with higher resolutions. The proposed method must recalculate the coefficients for every line separately, thus lower horizontal resolution will reduce the performance gain. A slightly lower speedup value for the Poznan Blocks sequence appears to be justified by a more complex depth map with many small patches of different depths. In such a situation the speedup offered by our algorithm is slightly reduced due to the fact that if a new depth value is encountered, the coefficients need to be recalculated. The proposed method performs best for cases where there are large patches of constant values in depth map. The properties of the depth maps that influence the performance of the presented algorithm will be studied in our future work as we aim to provide a metric for estimating the possible gains provided by our method. Based on this data, the depth map preprocessing algorithms will be proposed that allow to speed up the reprojection-based view synthesis process.

The obvious concept for future work is algorithm scalability by using parallel computations. Since processing of each image row is independent from others, the work could be split to multiple work units and distributed across CPU/GPU cores.

VIII. SUMMARY

In the paper, we have presented a new formulation of the position transformation operation which results in a new fast position transformation algorithm. The proposed formulation is a result of authors analysis of the existing algorithm used in DERS software. The devised method of changing the order of operations and replacing the variables used in the existing algorithm resulted in a significant increase of the execution speed while providing exactly identical output data as the unmodified algorithm. The proposed algorithm can be used to speed up view synthesis and depth estimation algorithms that heavily rely on position transformations. The algorithm can easily be applied wherever the original version is used. It provides a significant speedup of approximately 20% of computation time at virtually no cost at all since the speedup is based on the rearranging of matrix operations that allow to reuse parts of calculations for consecutive pixels of reprojected views.

APPENDIX A VARIABLES USED IN THE TEXT

In the table below we summarize the most important variables used in the text.

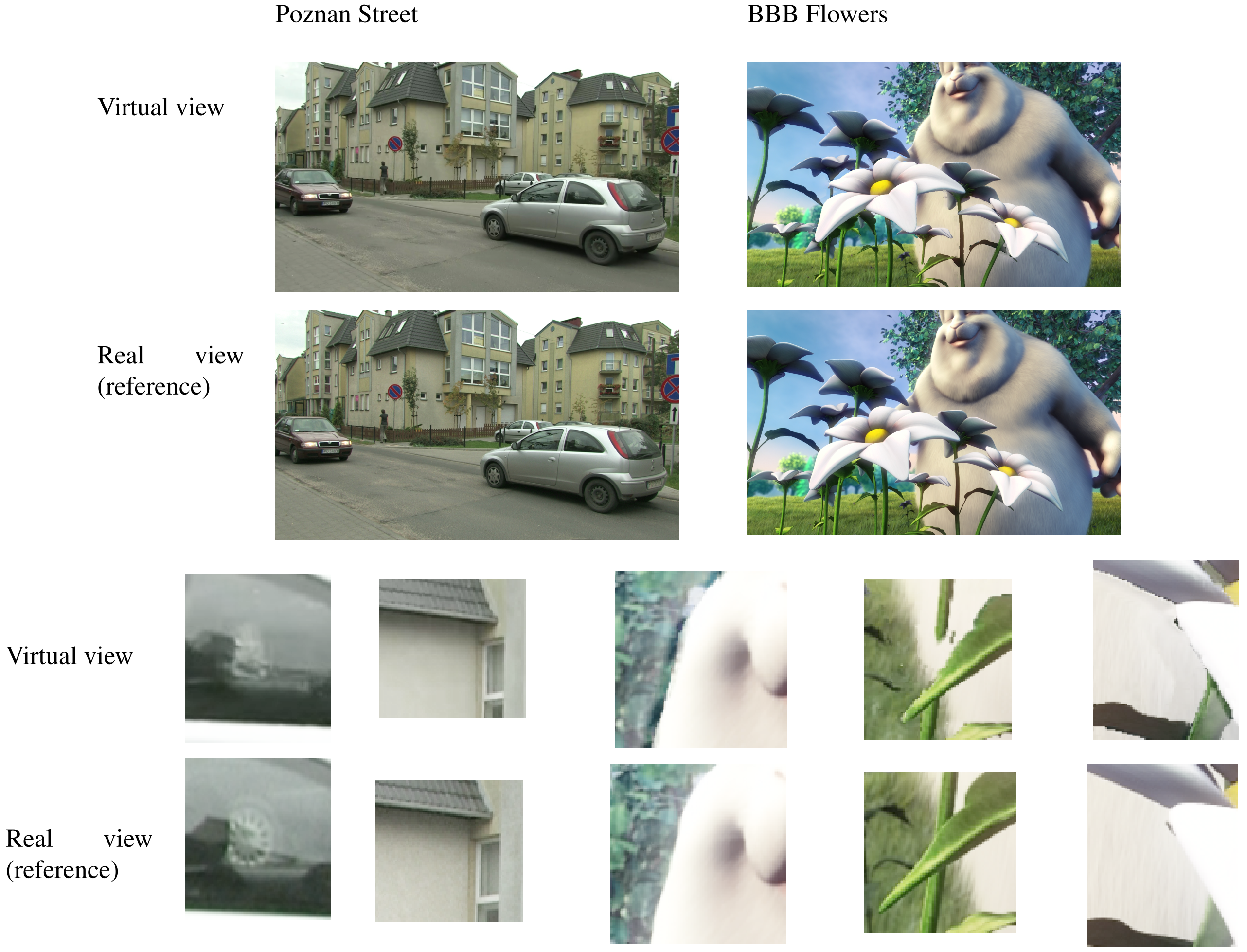


FIGURE 3. Sample virtual views obtained during the tests. Poznan Street to the left, BBB Flowers to the right. Views from real cameras are given for reference below virtual views. Some details of the images are shown in greater scale in the two lower rows.They demonstrate the errors in the virtual view generation common to the VSRS, from left to right: improper handling of transparency, blurring, sharp edges of objects in virtual view, inpainted areas with missing features (features not present on any source views), ghosting due to different lighting of source views

TABLE 6. Variables used in the text

Variable name	Description
$\mathbf{M}$	point coordinates in 3D space
m_x	point coordinates in 2D image space of camera *x*
$\mathbf{m'_x}$	homogenous point coordinates in 2D image space of camera *x*
u_x, v_x	horizontal and vertical coordinates of point in image space of camera *x*
z_x	homogenous coordinates multiplier for camera *x*
$\mathbf{P_x}$	projection matrix of camera *x*
$\mathbf{H_{1\to 2}}$	transformation (homography) matrix between camera 1 and 2
$\mathbf{m_{xh}}$	normalized homogenous point coordinates $\mathbf{m}'_x$
W, H	image width and height, correspondingly

REFERENCES

[1] Ozgur Oyman, Rob Koenen, Paul Higgs, Chris Johns, Richard Mills, and Mick O'Doherty. Virtual reality industry forum's view on state of the immersive media industry. *SMPTE Motion Imaging Journal*, 128(8):91–96, 2019.

[2] Olgierd Stankiewicz, Marek Domański, Adrian Dziembowski, Adam Grzelka, Dawid Mieloch, and Jarosław Samelak. A free-viewpoint television system for horizontal virtual navigation. *IEEE Transactions on Multimedia*, 20(8):2182–2195, 2018.

[3] S. Zinger, L. Do, and P.H.N. de With. Free-viewpoint depth image based rendering. *Journal of Visual Communication and Image Representation*, 21(5):533–541, 2010. Special issue on Multi-camera Imaging, Coding and Innovative Display.

[4] Michael Schmeing and Xiaoyi Jiang. *Depth Image Based Rendering*, pages 279–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[5] Xiaodong Chen, Haitao Liang, Huaiyuan Xu, Siyu Ren, Huaiyu Cai, and Yi Wang. Virtual view synthesis based on asymmetric bidirectional dibr for 3d video and free viewpoint video. *Applied Sciences*, 10(5), 2020.

[6] Zengming Deng and Mingjiang Wang. Reliability-based view synthesis for free viewpoint video. *Applied Sciences*, 8(5), 2018.

[7] Xiaodong Chen, Haitao Liang, Huaiyuan Xu, Siyu Ren, Huaiyu Cai, and Yi Wang. Artifact handling based on depth image for view synthesis. *Applied Sciences*, 9(9), 2019.

[8] Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 1–17, Cham, 2020. Springer International Publishing.

[9] Masayuki Tanimoto, Mehrdad Panahpour Tehrani, Toshiaki Fujii, and Tomohiro Yendo. Ftv for 3-d spatial communication. *Proceedings of the IEEE*, 100(4):905–917, 2012.

[10] Christoph Fehn, René De La Barré, and Siegmund Pastoor. Interactive 3-dtv-concepts and key technologies. *Proceedings of the IEEE*, 94(3):524–538, 2006.
[11] Adrian Dziembowski, Dawid Mieloch, Olgierd Stankiewicz, Marek Domański, Gwangsoon Lee, and Jeongil Seo. Virtual view synthesis for 3dof+ video. In *2019 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2019.
[12] Adrian Dziembowski, Adam Grzelka, Dawid Mieloch, Olgierd Stankiewicz, Krzysztof Wegner, and Marek Domański. Multiview synthesis—improved view synthesis for virtual navigation. In *2016 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2016.
[13] Guangcheng Wang, Kui Jiang, Ke Gu, Hongyan Liu, Hantao Liu, and Wenjun Zhang. Coarse- and fine-grained fusion hierarchical network for hole filling in view synthesis. *IEEE Transactions on Image Processing*, 33:322–337, 2024.
[14] Jiahe Wang and Jiayue Liu. Virtual viewpoint rendering method based on multi-threshold layering. In *2023 8th International Conference on Image, Vision and Computing (ICIVC)*, pages 639–644, 2023.
[15] Zhihui Ke, Xiaobo Zhou, Dadong Jiang, Hao Yan, and Tie Qiu. Collabvr: Reprojection-based edge-client collaborative rendering for real-time high-quality mobile virtual reality. In *2023 IEEE Real-Time Systems Symposium (RTSS)*, pages 304–316, 2023.
[16] Hui Chen and Lei Zhu. Digital twin oriented virtual view rendering algorithm based on k-means. In *2023 IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 116–119, 2023.
[17] Yifan Wang, Fuzheng Yang, Ying Chen, and Wei Zhang. Virtual view synthesis using joint information from multi-view. *Journal of Visual Communication and Image Representation*, 92:103799, 2023.
[18] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. Mods: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 141:81–93, 2015.
[19] Takanori Senoh, Jamamoto Kenji, Tetsutani Nobuji, Yasuda Hiroshi, and Krzysztof Wegner. View synthesis reference software (vsrs) 4.2 with improved inpainting and hole filing. In *ISO/IEC JTC1/SC29/WG11 MPEG2017*, 2017.
[20] Pasquale Lafiosca and Marta Ceccaroni. Rectifying homographies for stereo vision: Analytical solution for minimal distortion. In Kohei Arai, editor, *Intelligent Computing*, pages 484–503, Cham, 2022. Springer International Publishing.
[21] Richard I Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, nov 1999.
[22] Dawid Mieloch and Adam Grzelka. Segmentation-based method of increasing the depth maps temporal consistency. *International Journal of Electronics and Telecommunications*, 64(3), 2018.
[23] Yanwen Qin, Xin Jin, Yanqin Chen, and Qionghai Dai. Enhanced depth estimation for hand-held light field cameras. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2032–2036. IEEE, 2017.
[24] Ségolène Rogge, Daniele Bonatto, Jaime Sancho, Rubén Salvador, Eduardo Juarez, Adrian Munteanu, and Gauthier Lafruit. Mpeg-i depth estimation reference software. In *2019 International Conference on 3D Immersion (IC3D)*, pages 1–6, 2019.
[25] Xiaoyue Wan, Zhuo Chen, and Xu Zhao. View consistency aware holistic triangulation for 3d human pose estimation. *Computer Vision and Image Understanding*, 236:103830, 2023.
[26] Maciej Kurc. Hybrid techniques of depth map estimation and their application in three-dimensional video systems. *PhD Dissertation at Poznan University of Technology*, 2019.
[27] Yun-Suk Kang and Yo-Sung Ho. Disparity map generation for color image using tof depth camera. In *2011 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, 2011.
[28] Haoliang Zhao, Huizhou Zhou, Yongjun Zhang, Yong Zhao, Yitong Yang, and Ting Ouyang. Eai-stereo: Error aware iterative network for stereo matching. In *Proceedings of the Asian Conference on Computer Vision*, pages 315–332, 2022.
[29] Gangwei Xu, Junda Cheng, Peng Guo, and Xin Yang. Attention concatenation volume for accurate and efficient stereo matching. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12971–12980, 2022.
[30] Junpeng Jing, Jiankun Li, Pengfei Xiong, Jiangyu Liu, Shuaicheng Liu, Yichen Guo, Xin Deng, Mai Xu, Lai Jiang, and Leonid Sigal. Uncertainty guided adaptive warping for robust and efficient stereo matching. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3295–3304, 2023.
[31] Gangwei Xu, Yun Wang, Junda Cheng, Jinhui Tang, and Xin Yang. Accurate and efficient stereo matching via attention concatenation volume. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2461–2474, 2024.
[32] Xiangyin Meng, Jie Wen, Yang Li, Chenlong Wang, and Jingzhen Zhang. Dfnet-trans: An end-to-end multibranching network for depth estimation for transparent objects. *Computer Vision and Image Understanding*, 240:103914, 2024.
[33] Shuai Guo, Jingchuan Hu, Kai Zhou, Jionghao Wang, Li Song, Rong Xie, and Wenjun Zhang. Real-time free viewpoint video synthesis system based on dibr and a depth estimation network. *IEEE Transactions on Multimedia*, pages 1–16, 2024.
[34] Xiongli Chai, Feng Shao, Qiuping Jiang, and Yo-Sung Ho. Mstgar: Multioperator-based stereoscopic thumbnail generation with arbitrary resolution. *IEEE Transactions on Multimedia*, 22(5):1208–1219, 2020.
[35] Xiongli Chai, Feng Shao, Qiuping Jiang, and Yo-Sung Ho. Roundness-preserving warping for aesthetic enhancement-based stereoscopic image editing. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1463–1477, 2021.
[36] Jakub Stankowski and Adrian Dziembowski. Fast view synthesis for immersive video systems. In *28th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision : WSCG 2020*, 2020.
[37] Bogusław Cyganek and J. Paul Siebert. *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, Ltd, 2009.
[38] Marek Domański, Tomasz Grajek, Krzysztof Klimaszewski, Maciej Kurc, Olgierd Stankiewicz, Jakub Stankowski, and Krzysztof Wegner. Poznań multiview video test sequences and camera parameters m17050. In *ISO/IEC JTC1/SC29/WG11, MPEG2009*, 2009.
[39] Marek Domański, Adrian Dziembowski, Agnieszka Kuehn, Maciej Kurc, Adam Łuczak, Dawid Mieloch, Jakub Siast, Olgierd Stankiewicz, and Krzysztof Wegner. Poznan blocks - a multiview video test sequence and camera parameters for free viewpoint television m32243. In *ISO/IEC JTC1/SC29/WG11 MPEG2014*, 2014.
[40] P Goorts. Real-time adaptive plane sweeping for free viewpoint navigation in soccer scenes. *doctoral dissertation*, 2014.
[41] C. Lawrence, Zitnick Sing, Bing Kang, Matthew Uyttendaele, Simon A. J. Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM SIGGRAPH 2004 Papers*, 2004.
[42] Adrian Dziembowski, Dawid Mieloch, and Marek Domański. Color-corrected poznan fencing sequence m48095. In *ISO/IEC JTC1/SC29/WG11 MPEG2019*, 2019.
[43] P Kovacs, A Fekete, K Lackner, VK Adhikarla, and A Zare. Big buck bunny light-field test sequences m35721. In *ISO/IEC JTC1/SC29/WG11 MPEG2015*, 2015.

KRZYSZTOF WEGNER (SM) received the M.Sc. degree from the Poznan University of Technology, in 2008. He is a co-author of several papers on free view television, depth estimation, and view synthesis. His professional interests include video compression in multipoint view systems, depth estimation form stereoscopic images, view synthesis for free view television, and face detection and recognition. He was involved in ISO standardization activities, where he contributes to the development of the future 3D video coding standards. He was a Co-Chair of the MPEG FTV Group, which aimed at the development of a new generation of coding standards for super-multiview displays and free-viewpoint navigation. He is a Senior Member of Institute of Electrical and Electronics Engineers (IEEE)

TOMASZ GRAJEK (M'14—SM'18) received his M.Sc. and Ph.D. degrees from Poznan University of Technology in 2004 and 2010 respectively. He has been leading several projects for industrial research and development. At present he is an assistant professor at the Institute of Multimedia Telecommunications. He is an author and co-author of over eighty papers (journals, proceedings of international conferences, and also MPEG/JPEG databases) on digital video compression, entropy coding and modeling of advanced video encoders. He holds ten patents (EPO and US) as well as several patent applications. He is a Senior Member of Institute of Electrical and Electronics Engineers (IEEE) and Member of Polish Society for Theoretical and Applied Electrical Engineering (PTETiS).

KRZYSZTOF KLIMASZEWSKI (M'87) received M.S. degree in telecommunications in 2005 and Ph.D. degree in telecommunications in 2012 from the Poznan University of Technology, Poznan, Poland. He took part in several projects, both academic and industrial, where he worked as a image and video processing engineer. He is an Assistant Professor with the Institute of Multimedia Telecommunications, Poznan University of Technology, Poznan, Poland. His research interests include image and video processing and embedded systems. He is an author and co-author of more than 60 publications, mostly in the image and video processing fields. He is a co-author of 11 patents.