CHAPTER

1 Multiview video: Acquisition, processing, compression, and virtual view rendering

Olgierd Stankiewicz*, Gauthier Lafruit†, Marek Domański*
*Poznań University of Technology, Poznań, Poland** *Brussels University (French wing), Brussels, Belgium†*

ACRONYMS

2D	two-dimensional
3D	three-dimensional
AVC	advanced video coding
CAVE	Cave Automatic Virtual Environment
DERS	Depth Estimation Reference Software
DIBR	depth image-based rendering
FTV	Free viewpoint TV
HEVC	high efficiency video coding
3D-HEVC	high efficiency video coding with depth maps
HMD	Head Mounted Device
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union-Telecommunication sector
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group
MVD	multiview video plus depth
VCEG	Video Coding Experts Group
VR	Virtual Reality
VSRS	View Synthesis Reference Software

1.1 MULTIVIEW VIDEO

Ever since the success of three-dimensional (3D) games where the user selects his/her own viewpoint to a premodeled 3D scene, much interest has risen to be able to mimic this functionality also on real content. In 1999, the science-fiction movie The Matrix popularized the bullet time effect, showing a continuously changing camera viewpoint around the action scene. This was obtained by cleverly triggering and

Academic Press Library in Signal Processing, Volume 6. https://doi.org/10.1016/B978-0-12-811889-4.00001-4

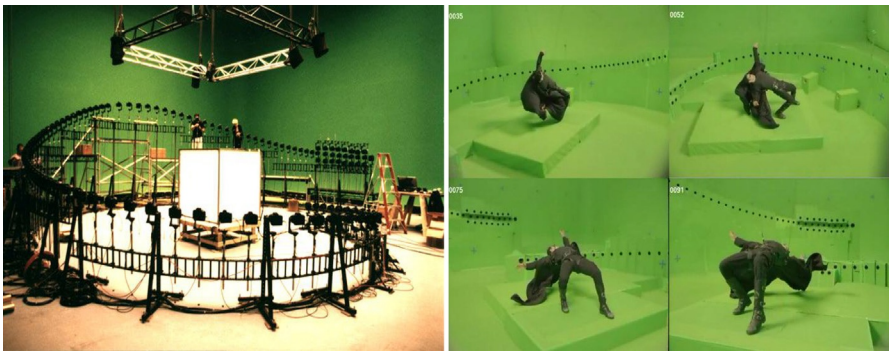

FIG. 1.1

Hundreds of cameras to capture The Matrix bullet effect.

© *Wikipedia.*

interpolating hundreds of camera views, cf. Fig. 1.1, which we refer as multiview video in the remainder of the chapter.

Think of the possibilities if we could create such Free Navigation effect on the fly at low production cost, e.g., in sports events, where each individual TV viewer can choose his/her own viewpoint to the scene.

Moreover, synthesizing two adjacent viewpoints at any moment in time supports stereoscopic viewing, and closing the loop with a stereoscopic Head Mounted Device (HMD) that continuously renders the stereo viewpoints corresponding to the actual HMD position, brings Virtual Reality (VR) one step closer to a "Teleported Reality," also called Cinematic VR.

Such topics are actively explored in many audiovisual standardization committees. Joint Photographic Experts Group (JPEG) and Moving Picture Experts Group (MPEG), for instance, are heavily involved in getting new multimedia functionalities off the ground, far beyond simple two-dimensional (2D) pictures and video. More specifically, 3D data representations and coding formats are intensely studied to target a wide variety of applications [1], including VR.

1.1.1 MULTIVIEW VIDEO AND 3D GRAPHIC REPRESENTATION FORMATS FOR VR

The specificity of VR over single view video is that VR renders content from many user-requested viewpoints. This puts many challenges in modeling the data, especially when reaching out for photorealistic Cinematic VR.

A striking example of this challenge is the multiformat data representation shown in Fig. 1.2 [2], capturing actors (e.g., Haka dancers) with close to hundreds of multimodal cameras (RGB, IR, depth) and transforming this data successively into Point Clouds and 3D Meshes with 2D Textures. In this way, rendering under any perspective viewpoint becomes possible at relatively low cost at the client side, through the OpenGL/DirectX 3D rendering pipeline.

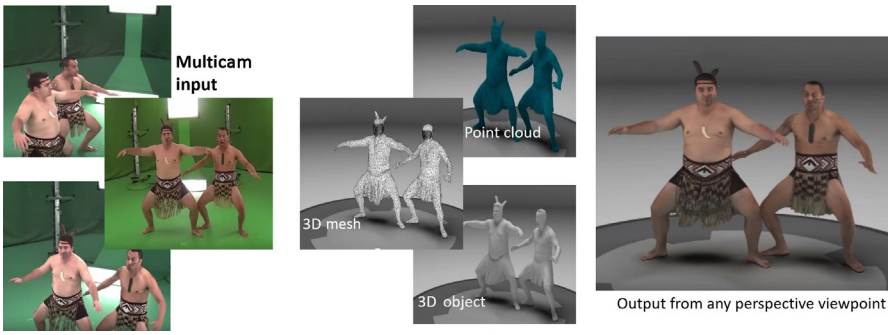

FIG. 1.2

From multimodal camera input (left) to Point Clouds and 3D meshes (middle) for any perspective viewpoint visualization (right).

These successive transformations, however, involve heavy preprocessing steps, and if not sufficient care is taken in cleaning and modeling the data, the rendering might look synthetic with too sharp object silhouettes and unnatural shadings, as if they came out of a discolored old movie.

Other solutions that do not explicitly involve geometry information are a good alternative for synthesizing virtual views. Because they merely use the input RGB images and depth images, they are referred to as depth image-based rendering (DIBR) and are the main subject of this chapter.

1.1.2 SUPER-MULTIVIEW VIDEO FOR 3D LIGHT FIELD DISPLAYS

The feeling of immersion can also be obtained with large super-multiview (SMV) or Light Field displays instead of HMDs, cf. Fig. 1.3, that project slightly varying images in hundreds of adjacent directions, so creating a 3D Light Field with perceptively correct views. Ref. [3] interpolates a couple of dozens of camera feeds to create these hundreds of views. Anybody in front of the display will then be able to capture with his/her eyes the correct stereo pair of images, without wearing stereo glasses. In a sense, such Light Field display is the glasses-free Cave Automatic Virtual Environment (CAVE) equivalent of Cinematic VR with HMD.

1.1.3 DIBR SMOOTH VIEW INTERPOLATION

Image-based rendering (IBR) solutions, solely using the input views and some image processing are more appealing for photorealistic rendering, especially in live transmission applications. Unfortunately, the solution is less simple than was presented in Fig. 1.1, since it involves much more than switching between input camera views.

Very challenging are the discontinuous jumps between adjacent views that have to be resolved with DIBR techniques, where depth plays a critical role in nonlinearly interpolating adjacent views. Figs. 1.4 [6] and 1.5 [7] (the reader is invited to use

FIG. 1.3

Holografika Super-MultiView 3D display at Brussels University: (left) clear perspective changes, and (right) a MPEG test sequence visualized.

FIG. 1.4

Fencing view interpolation.

FIG. 1.5

Soccer view interpolation.

© *Hasselt University; P. Goorts, S. Maesen, M. Dumont, S. Rogmans, P. Bekaert, Free viewpoint video for soccer using histogram-based validity maps in plane sweeping, in: Proceedings of The International Conference on Computer Vision Theory and Applications (VISAPP), 2014, pp. 378–386, https://www.youtube.com/watch?v=6MzeXeavE1s.*

the links in the references for a video animation) show that very good results can be obtained with as little as a dozen of cameras cleverly positioned around the scene.

1.1.4 BASIC PRINCIPLES OF DIBR

View interpolation creates a continuous visual transition between adjacent discrete camera views by synthesizing spatial views in between them with DIBR. To better apprehend the process of this depth-based view interpolation, let us explain it with a simple experiment: the view interpolation of fingers from both hands. Let us place one finger of our left hand at arm length in front of our nose, and one finger of our right hand touching the tip of our nose, as shown in Fig. 1.6. And now, let us blink our eyes, alternating between left and right eye, keeping one eye open, while the other is closed.

What do we observe? When our left eye is open, we see our front right-hand finger a couple of centimeter more to the right of our rear left-hand finger. Conversely, when our right eye is open, we see our front right-hand finger a couple of centimeter more to the left of our rear left-hand finger. Our front finger, which is very close to our eyes, clearly undergoes a large displacement when alternating between our eyes, while our rear finger is hardly moving.

This phenomenon is called disparity, which by definition is the displacement from the left to the right camera view (our eyes) of a 3D point in space. Obviously, the disparity of any 3D point in space is inversely proportional to its depth: our rear left-hand finger with a large depth will have a much smaller disparity than our front

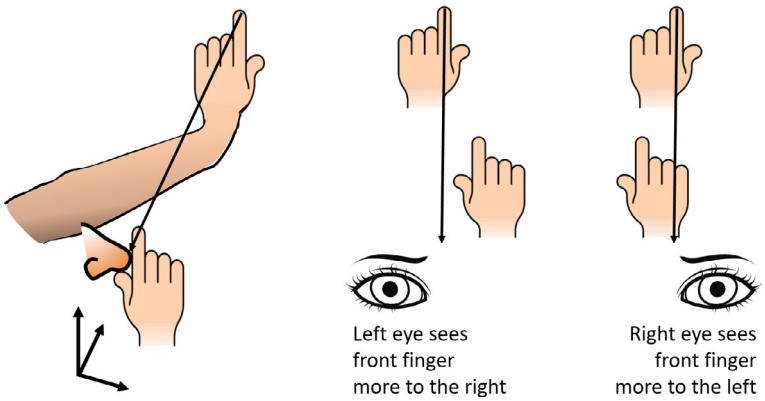

FIG. 1.6

DIBR view interpolation basic principles.

right-hand finger. This phenomenon is exactly what will be exploited for depth-based view interpolation.

Indeed, suppose we would like to look to the scene from a third eye, virtually positioned more or less halfway between our left and right eye. Let us say that this virtual third eye is—starting from the left eye—positioned at 3/8 of the distance between left and right eye, cf. Fig. 1.7, which shows the left end right eye view of Fig. 1.6 combined.

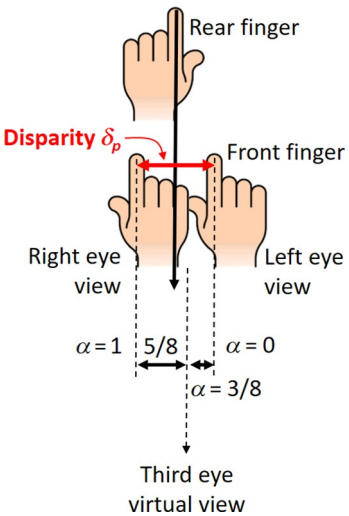

FIG. 1.7

DIBR principle in synthesizing a new virtual view at fractional position α from two camera views.

If we want to synthesize the image that this virtual third eye would see, all we have to do is to displace all pixels we see in our right eye to the right with 5/8 ($=1-3/8$) of its corresponding disparity δ_p, where the subscript p clearly indicates that the disparity is pixel p dependent.

In general, for the virtual third eye fractional position α ($\alpha=0$ at the left eye, and $\alpha=1$ at the right eye), the pixels we see in our right eye should be displaced with $1-\alpha$ their disparity δ_p to the right in order to create the corresponding virtual view. For $\alpha=0$ (the virtual view corresponds to the left view), all pixels of the right view will be displaced to the right with their disparity δ_p, recovering almost perfectly the left view.

Notice the subtle "almost perfectly" in the previous sentence. Actually, most often, adjacent pixels at each side of an object silhouette will not be displaced with the same amount from left to right, since their depth and hence their disparity is likely to be very different. This results in unfilled, empty spaces around the objects' silhouette of the virtual view, which is solved by:

- Not only displace the pixels from the right image to the virtual position α, but also displace the pixels from the left image to the virtual position α
- Blend the so-obtained images, drastically reducing the empty spaces
- The remaining empty cracks are filled with inpainting techniques, cf. Section 1.5.3, that may be compared to ink that is dropped at one border of the crack and diffuses its color to the other side of the crack (note that the diffusion process propagates in both directions, in a similar way as the aforementioned left-to-virtual or right-to-virtual image transformations are done)

More details about the DIBR steps are provided in Section 1.5.

1.1.5 DIBR VS. POINT CLOUDS

Is DIBR on multiview video sequences the best approach to obtain nice Free Navigation effects? Not necessarily. As shown in Fig. 1.2 already, many different data formats can serve the purpose of Free Navigation. In particular, the Point Cloud data representation of Fig. 1.8 [8] provides all means to freely navigate into the scene. At large distances, cf. Fig. 1.8(A), the effect is very nice. However, at shorter viewing distances, cf. Fig. 1.8(C)-(E), blocky splatting effects with increased point sizes start to occur.

Fig. 1.9, gives a glimpse on the preprocessing and rendering quality differences between DIBR (left—use the link of the reference to see the animation [9]) and Point Clouds (right). Moreover, to obtain this high-density Point Cloud, many different acquisition positions of the Time-of-Flight Lidar scanner had to be covered, followed by an intense data preparation preprocessing steps [10]. DIBR techniques, on the contrary, restrict themselves to depth extraction and image warping operations to obtain an acceptable rendering without too many noticeable artifacts.

Nevertheless, Fig. 1.10 shows that DIBR and Point Cloud representations are mathematically very similar. In essence, in a Point Cloud representation, each 3D

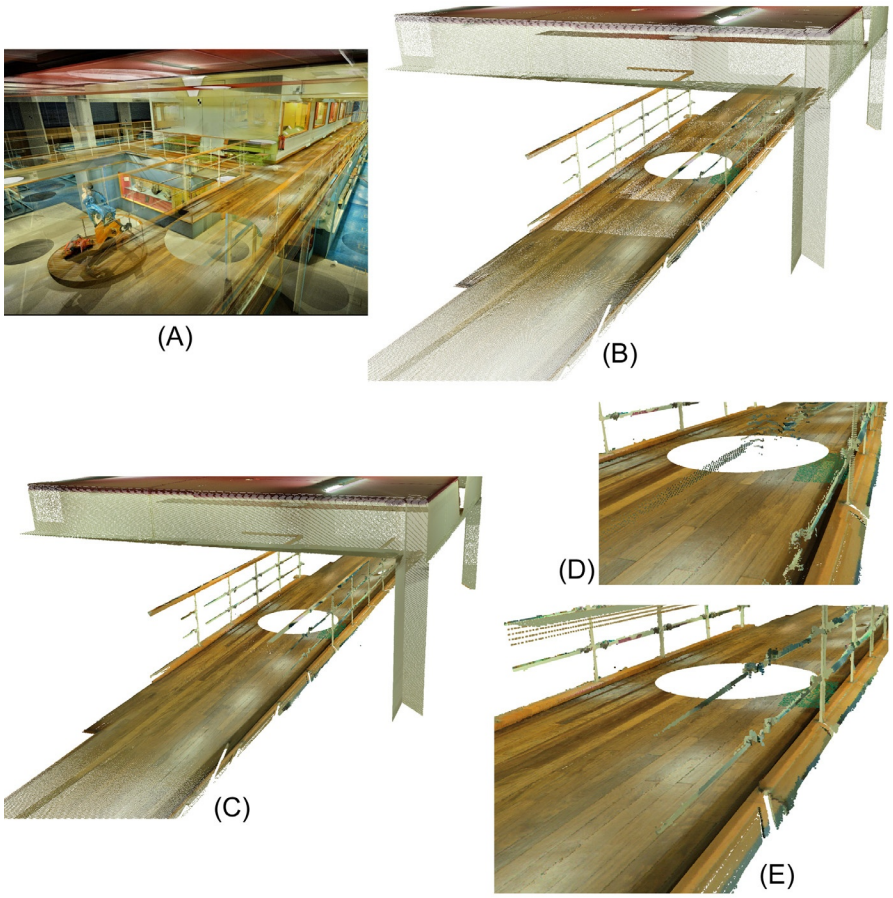

FIG. 1.8

London Science Museum Point Cloud (A) decomposed into individual clusters (B) taken from the position indicated by the empty circles. Splat rendering in (C) to (E).

ScanLab projects Point Cloud data sets, http://grebjpeg.epfl.ch/jpeg_pc/index_galleries.html.

point in space captured by a Time-of-Flight active depth sensing device is given by its position (x,y,z) and the light colors it emits in all directions (referred to as a BRDF).

Its complementary image-based representation will capture these light rays into RGB images, e.g., pixel (u,v) captures the light ray under angles (θ, φ) emanating from the point (x,y,z). Capturing multiple images from different viewpoints gives the possibility to estimate depth for each point (e.g., by stereo matching), adding depth and hence resulting in a fully corresponding DIBR representation of the Point Cloud.

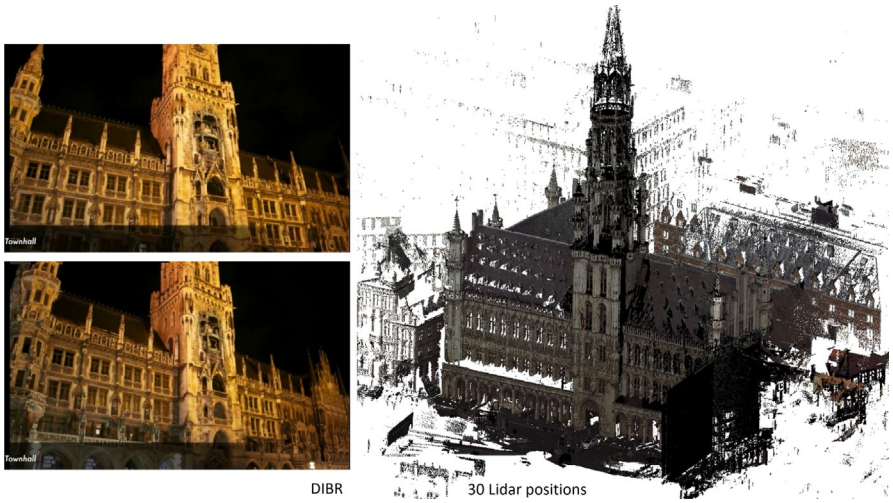

FIG. 1.9

DIBR perspective view synthesis (left) vs. Point Cloud rendering (right).

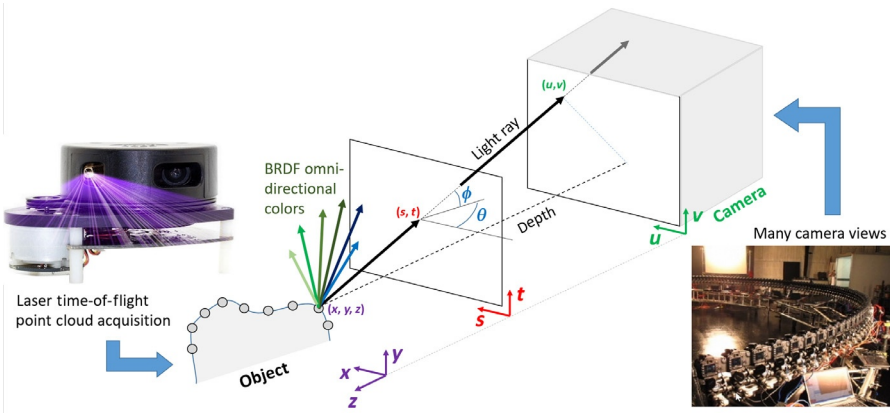

FIG. 1.10

Point Cloud data format (left) and its equivalence with DIBR (right).

The similarity between DIBR and Point Clouds will become more apparent in Section 1.5. More details about this discussion can also be found in the JPEG/MPEG JAhG (Joint Ad-Hoc Group) Light/Sound Fields technical output document [11], as well as in Ref. [1].

Though the underlying data formats are known to be mathematically equivalent between DIBR and Point Clouds, the compression and rendering qualities may differ and heavily depend on the preprocessing level of the acquired data. DIBR is the

FIG. 1.11

MPEG test sequences used in the DIBR exploration activities.

preferred technique proposed in the MPEG-FTV group for Free Navigation and SMV applications. We have therefore selected the DIBR technique to be presented in more details in the remainder of the chapter.

1.1.6 DIBR, MULTIVIEW VIDEO, AND MPEG STANDARDIZATION

MPEG, which mission is to develop audiovisual compression standards for the industry, is currently actively involved in all aforementioned techniques: DIBR, Point Cloud, and 3D mesh coding. The interested reader is referred to Ref. [1] for an in-depth overview of use cases and 3D related data representation and coding formats.

In particular, in the MPEG terminology, the multicamera DIBR technology this chapter is devoted to, is referred to as multiview video. Multiview test sequences are made available to the community, helping in developing improved depth estimation, view synthesis, and compression techniques, cf., respectively, Sections 1.4–1.6. Some of the test sequences are presented in Fig. 1.11 and have been acquired with the acquisition systems which will be presented in Section 1.2.4.

1.2 MULTIVIEW VIDEO ACQUISITION

1.2.1 MULTIVIEW FUNDAMENTALS

The term "multiview" is widely used to describe various systems. In this section we are dealing exclusively with the systems where the outputs from all cameras are

processed jointly in order to produce some general representation of the scene. Therefore, for example, the video surveillance systems where video is registered and viewed from each camera independently are out of the scope of our considerations.

The goal of multiview video acquisition considered is to produce a 3D representation of a scene. Such 3D representation is needed to enable functionalities related to VR free navigation, e.g., to synthesize virtual views as seen from arbitrary positions in the scene. Currently, the most commonly used format of such representation is multiview video plus depth (MVD). In MVD many 2D (flat) images from cameras are amended with depth maps which bring additional information about the third dimension of a scene. Depth maps are matrices of values reflecting distances between the camera and points in the scene. Typically, depth maps are presented as gray-scale video, cf. Fig. 1.12, where the closer objects are marked in high intensity (light) and the far objects are marked in low intensity (dark).

The particular meaning of the term "depth" varies in literature and often is used interchangeably with others such as disparity, normalized disparity, z-value. In fact, in Fig. 1.12, the "depth" represented by the gray-scale value of the pixels corresponds to disparity: the closer pixels have a higher value, which is in line with a higher disparity value. In order to understand the exact meanings of each of these terms "depth" and "disparity," let us consider a pinhole model of a camera.

A *pinhole camera* is a simplified Camera obscura in which the lenses have been replaced with a single pinhole which acts as an infinitely small aperture. Light rays from the 3D scene pass through this single point and project an inverted image on the opposite side of the box, cf. Fig. 1.13A. In some formulations of the model of the pinhole camera, the fact that the projected image is inverted is often omitted by

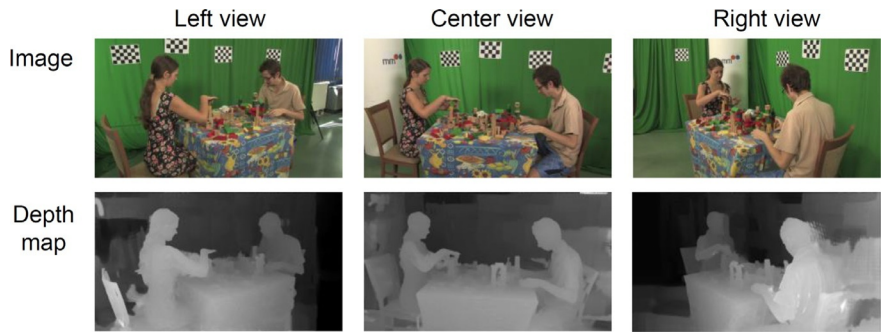

FIG. 1.12

The original view (A) and corresponding depth map (B) of a single frame of "Poznan Blocks" [104, 105] 3D video test sequence. In the depth map, the closer objects are marked in high intensity *(light)* and the far objects are marked in low intensity *(dark)*.

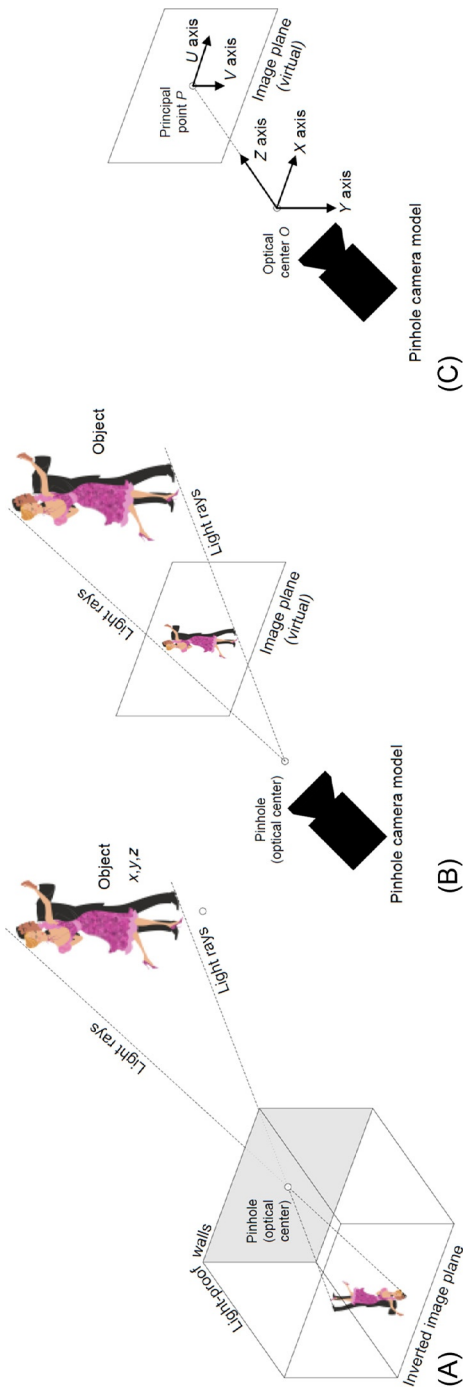

FIG. 1.13

A visualization of a pinhole camera (A), its simplified model (B), and the corresponding coordinate system (C).

assuming that the image plane (virtual) is placed on the same side of the pinhole as the scene, cf. Fig. 1.13B. Therefore the aperture equation is:

$$\frac{f}{z}=\frac{u}{x}=\frac{v}{y}. \tag{1.1}$$

where f is the focal length (distance of the image plane along the z axis), u and v are coordinates on the projected image plane of some point with x, y, z coordinates in 3D space. In the mentioned equation the projected coordinate system UV is shifted so that it is centered on the principal point P, cf. Fig. 1.13C. In such a case the perspective projection is as follows:

$$u=\frac{x}{z}\cdot f, \quad v=\frac{y}{z}\cdot f. \tag{1.2}$$

Using such a pinhole camera model, we can consider object F, cf. Fig. 1.14, observed from two cameras (left and right). In each of views, the given object is seen from a different angle and position, cf. Fig. 1.14, and therefore its observed positions on image planes are different (F_L in the left image and F_R in the right image). With only a single view (e.g., the left one) we cannot tell what is the distance of the object observed at position F_L. For the real position F, the object can be thought as being in position of G, I, or J, which in the right view would be observed in positions F_R,

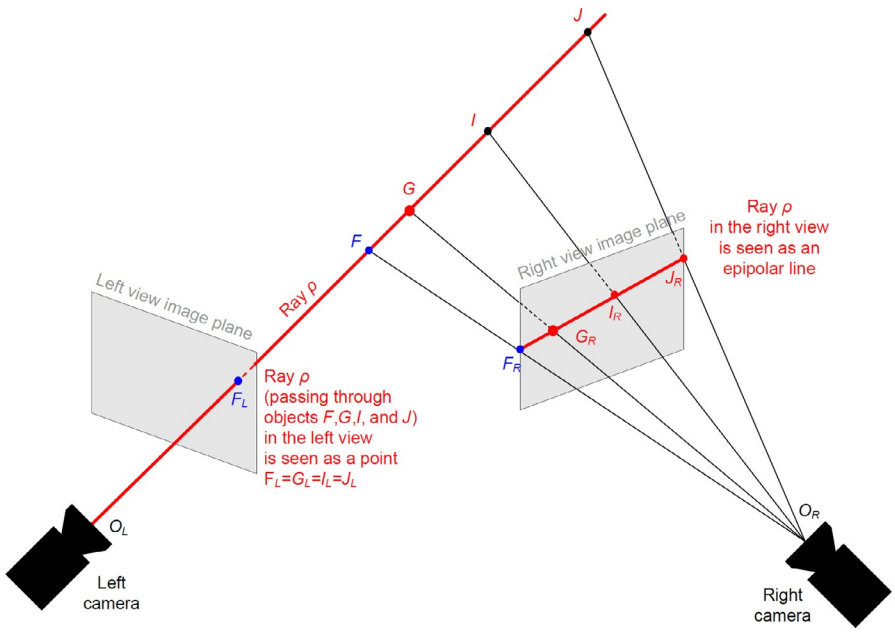

FIG. 1.14

Epipolar line as a projection to the right view of all potential position of object residing on ray ρ, seen in the left view as a single point.

G_R, I_R, and J_R, respectively. The set of all potential 3D positions of observed object E_L, projected onto the other image plane, is geometrically inclined to lie along a so-called epipolar line.

An epipolar line is a projection of a ray, pointing from an optical center of one camera to a 3D point, to the image plane of another view. For example, in Fig. 1.14, a ray ρ (starting in O_L and passing through F) is seen by the left camera as a point because it is directly in line with that camera's projection. However, the right camera sees this ray ρ in its image plane as a line. Such projected line in the view of the right camera is called an epipolar line. An epipolar line marks potential corresponding points in the right view for pixel F in the left view.

Positions and orientations of epipolar lines are indicated by the locations of the cameras; their orientations; and other parameters, such as focal length, angle of view, etc. Typically, all of those parameters are gathered in the form of intrinsic and extrinsic camera parameter matrices, cf. Section 1.3.1. In a general case, epipolar lines may lie along arbitrary angles, cf. Fig. 1.16 (top).

A common and important case for considering depth is the linear arrangement of the cameras with parallel axes of the viewpoints, cf. Fig. 1.15. Such setup can be obtained both by precise physical positioning of the cameras or by postprocessing, called rectification, cf. Fig. 1.16 (bottom).

In such a linear arranged case, the image planes of all views coincide, cf. Fig. 1.16 (bottom), and the epipolar lines are all aligned with the horizontal axes of the images. Therefore the differences in observed positions of objects become disparities along horizontal rows. Due to the projective nature of such video system with linear camera arrangements, the further the object is from the camera, the closer is its projection to the center of the image plane.

1.2.2 DEPTH IN STEREO AND MULTIVIEW VIDEO

Let us now define various terms related to the *depth*, using the camera geometry of Fig. 1.17.

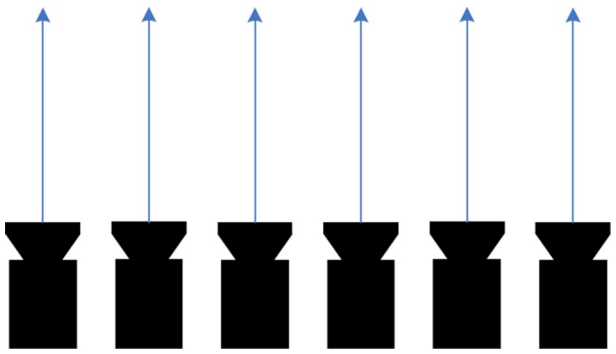

FIG. 1.15

Linear arrangement of the cameras with parallel axes of the viewpoints.

FIG. 1.16

Original (top) and rectified (bottom) images from "Poznan Blocks" [104, 105] multiview video sequence for some selected pair of views. Exemplary epipolar line for ray ρ has been marked with a solid line.

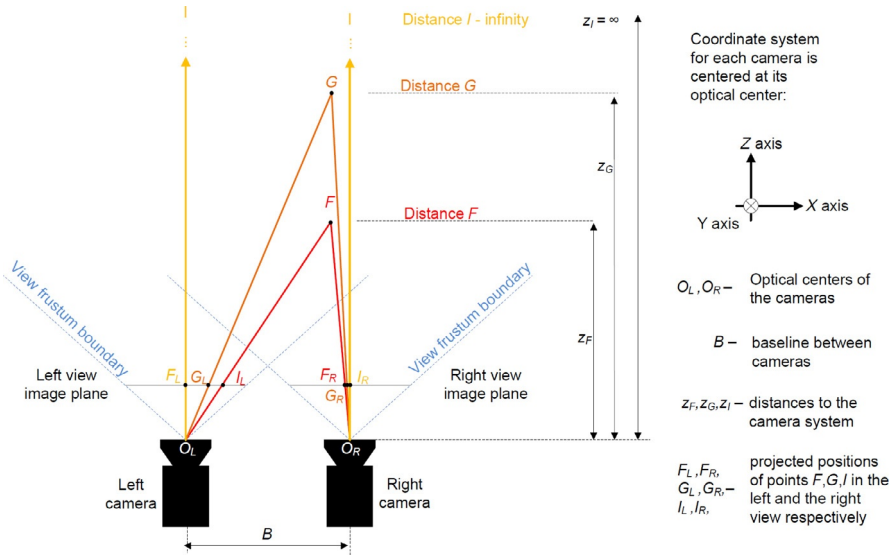

FIG. 1.17

Exemplary objects E, F, and G—projected onto image planes of two cameras.

For a given camera (e.g., the left one in Fig. 1.14) the distance between the optical center (O_L) and the object (e.g., represented by a single pixel) along the z axis is called the z-value. The unit of the z-value is the same as the distance units in the scene. These may be, e.g., meters or inches, and depends on the scale of the represented scene.

The distance between observed positions of object F, with F_L in the left image and F_R in the right image, is called *disparity*:

$$\Delta_E = F_L - F_R. \tag{1.3}$$

In a general case, the disparity is a vector lying along the respective epipolar lines. In case of linear rectified camera arrangements, disparity vectors lie along horizontal lines.

The z-value and the disparity between two views are mathematically related and can be derived if the positions and models of the cameras are known. For a stereoscopic pair of cameras, horizontally aligned (or rectified), distant by the length of baseline B, the z-value Z_F of point F can be calculated as follows:

$$Z_F = f \cdot B \cdot \frac{1}{\Delta_F}, \tag{1.4}$$

where Z_F is the distance along z-axis of point F from the camera's optical center, B is the baseline distance between the pair of cameras, and Δ_F is the disparity of point F.

Obviously, a similar but more complex relation between the z-value and the disparity can be found for other arrangements of camera pairs.

The term "depth," depending on the context, may refer to different meanings, e.g., it is often used interchangeably with disparity and with z-value distance, or sometimes as a generalized term, describing both.

Usually, in literature term "depth" is just used to refer to a data stored in depth maps, e.g., in the form of images or files on disk. For example, in OpenEXR Blender files [12] the depth is stored in the form of z-values represented as 32-bit floating point number (IEEE-754 [13]).

The most common format of depth value representation is an 8-bit integer representing, the so-called *normalized disparity*. The disparity is then normalized so that the represented range (0–255) covers the range of disparities from Δ_{min} to Δ_{max}, corresponding to z-values from z_{far} to z_{near}, respectively. Such a format combines the advantages of representing depth as a disparity or as a z-value. The representation of depth as disparity better suits the human visual system (foreground is represented at a finer quantization than the background), while disparity can be almost directly used for view synthesis and is actually a direct product of depth estimation algorithms. The representation of depth as disparity actually depends on the considered camera arrangement (e.g., disparity is proportional to baseline B).

The representation of depth as z-value does not have such disadvantage. The same goes for normalized disparity, because the disparity values are normalized with respect to the z_{far} to z_{near} scale.

The depth defined as normalized disparity can be calculated from the following equation:

$$d = d_{max} \cdot \left(\frac{\Delta - \Delta_{min}}{\Delta_{max} - \Delta_{min}} \right), \tag{1.5}$$

$$d = d_{max} \cdot \left(\frac{1}{Z} - \frac{1}{z_{far}} \right) \Big/ \left(\frac{1}{z_{near}} - \frac{1}{z_{far}} \right), \tag{1.6}$$

where d_{max} is the maximal value for a given integer binary representation, e.g., 255 for 8-bit integers, or 65,535 for 16-bit integers.

The selected depth range (e.g., Δ_{min} to Δ_{max}, z_{far} to z_{near}) along with the bit width of integers used together affect the *precision* with which the depth is represented. For a given bit width, e.g., 8-bit, the broader the selected depth range, the worse is its precision. On the other hand, enforcing sufficient precision narrows the depth range which at some point may disallow the representation of some objects in the scene. For sophisticated scenes, with nonlinear camera arrangements, it is often impossible to meet both requirements at the same time and usage of wider integer range (e.g., 16-bit) becomes mandatory.

Apart from the depth precision, which is an attribute of the representation format, another important factor is the *depth accuracy* which results from the method of depth acquisition. The depth can be acquired with the use of specialized equipment, like depth sensors, or by means of algorithmic estimation, as explained in Section 1.4.

In the case of depth sensors, the achievable depth accuracy results from the used technology. For example, for Microsoft Kinect [14], cf. Fig. 1.18 (top), the depth error is from 2 mm for close objects to 7 cm for far objects [14]. For the Mesa Imaging SR4000 [15] depth sensor, cf. Fig. 1.18 (bottom), the accuracy is about 15 mm. It must be noted, however, that the accuracy of depth sensors depends on factors like reflectivity of objects in the scene, distance of the scene, etc.

In the case of depth estimation algorithms, like the ones in Section 1.4, the accuracy is strongly influenced by the resolution of the analyzed images. The depth is estimated by finding image correspondences between the views, and thus the direct outcome of such is disparity. The measurement of distances in the images is limited by their resolutions and thus the higher the resolution, the more accurate the depth is [16]. Also it is crucial that the image frames from different cameras are taken at the same moment, and thus the cameras must be synchronized in time. The desired degree of synchronization is at the subframe level, e.g., per scan line. This is one of the factors influencing the choice of equipment for building a 3D camera system, which is a topic of the next section.

FIG. 1.18

Photography of Microsoft Kinect depth sensor (left) and Mesa Imaging SR4000 depth sensor (right).

1.2.3 MULTICAMERA SYSTEM

Considerations on multiview camera system features are highly defined by specifics of the current multiview technology, which is not mature nowadays. On one hand, there are no comprehensive industrial solutions yet, and on the other hand, the construction of an experimental multiview system is a very technically demanding task. Effectively, a universal system that would satisfy all needs currently cannot be reached. In order to develop a useful multiview camera system, many compromises have to be made. As it will be shown, some of the constraints are interconnected and cannot be considered separately. Therefore it is practical to start with some set of initial requirements and then refine them according to technical possibilities. The following issues in Fig. 1.19 should be considered, together with the practical setup choices of Table 1.1.

Number of cameras—perhaps the central parameter of the whole multiview camera system definition. We consider three or more cameras. Less than three cameras does not allow production of real multiview material, useful for SMV or depth estimation. Up to some extent, the more cameras we have the better, but typically there are three main limitations: budget, size, and throughput (of processing and transmission/storage). The first two are very loose, while the latter can be a serious problem.

Processing throughput—depending on the target usage (real time or off-line), some of the processing has to be done on the fly. Depending on our needs, we should anticipate computational power for stream synchronization, rectification, color

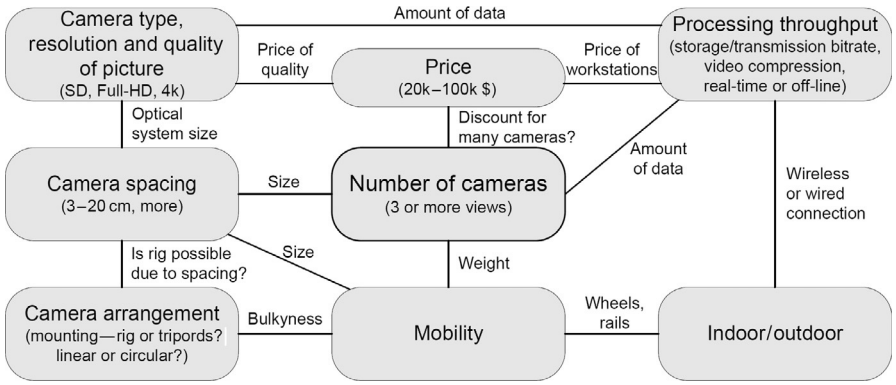

FIG. 1.19

Mutual relations among features and between parameters of a multiview camera system.

Table 1.1 Comparison of Camera Arrangements

	Linear Arrangement	**Angular Arrangement (Convergent)**	**Angular Arrangement (Divergent)**
Depth estimation	Simple, many cameras can be used. Disparity estimation by horizontal search only	Difficult, simple disparity estimation cannot be used	Difficult, simple disparity estimation cannot be used
System calibration (rectification)	Simpler	More difficult	More difficult
Camera setup (arrangement)	Relatively easy to set the camera in a line	More difficult to set the cameras in an arc	More difficult to set the cameras in an arc
Camera rig	Compact, can be mobile	Depends on the arc size; can be large and difficult to move	
Parallax/3D effect	Limited, especially in the case of distant objects	Great inside arc. Not good at looking away	Good for looking-around feature, but bad at close distances
Target display system	Stereoscopic display, auto-stereoscopic display	Stereoscopic, wide arc can be used for SMV displays. Data for SMV displays is more difficult to prepare. VR glasses	Panoramic displays and VR glasses

correction, and depth map estimation. Also we should estimate the required bitrate for our medium. For uncompressed raw bitstreams we can use the following equation:

$$B = N \cdot W \cdot H \cdot C \cdot D \cdot F, \tag{1.7}$$

where B is the required bitrate (in bits/s) of a medium—this may be transmission bandwidth of a connection or bitrate of a file; N is the number of cameras; W and H are, respectively, the width and height (in pixels) of the picture; C is the number of full-scale color components (3.0 for 4:4:4 format, 1.5 for 4:2:0 format); D is the bit-depth (8 for typical images); and F is the frame rate (in Hz). For example, if we consider uncompressed 8-bit 4:2:0 raw data stream from nine HD (1920×1080) resolution cameras, with frame rates of 25 frames/s, we get approximately a required bitrate of about 5.6 GBit/s.

Indoor/outdoor usage—the previous parameters can be greatly influenced by the available space to install the system. Outdoor systems cannot use long-range cable connections to any workstations, while wireless connections offer much lesser bitrate and some video compression might be required. The outdoor systems are typically desired to be mobile, which is very difficult to achieve if uncompressed video is acquired. In such systems, the maximum cable length is an important factor that influences mobility and capability to outdoor video acquisition.

Mobility—not only means the ability of the system to be easily rearranged in a different place, but also the ability to shot with moving scenes. Depending on the camera arrangement and mounting, some wheels, caterpillars, or rails may be required.

Camera arrangement—the most commonly used camera setups are linear and circular. The linear setup offers good visual sensations for distant scenes and is easier to handle from the software point of view (rectification and depth map algorithms are commonly investigated for linear setup). On the other hand, circular setups give better 3D depth sensation in case of short-range, indoor scenes. It also more resembles the human visual system. Depending on the mobility, the camera number and spacing, the use of a rig should be considered. Although a rig can be bulky and heavy, it offers much better stiffness and rigidness than the use of separate camera tripods.

Camera spacing—this also depends on the type of target scenes. Short-range scenes require short base intercamera distance. This distance often resembles the human interocular distance (about 6.5 cm) as much as possible. Shooting distant scenes requires longer intercamera distances, so that differences between views acquired by cameras is big enough.

Camera type—probably the most sensitive feature of the system. The cameras must be selected with special care. Implicitly, selection of the camera type will impact on all parameters of the whole system. The size of a camera optical system influences the minimal camera spacing and thus the camera arrangement and mounting possibilities. Resolution of the camera (SD, HD, or even 4K) influences the required processing throughput which can enforce off-line processing

with the use of intermediate video compression for transmission or can allow real-time processing of uncompressed data streams. Also, depth estimation requires exact temporal synchronization which can be supported by the given camera or not. Tools for reaching temporal synchronization include synchronization input/output ports (genlock), time-stamp signature [17], and control interfaces like LANC [18].

Another factor is the type of the shutter. In order to minimize the influence of motion in the scene (which is observed in parallel with all of the cameras) in the construction of 3D model of the scene, the preferred shutter type is the global one. Unfortunately, many cameras, especially cheap ones, provide rolling shutters which makes shooting fast moving scenes more difficult. In the end, prices of cameras vary very much, which depending on the available budget influences the number of cameras that can be afforded.

Table 1.2 summarizes the most important properties of different types of cameras with respect to available features and interfaces. The features listed in the table are the most popular ones.

1.2.4 ACQUISITION SYSTEM EXAMPLES

As mentioned in previous section, it is impossible to build a universal system which would satisfy all possible needs. Nonetheless, we can consider some of successfully deployed systems that were built experimentally and are currently used for studies on the development of multiview technology, cf. Table 1.3. Of course the known multicamera systems are not limited to those examples.

1.2.4.1 Nagoya University multiview camera system

The system [19] has been built at Nagoya University, Japan, cf. Fig. 1.20. It is composed of 80 cameras which are fixed on a steel frame with 50-mm interval. Depending on the setup the cameras can be placed linearly or on an arc, which converges at about 8 m from the camera array. The cameras that are used are JAI Pulnix TMC-1400CL. They enable acquisition of SXGA images with a frame rate of almost 30 FPS. The shooting is temporally synchronized up to about 1-μs precision. The system also consists of one host-server PC and 100 client PCs called nodes. The interface between camera and PC is CameraLink.

1.2.4.2 Fraunhofer HHI camera system

The system [20] has been developed at the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, HHI, Germany, cf. Fig. 1.21. It consists of 16 machine vision cameras arranged linearly on a rig. The cameras are Hitachi 3-chip CCD progressive scan RGB cameras (HVF31CL-S1) with a XGA resolution. These are equipped with high quality 6-mm lenses (Fujinon DF6HA-1B) with horizontal Field of View (FoV) of 44 degree and vertical FoV of 33 degree. The trigger period is 20 ms, which results in a frame rate of 16.67 FPS. The data is captured in raw RGB format using the CameraLink interface via a raid-like PC cluster.

Table 1.2 Comparison of General Characteristics of the Most Common Classes of Cameras

		Industrial Cameras	Surveillance (Megapixel) Cameras	Consumer (Handy Cams)	Professional Cameras	Cinematic Cameras	
Video capture		Resolution	Up to several megapixels	Up to several megapixels	SD/HD	SD/HD/2K	2K/4K/8K and more
		Frame rate (per second)	10–1000 wide range (trade-off with resolution)	5–60 wide range (trade-off with resolution)	25–60	24–60 and even to 120 in slow motion mode	24–60 and even to 120 in slow motion mode
		Shutter	Rolling/global	Typically rolling	Typically rolling	Global or tiled	Global or tiled
		Dynamic range	Low	Low but often with night-mode	Medium	High	High
Video interface		CameraLink	+	–	–	–	–
		SD\HD-SDI	–	–	–	+	+
		Gigabit Ethernet	+	+	–	–	–
		USB	+	+	+	–	–
		FireWire	+	+	+	+	+
Control interface		Ethernet	+	+	–	–	–
		FireWire	+	+	+	+	–
		LANC	–	–	+	+	–
		Tri-Leve Sync	–	–	–	+	–
		Time Code	–	–	–	+	+
		Dedicated	+	+	–	–	–
Market and technology		Accuracy/ repeatability	Satisfactory	Low	Satisfactory	Satisfactory	Satisfactory
		Price	Medium	Low to medium	Low	Medium	High

Table 1.3 A Survey of Some Experimental Multiview Systems

	Nagoya University	Fraunhofer HHI	Poznan University of Technology Linear Rig	Poznan University of Technology Modular System	Hasselt University Linear Rig	Hasselt University Modular Curvilinear System
Year of production	2005–07	2008	2009	2013	2012	2012
Number of cameras	100	16	9	10	8	16
Camera spacing	50 mm	65 mm	137.5 mm	Mobile modules, minimally about 50 cm	About 1 m	About 10 m
Camera type	JAI Pulnix TMC-1400CL	Hitachi 3-chip CCD progressive scan RGB camera HV-F31CL-S1	Canon XH-G1 CCD	Canon XH-G1 CCD	Basler avA1600-50gc	Prosilica GC
Resolution	1392×1040	1024×748	1920×1080	1920×1080	1600×1200	1920×1080
Frame rate	29.4114 FPS	16.67 FPS	24 FPS	24 FPS	60 FPS	25 FPS
Interface	CameraLink	CameraLink	SDI	SDI	Gigabit	Gigabit
Recording	PC computers	Raid-like PC cluster	Raid-like PC cluster	Internal	PC Cluster	PC Cluster
Processing	PC computers	Raid-like PC cluster	Raid-like PC cluster	Raid-like PC cluster	PC Cluster	PC Cluster
Mobility	Immobile	Immobile	Mobile rig	Mobile individual modules	Mobile individual modules	Mobile individual modules
Arrangement	Arc/linear	Linear	Linear	Any	Linear	Any

FIG. 1.20

Nagoya University camera system [19].

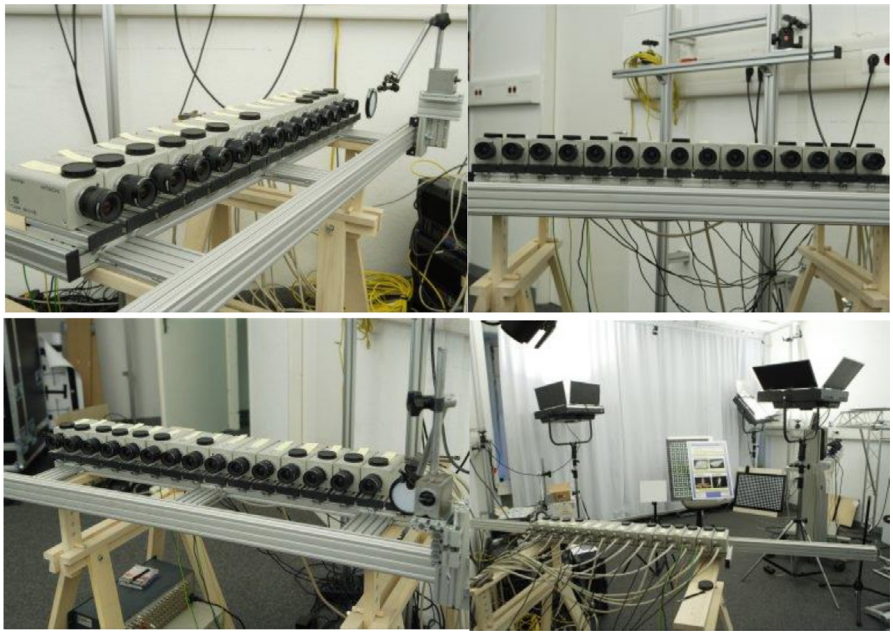

FIG. 1.21

Fraunhofer HHI Camera system [20,21].

1.2.4.3 Poznań University of Technology multiview camera system (linear rig)

The system has been built at the department of Multimedia Electronics and Telecommunications, Poznań University of Technology, Poland, as an experimental framework for studies on future 3D television [22,23]. The system of Fig. 1.22 consists of nine cinematic Canon XH-G1 cameras placed on a mobile (wheeled) metal rig. The rig has been manufactured exclusively to provide special mounting pads that allow precise alignment of the cameras. The output video signal is HDTV (1920×1080) and is provided via SDI interface. All streams are temporally synchronized with the use of a genlock and captured by a raid-like PC cluster. The whole processing is done offline.

1.2.4.4 Poznań University of Technology multiview camera system (modular)

This is the second multicamera system built at Poznań University of Technology. It employs the same Canon XH G1 cameras, but this time they are mounted on special wireless mobile units, cf. Fig. 1.23. This allows for placing the cameras anywhere in the scene. There are 10 of such mobile camera units, cf. Fig. 1.24. Each of them is also equipped (apart from the camera) with a power supply (battery), a wireless synchronization receiver, a remote control receiver, and a HDD recorder. Thanks to that, each camera module is able to record about 30 minutes of high resolution video. All cameras are precisely synchronized with the use of a wireless dedicated 869-MHz link. Each captured frame is signed with a time code for error resilience. This also allows for the detection of miss-synchronization. All cameras can be controlled by a dedicated system that also uses a separate WiFi wireless link.

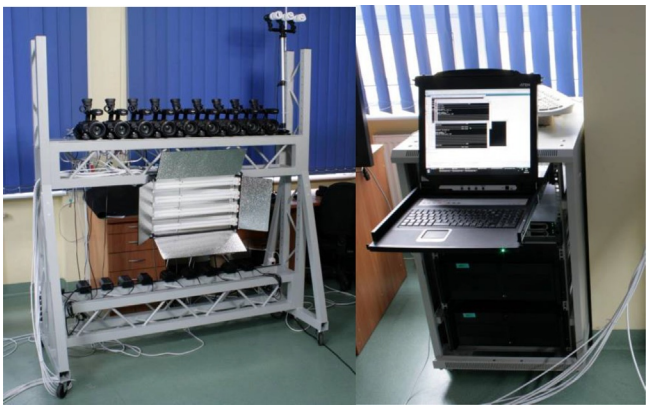

FIG. 1.22

Experimental multiview camera system built at Poznań University of Technology (left—camera rig, right—part of recording system).

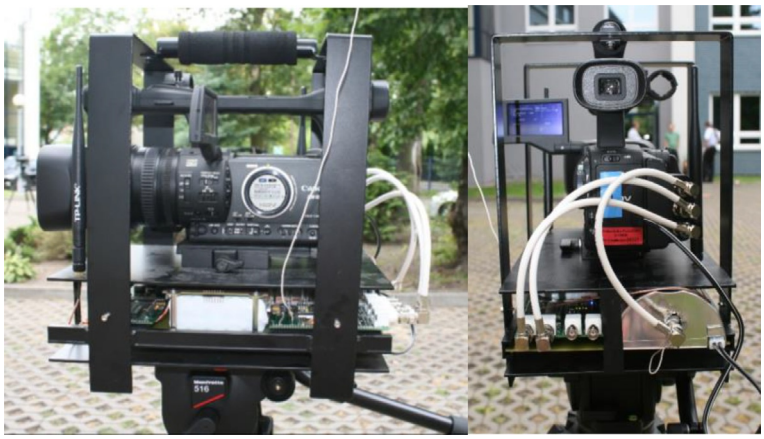

FIG. 1.23

Wireless mobile camera module (side and rear views).

FIG. 1.24

Multicamera setup on an exemplary arc.

1.2.4.5 Hasselt University multiview camera system

Hasselt University has provided soccer test sequences to the MPEG community, one with a linear camera arrangement and a second one with an arc arrangement, cf. Fig. 1.25.

For the linear setup, eight Basler avA1600-50gc cameras with a resolution of 1600×1200 were placed on a line approximately 1 m apart from each other. The first half of the cameras uses 25-mm lenses, the second half 12:5-mm lenses.

FIG. 1.25

Linear (top) and curvilinear (bottom) multicamera system of Hasselt University.

The recordings were made at 60 Hz with perfect synchronization. The data was transferred through a 10-Gigabit fiber Ethernet switch to a centralized computer.

The curvilinear data set was captured with 16 Prosilica GC cameras, 10 m apart from each other. Sixteen millimeter lenses were used, and the recordings were done at full-HD 1920×1080, 25 fps.

1.3 MULTIVIEW VIDEO PREPROCESSING

1.3.1 GEOMETRICAL PARAMETERS

The previous section presented the pinhole camera model. Such model explains the nature of the projection of light rays onto the image plane and gives some understanding on various terms related to depth. The described pinhole camera model was defined with the use of x, y, z coordinates in 3D space and u, v coordinates in the image plane. In this section we will consider a system of multiple pinhole cameras, each having individual coordinate systems placed, respectively, to the camera position, cf. Fig. 1.26. Placement of these coordinate systems for each camera will be defined by camera parameters: intrinsic, extrinsic, and radial distortions parameters.

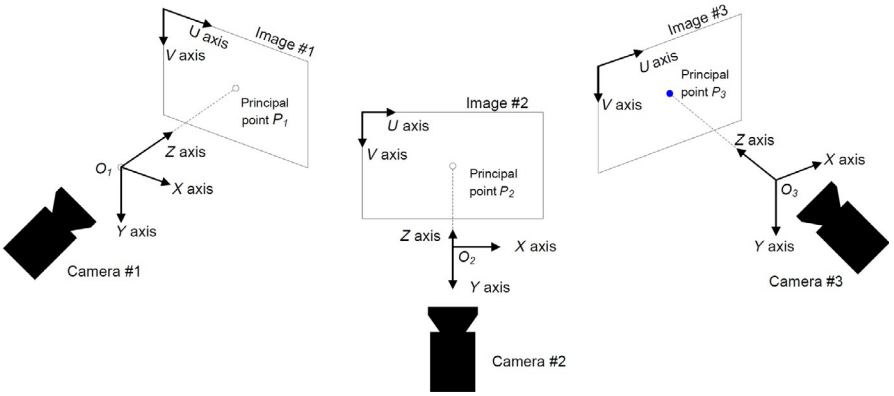

FIG. 1.26

A system of multiple pinhole cameras, each having individual coordinate systems: x, y, z in 3D space and u, v in the image plane.

1.3.1.1 Intrinsic parameters

We will use the described model with some slight modifications compared to Fig. 1.26: the u, v coordinates have been shifted and scaled so that they match units of pixels in the given image and are centered at the upper-left corner pixel (instead of on principal point P). For example, for Full-HD 1920x1080 resolution u, v coordinates (0,0) address the left-top-most pixel. The next pixel to the right has coordinates (1,0) and the bottom-right corner pixel has coordinates (1919,1079). The modified projection equations (1.2) are then modified to:

$$u = \frac{x}{z} \cdot f \cdot u_{scale} + P_u, \quad v = \frac{y}{z} \cdot f \cdot v_{scale} + P_v, \tag{1.8}$$

where (P_u, P_v) is the position of the principal point P in the u, v image plane space and is thus expressed in units of pixels. The terms $f \cdot u_{scale}$ and $f \cdot v_{scale}$ are often denoted as f_u and f_v, respectively, and they express the focal length in units of pixel size (horizontal or vertical). Because the pixel aspect-radio can be nonrectangular, f_u can be different than f_v.

Eq. (1.2) can be rewritten with the use of homogenous coordinates [24]. In contrast to Cartesian coordinates which define a point, homogenous coordinates define a line, which is parameterized with respect to the last coordinate, which customarily is denoted as 1, because it can be pulled out in front of the vector. For example, the following homogenous coordinates express the same line in 3D space:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = 2 \cdot \begin{bmatrix} x/2 \\ y/2 \\ z/2 \end{bmatrix} = z \cdot \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \tag{1.9}$$

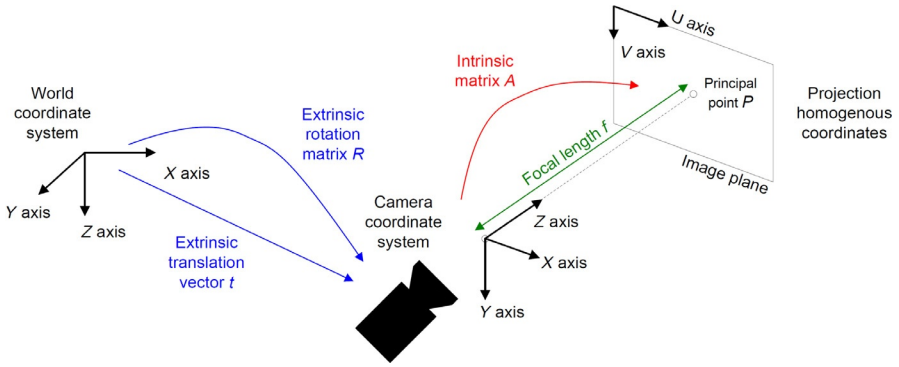

FIG. 1.27

The role of intrinsic and extrinsic matrices in transformation of coordinate systems.

As it can be seen, homogenous coordinates are useful for describing perspective projection along the last coordinate (e.g., z). Thus Eq. (1.2) can be rewritten as:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \text{where} \quad A = \begin{bmatrix} f_u & 0 & P_u \\ 0 & f_v & P_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.10}$$

where s is a scaling factor chosen to make the third output coordinate to be one (here $s = z$).

Matrix A is called the *intrinsic matrix*. Its role is to transform points in the coordinate system of the given camera to the projection homogenous coordinates, cf. Fig. 1.27. Thus the name "intrinsic" comes from fact that it describes transformation of coordinates inside of the camera (internal).

1.3.1.2 Extrinsic parameters

The second set of camera parameters is called *extrinsic*. Its role is to transform points in the global world coordinate system to the coordinate system of given camera. Therefore the name "extrinsic" comes from fact that it describes a transformation of coordinates outside of the camera (external). Extrinsic parameters typically consist of rotation and translation, as shown in Fig. 1.27.

Rotation in 3D space can be described with an orthonormal 3×3 matrix, here denoted as R:

$$c = R \cdot w, \tag{1.11}$$

where c is a vector of coordinates in camera coordinate system $c = [x_c\, y_c\, z_c]^T$ and w is a vector in the world coordinate system $w = [x_w\, y_w\, z_w]^T$. For example, R may be a rotation around the z-axis with angle α:

$$R = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.12}$$

Obviously, translation of the camera can be described with a vector t in the world coordinate system:

$$c = w + t, \quad \text{where} \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{1.13}$$

Often, the translation of the camera is described by means of the inverse translation vector $t' = -R \cdot t$. Although vectors t and t' both describe position of the camera—and they are often confused in the literature—the difference between them is crucial. Vector t describes the position of the camera in the world coordinate system. Vector t' describes the position of the origin of the world coordinate system in the given camera coordinate system.

The extrinsic set of parameters, i.e., rotation and translation, can be expressed as a single transform matrix, called *extrinsic matrix* E, again with the use of homogenous coordinates:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = E \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \quad \text{where} \quad E = [R|t']. \tag{1.14}$$

Such equation can be chained with the intrinsic transformation in order to obtain a combined equation, allowing the calculation of the image plane coordinates u, v from coordinates w in world space:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot E \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \tag{1.15}$$

where s is a scaling factor chosen to make the third output coordinate equal to one.

1.3.1.3 Lens distortion

Lens distortion is a deviation from the ideal projection considered in pinhole camera model. It is a form of optical aberration in which straight lines in the scene do not remain straight in an image. Examples of lens distortions are barrel distortion and pincushion distortion, cf. Fig. 1.28.

Most of the lens distortions can be corrected with the use of Brown-Conrady model [25]. The model is defined in the form of a transformation of a point from undistorted coordinates to distorted coordinates:

$$\begin{aligned} u_d &= u_n\left(1 + K_1 r^2 + K_2 r^4 + \cdots\right) + \left(P_2\left(r^2 + 2u_n^2\right) + 2P_1 u_n v_n\right) \cdot \left(1 + P_3 r^2 + P_4 r^4 + \cdots\right), \\ v_d &= v_n\left(1 + K_1 r^2 + K_2 r^4 + \cdots\right) + \left(P_1\left(r^2 + 2v_n^2\right) + 2P_2 u_n v_n\right) \cdot \left(1 + P_3 r^2 + P_4 r^4 + \cdots\right), \end{aligned} \tag{1.16}$$

where $r = \sqrt{(u_n - u_c)^2 + (v_n - v_c)^2}$, u_d, v_d are coordinates in the distorted image, u_n, v_n are coordinates in the undistorted image, u_c, v_c are coordinates of the distortion

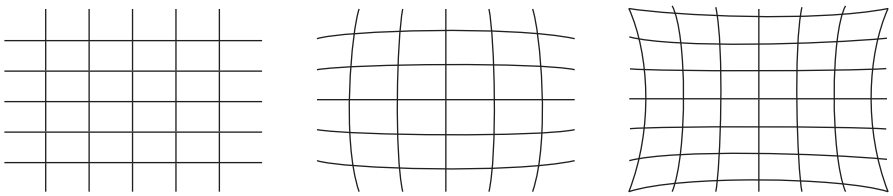

FIG. 1.28

Lens distortion shown on example of rectangular grid (left): barrel distortion (center) and pincushion distortion (right).

center (e.g., the principal point P_u, P_v), K_n are the radial distortion coefficients, and P_n are the tangential distortion coefficients.

Very often, only radial distortions of second order are considered. In such a case:

$$\begin{aligned} u_d &= u_n\left(1 + K_1 r^2 + K_2 r^4\right), \\ v_d &= v_n\left(1 + K_1 r^2 + K_2 r^4\right), \end{aligned} \tag{1.17}$$

Barrel distortion is typically modeled with negative K_1 value, whereas pincushion distortion has positive K_1 value.

1.3.1.4 Estimation of camera parameters

Practically any processing of data coming from a multicamera system, e.g., depth estimation, view synthesis, etc., requires knowledge about camera parameters described in this section. Therefore it is crucial to calibrate the system before use.

The most common way of calibration of geometrical parameters is the use of calibration patterns, like checkerboards, cf. Fig. 1.29. The calibration algorithms

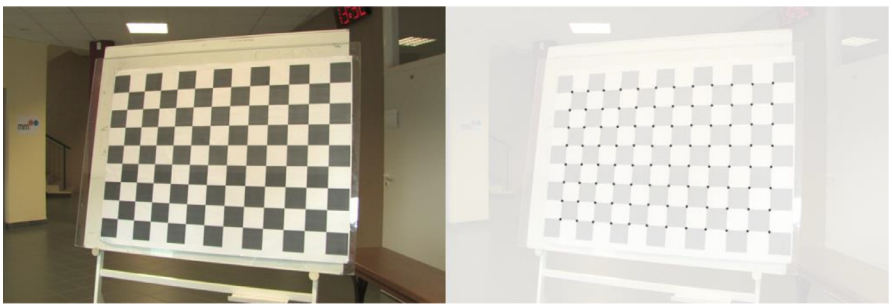

FIG. 1.29

Checkerboard pattern (left) used in calibration of camera system for "Poznan Hall" sequence [23] with marked corners which are used as features for calibration (right).

assume that the shape of the pattern in the real world is known, e.g., a checkerboard composed of 13×9 black/white boxes, each of size $10\,\text{cm} \times 10\,\text{cm}$. The location of the pattern in the image is found by feature search, e.g., edge/-corner detection. Good examples of calibration algorithms are described in [26,27].

Typically, calibration of some set of parameters (e.g., intrinsic matrix) requires that more primal parameters are known. Therefore it is best to perform calibration from lens distortion, then intrinsic parameters and finally extrinsic parameters.

Estimation of *lens distortion parameters* consists in solving a system of nonlinear equations (Eq. 1.16) for a set of points describing the calibration pattern. For the mentioned example we would assume that u_n, v_n are lying on a perfectly rectangular grid (although seen from a slight angle). The observed u_d, v_d allow for estimation of K_n and P_n lens distortion parameters. In order to achieve good results, the calibration checkerboard should be entirely visible and should cover most of the view of the camera being calibrated.

As soon as the lens distortion is calibrated (and preferably corrected) it is possible to calibrate the *intrinsic camera parameters*. The process is very similar to the previous one. We try to find intrinsic parameters that minimize the error of Eq. (1.10) for a set of features describing the calibration pattern. In our example of the checkerboard, we know that the points lie on a plane, on a perfectly rectangular grid (we assume that lens distortions have been already removed). We know what are the observed u, v coordinates of each feature and what are the assumed x, y, z coordinates. From proportions in our perspective projection, we can tell what are the coordinates P_u, P_v of the principal point in the image plane and what are the values of f_u, f_v expressing the focal length in units of pixel size (horizontal or vertical). It is worth to notice that is impossible to measure focal length f in world units (e.g., centimeters) without additional knowledge about positioning of the pattern (e.g., its z-value).

The final step, calibration of the *extrinsic parameters of the cameras*, requires a slightly different arrangement of the calibration pattern than before. Because we want to find relative positioning of the cameras in the scene, the calibration pattern should be visible in all of the views simultaneously. Of course such situation is not always possible in practical situations. One solution is to capture the checkerboard at multiple positions, so that it is visible in at least a few views at the same time, cf. Fig. 1.30A. Another solution is the use of a lighting diode moving in the fields of view of all cameras, cf. Fig. 1.30B.

Regardless of the exact method of finding features which are common between the cameras, the calibration of the extrinsic parameters employs them to construct a system of equations like Eq. (1.15). It is assumed that the given feature, although observed in different projected position u, v, resulting from different camera-space coordinates x_c, y_c, z_c (both individual for each camera) resides in a particular point in the world coordinate system x_w, y_w, z_w, common for all cameras. It is interesting to notice that even when a vast number of features are used, the equation system will

FIG. 1.30

Extrinsic camera parameters calibration: (A) checkerboard pattern placed in various places and angles in the scene in order to maximize visibility in the cameras, and (B) lighting diode moving in the fields of view of all cameras.

not become overdetermined. In fact, it is underdetermined and additional assumptions have to be made. The most commonly used ones are as follows:

- the global world coordinate system is identical with the local coordinate system of one of the cameras
- the extrinsic parameters matrix E is composed of the translation vector t' and proper rotation matrix R (Eq. 1.14) which is orthonormal
- the units in the world coordinate systems are scaled with respect to the real world units, e.g., meters.

1.3.1.5 Camera parameters file format

An example of format for storage of camera parameters is used within the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) MPEG Depth Estimation Reference Software (DERS) and View Synthesis Reference Software (VSRS) [28]. The format is ASCII based and is composed of multiple sections (one per camera) based on the following template (cf. Fig. 1.31). The dot sign (.) is used as a decimal point separator. "camera_name" can be any Latin characters string without spaces and end-of-line characters. The consecutive fields should be separated with spaces of tabulations. It can be noted that in most of the software packages, values of radial distortion coefficients are ignored.

1.3.2 VIDEO CORRECTION

Multiview video processing algorithms, like depth estimation or view synthesis, are very sensitive to imperfections of the acquisition path. For example, algorithms assume that objects have exactly the same color in all of the analyzed views or that there are no lens distortions.

camera_name			
$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	
$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	
K_1			
K_2			
$R_{1,1}$	$R_{1,2}$	$R_{1,3}$	t_x
$R_{2,1}$	$R_{2,2}$	$R_{2,3}$	t_y
$R_{3,1}$	$R_{3,2}$	$R_{3,3}$	t_z

FIG. 1.31

Camera parameter file format used by MPEG [28].

Therefore for the sake of simplicity of such processing algorithms, it is beneficial to preprocess the sequences in order to impose the required corrections, e.g., color correction or lens distortion correction.

1.3.2.1 Color correction

In multiview systems composed of high-class cameras, the mismatch between colors observed in different views can be invisible with a nonexpert eye. However, even in such optimistic case, there might be some slight inconsistencies in color profiles of the cameras. Therefore it is recommended to perform color calibration of the cameras, especially since the problem grows with the number of cameras.

The camera calibration needs a known calibration reference to be captured and the resulting output from the camera to be converted to color values. A correction profile can then be estimated (like ICC [29]) using the difference between the camera resulting values and the known reference values. The easiest way to perform this is to employ color calibration patterns, cf. Fig. 1.32, which are the counterparts of those used in geometrical calibration.

An alternative technique which can be used is to calibrate two or more cameras relatively to each other, to reproduce the same color values; not some reference color. In such a case [30], the target color profile may be, e.g., the average of the color profiles of all cameras.

The most common technique for correcting the colors is to usage Look-Up Tables (LUTs) which for given uncorrected color components (e.g., R_n, G_n, B_n in RGB color space) returns corrected color components (R_c, G_c, B_c). Due to implementation limitations, such lookup is typically performed independently for each component, e.g.:

$$R_c = LUT_R[R_n], \quad G_c = LUT_G[G_n], \quad B_c = LUT_B[B_n]. \tag{1.18}$$

Finally, it has to be noted that color correction does not solve all problems related to color mismatches between the views. For example, as a consequence of non-Lambertian reflections, surfaces seen from different angles exhibit different colors.

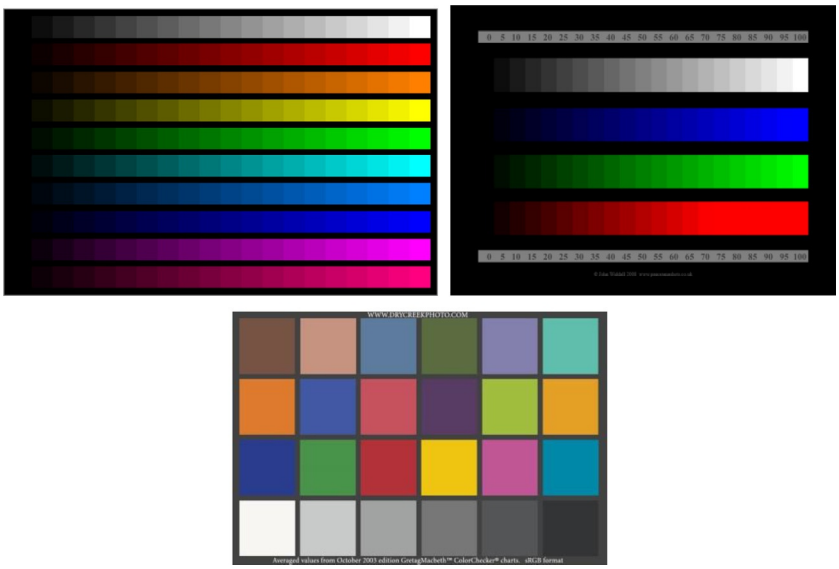

FIG. 1.32

Exemplary color calibration patterns.

Such issues have to be coped inside the respective algorithms, e.g., the depth estimation itself.

1.3.2.2 Lens distortion removal

Techniques that remove lens distortion consist in imitating a perfect pinhole camera so that further processing algorithms can be unaware of the actual acquisition imperfections. For such model, the distortions have to be known, as presented in Section 1.3.1.3.

In the simplest technique for lens distortion removal, cf. Fig. 1.33 (left), the model of the distortion is used directly for all pixels in the output undistorted image. For each pixel with coordinates u_n, v_n in the output undistorted image, the coordinates u_d, v_d in the input distorted image are calculated. A pixel targeted by those coordinates is copied to the output image. For the Brown-Conrady model Eq. (1.16) or (1.17) can be used. Of course, if the process has to be performed in real time for several frames with the same distortion model, u_d, v_d values can be stored in LUTs so that:

$$u_d = LUT_u[u_n, v_n], \quad v_d = LUT_v[u_n, v_n]. \tag{1.19}$$

Another solution for lens distortion removal in real time is to use a texture-mapping functionality supported by modern GPUs (graphics processing units). As shown in Fig. 1.33 (right), instead of processing single pixels, we use a triangle mesh covering the whole undistorted image. Content of each triangle is filled by means of texture mapping from the distorted image.

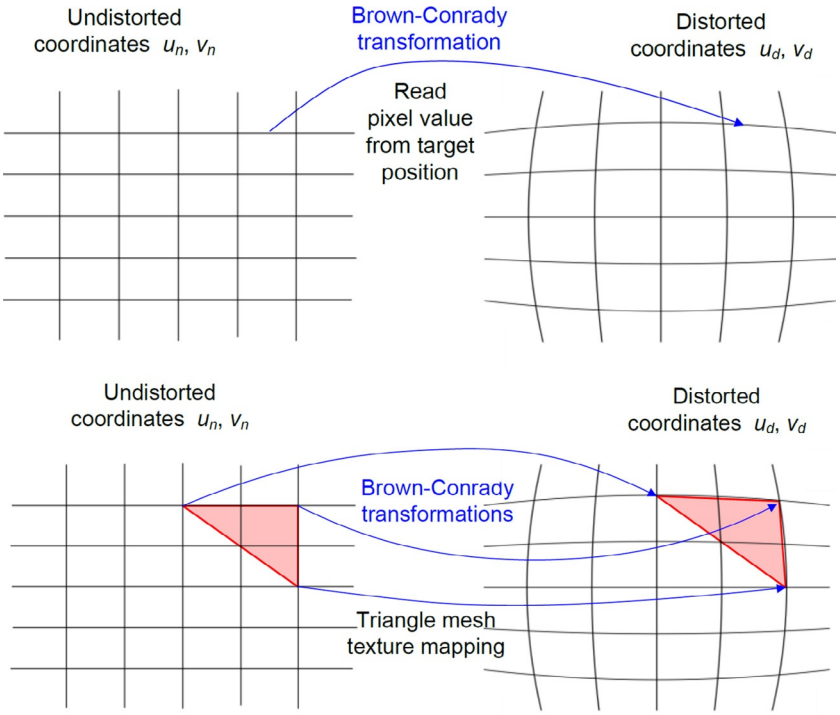

FIG. 1.33

Lens distortion removal by direct pixel copy (left) and by triangle-mesh-based texture mapping (right).

1.4 DEPTH ESTIMATION

As already explained in Section 1.1, depth plays an important role in synthesizing virtual views that do not correspond to existing camera views. Section 1.2 already briefly touched the subject of active versus passive depth estimation: the former uses active illumination techniques and responses thereof to estimate depth with Time-of-Flight sensors, while the latter is solely based on matching images acquired from multiple viewpoints to estimate depth. This is referred as stereo matching over a pair of input images. Of course, also hybrid solutions between active depth sensing and stereo matching are described in literature.

This section will give an overview of the passive depth estimation techniques that are related to the disparity phenomenon explained in Section 1.1: the more the 3D points in space are close to the cameras, the more disparity is observed over the camera views, or said differently, the more the 2D projection of that 3D point in space will move when switching from one camera view to the next.

The following sections will survey a couple of depth estimation techniques, ranging from local stereo matching where only the local neighborhood of each pixel is matched to the companion input image, to global methods involving all pixels of the input images at once for estimating the depth map. The number of input camera views is also an important parameter: some depth estimation techniques work well with only a pair of input images (stereo matching); others require at least a dozens of input images for depth estimation, which is the price to pay to provide very reliable depth maps.

Since depth estimation has been an active domain of research for many decades, a high abundance of papers exists in the field. A good starting point is the KITTI [31] and the Middlebury data set [32] providing test material, as well as indicative comparisons of the most prominent stereo matching algorithms against a small set of objective criteria. Recent comparisons between a multitude of stereo matching algorithms can be found in Refs. [33,34].

To simplify our discussion, we start from a set of perfectly parallel camera views, where the cameras have rigorously the same parameters (focal length, center of the optical axis, etc.). Referring to Section 1.3, the epipolar lines are then strictly horizontal. In practice, nonperfectly parallel input camera views can always be rectified in order to correctly obtain these horizontal lines. Each feature in an image is then guaranteed to lie on the same horizontal line in the other rectified input image, simplifying the stereo matching implementation.

We will often follow this rectification convention in the remainder of the section, largely simplifying our discussion. Of course, it is always possible to bypass the rectification step and directly do the stereo matching over the nonhorizontal epipolar lines, as shown in Fig. 1.16 (top).

1.4.1 LOCAL STEREO MATCHING

Local stereo matching will compare the surroundings of a pixel p in the left image to slightly translated positions q in the right image to estimate the disparity of pixel p. Each pixel is processed separately, without taking the full image context into account, which often results in noisy disparity images, especially in nontextured image regions influenced by any source of input noise (e.g., light flickering, slightly varying colors over adjacent camera views, etc.).

Ref. [35] provides a detailed overview of the different steps of local stereo matching in their paper's first figure. In view of giving the reader a bird's eye view over the different methods, we will restrict ourselves to the core disparity estimation kernel, which is followed by many refinements kept out of the scope of the current section.

To estimate the disparity in local stereo matching methods, the surroundings of a pixel p in the left image are compared to the surroundings of the pixel q in the right image, where q has been translated over a candidate disparity δ_p compared to p, cf. Fig. 1.34. For each pixel p, N candidate disparities $(\delta_p^1, \delta_p^2, \ldots, \delta_p^N)$ are tested ($N = 256$ and 65,536 for 8 and 16-bit depth maps, respectively), and the candidate

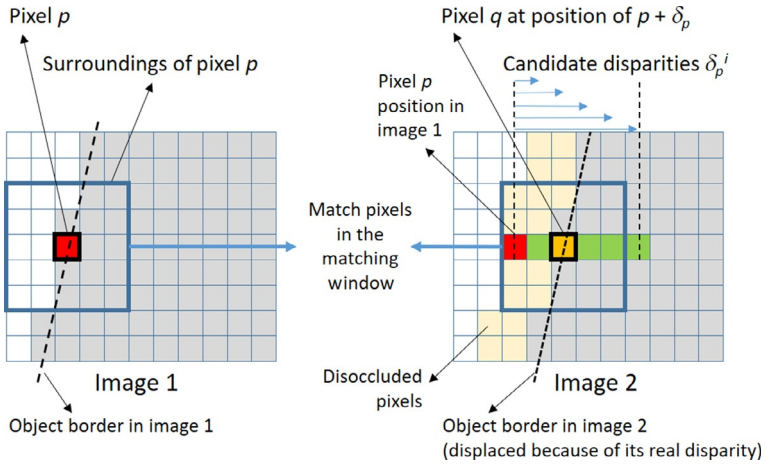

FIG. 1.34

Matching windows in stereo matching.

disparity resulting in the lowest matching cost is assigned to pixel p. Refs. [36,37] give a short overview of the most used matching costs.

Besides choosing a good matching cost metric, it is also important to well define the shape of the surrounding matching window around pixel p. Indeed, it is always implicitly assumed that all pixels within the window have more or less the same disparity δ_p^i, so that only one disparity can be reliably assigned to pixel p. As a counterexample, if the matching window lies half over one object and covers for the other half another object lying at a very different depth—e.g., the rectangular window of the object silhouette in Fig. 1.34—pixels of different disparities and/or partially disoccluded pixels (pixels not visible in the other image) will be matched together, yielding a best matching cost at a possibly wrong disparity value, very different from the real disparity that should be assigned to pixel p.

Reliable, local stereo matching approaches therefore take good care of the matching window shape, coinciding its borders with object borders. Ref. [38] includes gradient calculations to respect border objects, and Refs. [35,39] describe a method where, starting from pixel p and moving outward, the window shape is fixed at a position where neighboring pixel color differences start to be too large. This creates matching windows that never cross an object border, cf. Fig. 1.35, hence providing more reliable depth estimations.

The papers [35,38] describe these local stereo matching approaches in a tutorial step-by-step style, very useful to starters. Ref. [33] provides many visual results of different stereo matching techniques (including global ones, cf. next section) on the Middlebury, KITTI, and MPEG-FTV data sets.

FIG. 1.35

Border-aware window shapes (the cross-shaded regions) for stereo matching (each middle dot is the pixel for which the depth is estimated) [33].

1.4.2 GLOBAL STEREO MATCHING

In contrast to the local methods described in previous section, global stereo matching methods will assign disparities to all pixels in a coherent way over the full image, trading off competing local costs. These competing costs are as follows:

- the disparity matching cost similar to the ones in the previous section,
- a pixel coherency cost between neighboring pixels preserving local smoothness in the disparity values assigned to them,
- An occlusion/disocclusion cost to cope with pixels that are visible in one view, but not in another.

The latter two costs propagate from one pixel to the next throughout the full image, therefore taking a globally optimal decision over the image.

Two global methods are presented in the next sections. They offer very good performances and are often top ranked in the Middlebury and KITTI tests.

1.4.2.1 Graph Cut

As the name reveals, the Graph Cut technique [40,41] is associated to creating a graph that will be cut in N subgraphs, one for each candidate disparity δ_p^i. All pixels in a single subgraph will be associated the same disparity label, cf. Fig. 1.36.

To simplify the discussion, let us first take the example of the two fingers experiment of Section 1.1, where all pixels of each finger are associated to one disparity

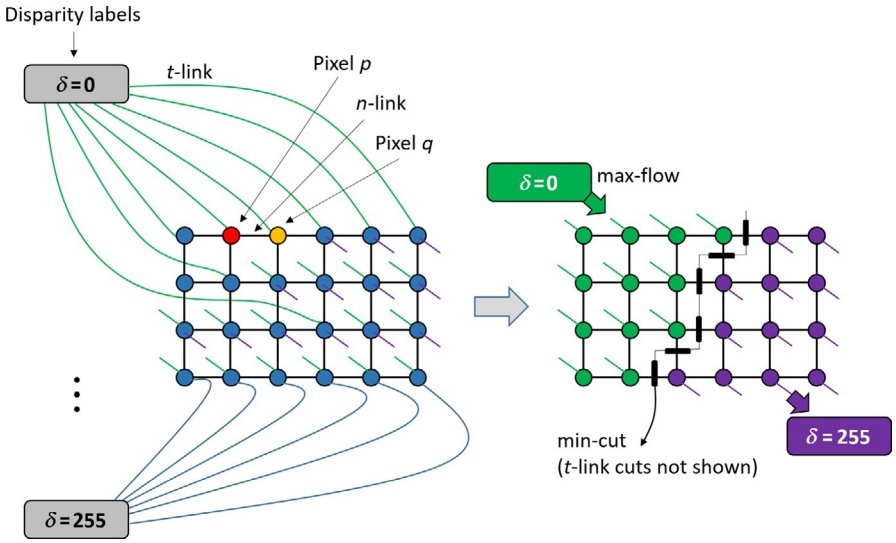

FIG. 1.36

Max-flow/min-cut approach for assigning disparities to pixels (right) from the original flow network (left).

value, either 0 (the rear finger) or 255 (for the large disparity front finger in 8-bit disparity format coding).

All pixels in the image correspond to nodes that are locally interconnected in a graph, along so-called *n*-links. The pixels/nodes are also linked—through the so-called *t*-links—to the aforementioned two disparity values 0 and 255, called disparity labels. These are thought of as a source and a sink, from/to which conceptually water will flow throughout the graph network. Both the *n*-links between pixels and the *t*-links between the pixels and the disparity labels are assigned a weighting/cost factor, which will somehow influence the water flow in the graph.

For the *n*-links, if two neighboring pixels p and q are eventually assigned the same disparity label, their *n*-link cost is zero. However, for a different disparity label, the *n*-link cost is a positive value. Such cost value effectively expresses the penalty of having two neighboring, *n*-link interconnected pixels that may end up with a different disparity label. This is one of the costs that the system will try to minimize, effectively favoring local smoothness in the assigned disparity labels over neighboring pixels.

For the *t*-links, the cost factor corresponds to the local disparity matching cost as in the previous section, between pixel p attached to the *t*-link, and the pixel $p + \delta_p^i$ for the associated disparity label δ_p^i. The cost is hence high when p is associated to a disparity label that is different from the real disparity that pixel p has in reality. This is a second cost that globally—in balance with the previous one over the *n*-links—should be minimized.

The global cost minimization in the Graph Cut technique is obtained by cutting the graph in two along a minimal cut (min-cut), i.e., a cut along which the sum of costs is minimal. In doing so, one subset of pixels will remain attached to the source and its associated disparity label, while the other subset of pixels is attached to the sink. These latter pixels get the sink disparity label assigned. There is clearly a trade-off for each pixel between getting the correct disparity label (the one that corresponds to reality) and keeping disparity smoothness with its neighbors.

There are actually no efficient algorithms that easily solve this min-cut problem, but its complementary problem—finding the maximum flow (max-flow) through the graph—has good implementations. As well explained in Ref. [40], gradually increasing the water flow coming out of the source to the sink, some links get saturated w.r.t. their capacity associated to the weighting costs, jeopardizing the further increase/maximization of the water flow through the network. Consequently, these are the links that should first be cut away in the equivalent min-cut problem statement, to enforce a higher water flow through the rest of the network.

This method can be generalized for N disparity labels, with a multimin-cut and equivalent multimax-flow approach, reaching N subsets of pixels, each assigned to the corresponding disparity label δ_p^i.

Further extensions to (i) inclusion of occlusion cost factors and (ii) more than two input images are presented in Ref. [41]. The DERS used in MPEG [42] for sustaining Free viewpoint TV (FTV) and 6-DoF VR applications uses such Graph Cut depth estimation approach.

1.4.2.2 Belief propagation

Belief propagation propagates evidence from some pixel to its neighbors by statistical inference. Similar to graph cuts, two competing costs—rather called potentials here—are introduced: a pixel intensity difference φ between a pixel p in the left image and the pixel q that is translated over an assumed disparity, as well as a smoothness ψ for neighboring pixels with unequal disparities. The difference, however, is that these functions are thought of as a statistical process.

For instance, in the function φ, the likelihood that pixels p and q under disparity δ have a different intensity I is expressed as a Gaussian distribution [43]. Each pixel will update its potential functions for a given disparity based on the observations/beliefs from adjacent pixels through iterative message passing, until a maximum a posteriori global potential encapsulating φ and ψ is reached under the disparity values eventually assigned to all pixels.

Fig. 1.37 [37] compares the results from the DERS of MPEG [42] applied on some test sequences of Fig. 1.11 with the belief propagation method described in Ref. [33]. Clearly, belief propagation obtains results that respect the object boundaries better than DERS. This was actually a requirement in all previously described methods.

Finally, Ref. [44] shows the similarities—both in concept and results—between Graph Cut and Belief propagation, which are the methods that in general rank best on the Middlebury data set [32].

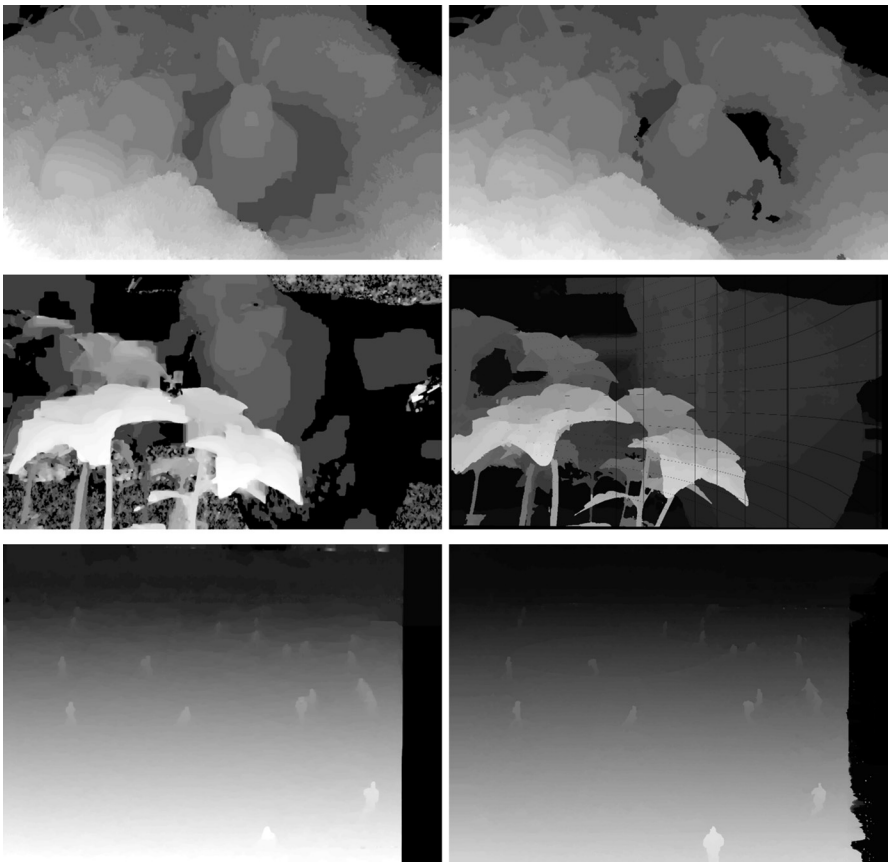

FIG. 1.37

DERS from MPEG (left) vs. Belief Propagation (right).

1.4.3 MULTICAMERA DEPTH ESTIMATION

Having only two input views as in the stereo matching techniques presented in the previous sections, is a highly limiting factor in reaching reliable depth estimation. Obviously, the more input images can be included in the depth estimation, the better the results are expected to be.

One straightforward extension of stereo matching consists in doing multiple, pairwise stereo matching estimations over the set of cameras, and combining the results. The multicamera setup of Fraunhofer HHI in Section 1.2.4.2 follows this approach and iteratively improves the estimated depth maps.

Other methods, however, exist for combining the information from all input cameras simultaneously in reaching a globally consistent depth estimation. Two methods are further detailed in the next sections.

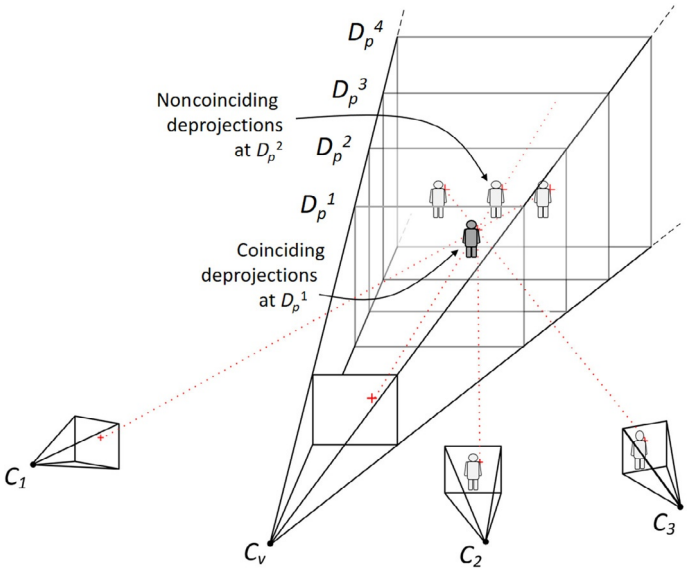

FIG. 1.38

Deprojections from camera views C_i to a depth plane D_p^j.

Courtesy of Hasselt University.

1.4.3.1 Plane sweeping

As the name suggests, plane sweeping [45] is a method where a plane sweeps over successive depth candidates $(D_p^1, D_p^2, \ldots, D_p^N)$ for each pixel p (or object P constituted of many pixels p). These depths are equivalent to the candidate disparities $(\delta_p^1, \delta_p^2, \ldots, \delta_p^N)$ presented in the stereo matching sections before.

All the camera input images are deprojected—i.e., projected from the 2D camera views into 3D space—onto a candidate depth plane D_p^i, as shown in Fig. 1.38. Note that the relative positions of the cameras (extrinsics), as well as their field of view and optical axis (intrinsics) should be known in this deprojection operation.

For objects that in reality really lie on the candidate depth plane D_p^i, the camera views will be deprojected toward coinciding images on the depth plane, cf. the brown/blue-clothed player in the middle of Fig. 1.39 (bottom). The reprojections of an object that originally is not positioned at the candidate depth plane D_p^i will be distinct for each camera view, yielding hardly overlapping, faded copies of the object, cf. each yellow player in Fig. 1.39.

Calculating the color histogram of the so-obtained depth plane image (i.e., all deprojections blended over each other into one image) will reveal whether the candidate depth plane D_p^i is the right depth plane or not: a very sharp histogram suggests that all deprojected images coincide at the right depth plane; otherwise the candidate depth plane D_p^i is not the right one for that object. In the latter case, the next candidate depth plane D_p^{i+1} is tested (i.e., deprojection and histogram tests), and all depth

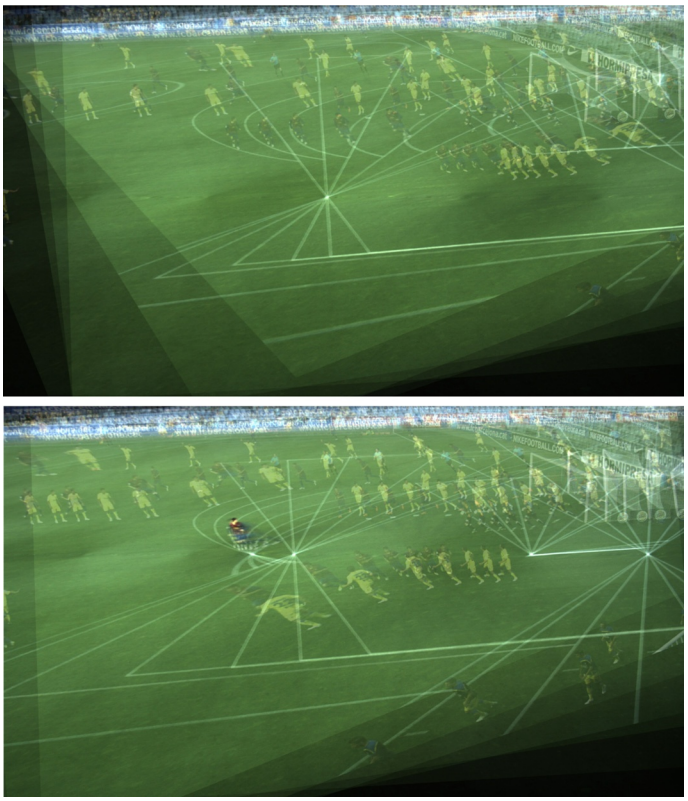

FIG. 1.39

Plane sweeping at different candidate depth planes.

planes are iteratively traversed, seeking for the peakiest histogram distribution to settle the depth plane for each object (or to be precise, for each pixel p over all objects).

This plane sweeping method has shown to yield very stable depth estimation results [7,46], even in wide baseline camera array configurations with large perspective view changes, cf. Fig. 1.40, at a quality level competitive to the Graph Cut depth map of Fig. 1.12. To be precise, Fig. 1.40 is a disparity map (front objects have higher gray-scale pixel values), corresponding to the depth planes in the depth sweeping method.

1.4.3.2 Epipolar plane images

The concept of epipolar plane images (EPIs) originates from the late 1980s [47] when having multiple cameras was still expensive, but rich fellows could start thinking of their possibilities.

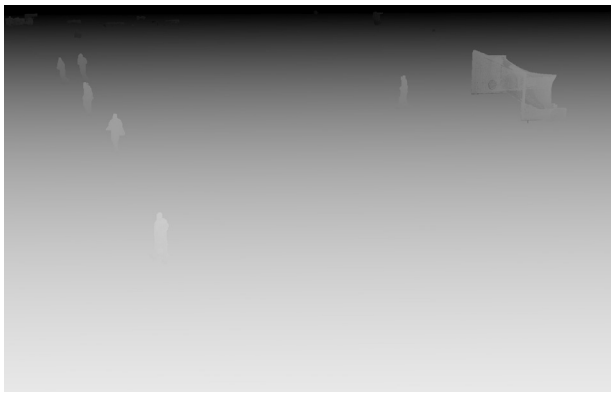

FIG. 1.40

Soccer depth map estimation results with plane sweeping.

To explain the concept, let us start by assuming a long, linear camera array with many small cameras put side by side, and registering all camera input images so that any feature point in one image always lies on the same scanline in the other images, cf. Section 1.3.1.2. These registered images are then stacked one behind the other, cf. Fig. 1.41. One horizontal section thereof is by definition an EPI, i.e., an Image that provides information about feature points in the same horizontal scanline—i.e., the same Epipolar Line after registration—over all registered input images.

The physical interpretation of such an EPI follows the disparity concepts presented in Section 1.1. Each horizontal scanline in the EPI originates from a specific input camera view, and when switching from one camera view to the next (thus from one scanline to the next in the EPI), a 3D point in space will be displaced as a function of its disparity. More specifically, a 3D point far away in space (large depth) will have almost zero disparity, hence it will follow an almost vertical line in the EPI. A 3D point close to the cameras, however, will have a large disparity and hence will make large translations from one horizontal scanline to the next in the EPI. Consequently, such 3D point will follow a quite diagonal line in the EPI, cf. Fig. 1.41, which we will call an EPI-D-trackline for simplicity of our discussion: the slope of such line in the EPI corresponds to the disparity δ or depth D of the associated 3D point in space.

Interestingly, an EPI also provides information about occlusions. Following two 3D points in their EPI-D-tracklines with associated disparities/depths D, we see how these points move from one camera view to the next. Since the point with smallest depth moves more rapidly than the point with largest depth, the former might catch up the latter, i.e., cause an occlusion, so-creating a crossing in their respective EPI-D-tracklines.

Ref. [48] obtains very detailed depth maps in using such techniques, even when reducing the number of cameras from around hundred to a dozen, which corresponds

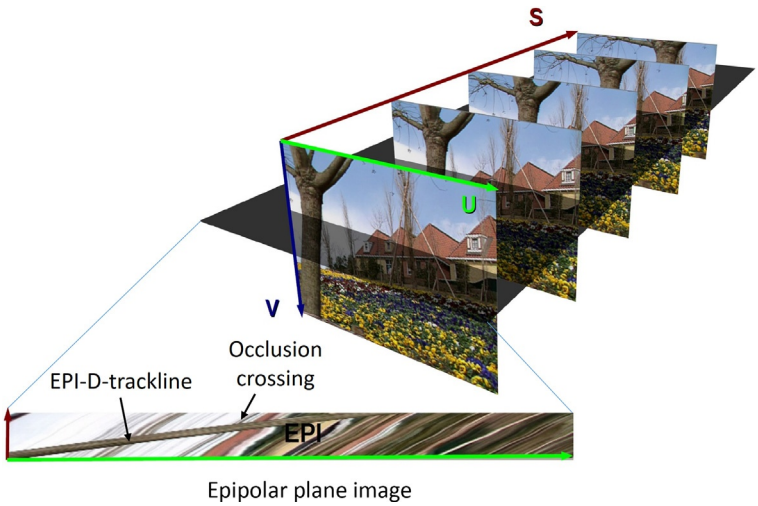

FIG. 1.41

Epipolar Plane Image is a horizontal section of a registered Multiview image stack.

to a heavily subsampled/discontinuous EPI, from which it is very difficult to extract the EPI-D-tracklines easily. Refs. [49,50] have further developed these techniques to create depth maps from any viewpoint, correctly synthesizing occlusions and disocclusions in the depth maps, which are used for further DIBR based virtual view synthesis.

1.5 VIEW SYNTHESIS AND VIRTUAL NAVIGATION

Virtual navigation in 3D scenes is a key feature of FTV, VR, and augmented reality systems. In order to allow the user to select virtual viewpoints, placed in positions different than real camera locations, a virtual view synthesis has to be performed. The concept of the virtual view synthesis is also exploited in storage and transmission, e.g., in high efficiency video coding with depth maps (3D-HEVC) extensions of high efficiency video coding (HEVC) to increase coding efficiency of a multiview video, cf. Section 1.6, as well as in error concealment techniques.

The fundamental goal of view synthesis is to generate any intermediate view in a position of a virtual camera placed in between the real captured (or transmitted) views, cf. Fig. 1.42. The content of those views is used within this process, which is hence often called view interpolation. The most common view synthesis employs DIBR in which two components of a view are used to render the new view: texture image and its associated depth map. Many input views can be used together, which constitutes a MVD scene representation format.

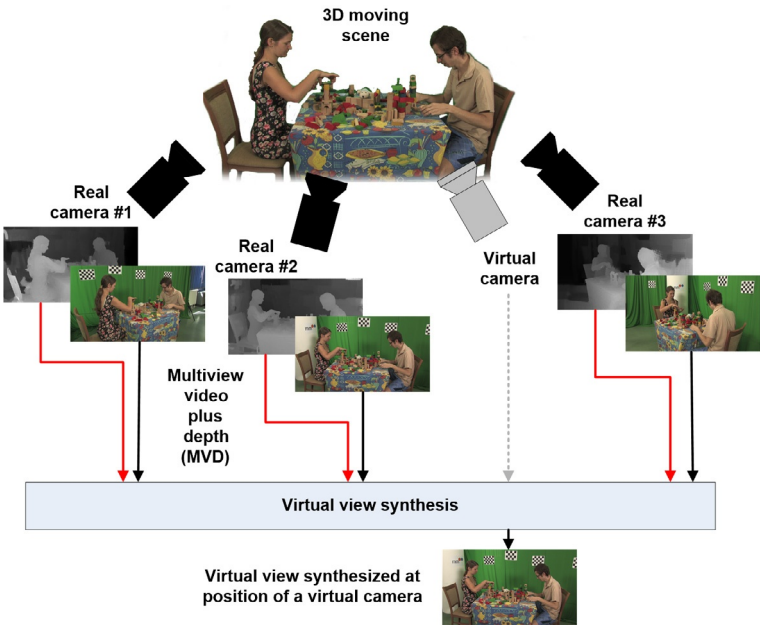

FIG. 1.42

Generation of virtual view in an intermediate position between real views.

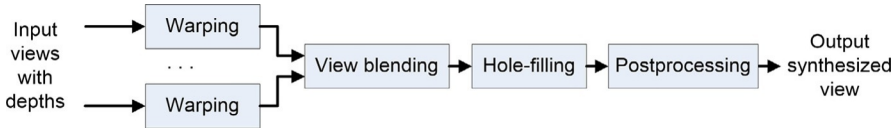

FIG. 1.43

General scheme of modern view synthesis algorithms.

Most modern view synthesis algorithms share a common scheme, cf. Fig. 1.43, consisting of warping the input views, view blending, inpainting (hole filling), and postprocessing. These steps are described in the following sections.

1.5.1 WARPING

The basic principle for generating new views based on images and depth is projective geometry presented in Section 1.2. The 3D information of the scene included in MVD data, as well as the camera arrangement (e.g., in the form of extrinsic and intrinsic camera parameters) is used to project pixels from the image plane of input view to 3D space and then reproject them back to an image plane, but this time related to the virtual view, cf. Fig. 1.44. Thus implicitly, in this process in 3D space,

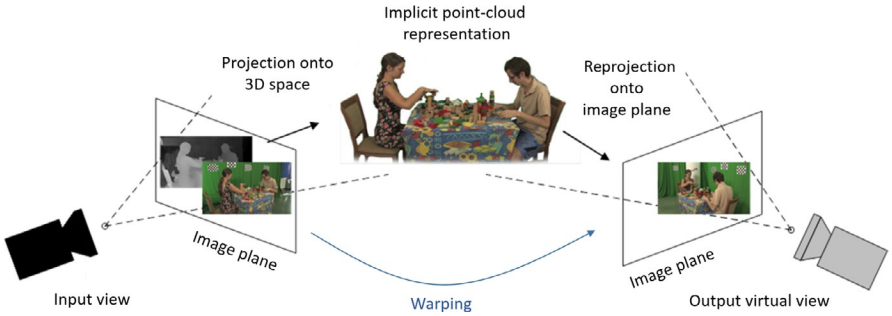

FIG. 1.44

Projection of pixels from the input image plane to 3D space and reprojection to the image plane of the output virtual view, which is equivalent to warping of pixels.

the scene is represented by means of a point cloud. Looked at from an end-to-end point of view, this step consists of warping pixels from an input to a different location determined by the associated depth. Of course, after projection some of the pixels may occlude others. Therefore care must be taken to use the corresponding depth information in order to cover farther pixels by the closer ones. This resembles the z-buffer occlusion techniques known from computer graphics [51].

There are two main approaches to perform this, called forward and backward projection (or synthesis).

Forward warping directly follows the scheme presented in Fig. 1.44. First, the output view is cleared with the background color (e.g., black) and an infinite depth value. Then, pixels from the input view are shifted to new positions according to their depth value. The pixels are written to the output virtual view, but only if their depth value is closer than the previously written depth value.

A major disadvantage of forward warping is that the generated depth (and thus the image) contains cracks on discontinuities between the projected pixels, cf. Fig. 1.45 (top). Even if in the input image two pixels are adjacent, after the warping they must be written into discrete positions, which—due to rounding—may yield a crack in between of them. This problem can be solved by projecting primitives like triangles instead of single pixels [54]. Another drawback of forward warping is that the generated depth cannot be postprocessed in order to improve the quality of the generated image.

Backward warping starts with processing of depth only, which is warped in the forward direction, like described before. Of course, this includes an occlusion test with the z-buffer technique. Then, the generated depth is postprocessed in order to fill the cracks. For example, this can be done with simple median filtering, cf. Fig. 1.45 (bottom-left). Finally, the depth is used to warp-back color content from the input image. Thanks to that, the resultant output image, cf. Fig. 1.45 (bottom-right) is almost free from cracks. The only holes that are generated are disocclusions, which are the parts of the scene that are not visible from the input view.

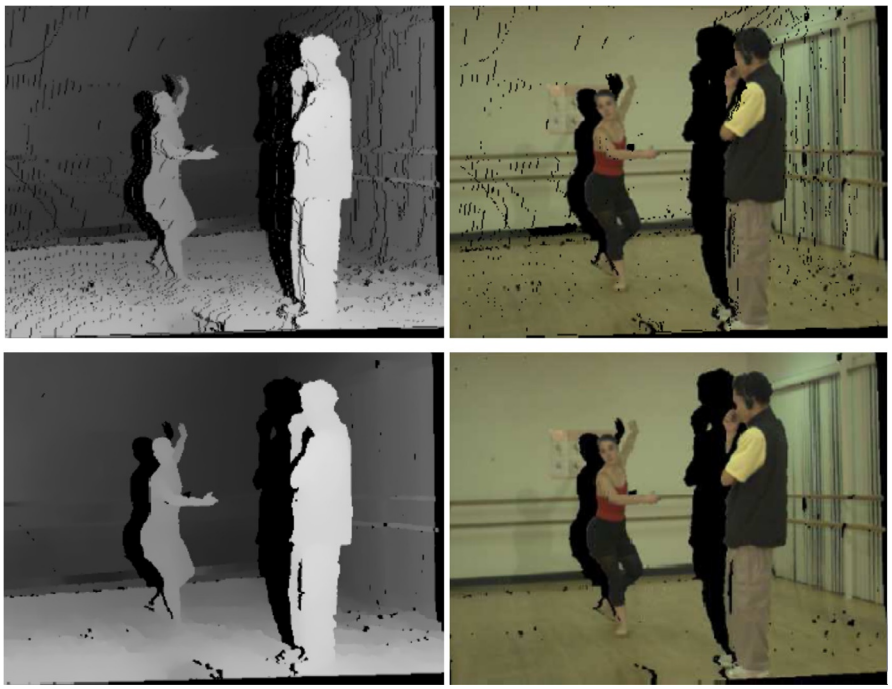

FIG. 1.45

View warping on example of "Ballet" sequence [52,53]. Forward warping leads to discontinuity artifacts in the depth (top-left) and consequently in the output image (top-right). Median filtering of the depth map (bottom-left) used in backward warping improves the results (bottom-right). Holes are marked in *black*.

Ballet sequence © Microsoft Research, rendering—courtesy of Poznań University of Technology.

Backward warping techniques are used more commonly than forward warping, i.e., in the MPEG VSRS [55] and in the view synthesis software from Poznan University of Technology [54].

1.5.2 VIEW BLENDING

The warping described in previous section exploits information only from a single view. Therefore in the output view, some of the regions cannot be synthesized because they are occluded in the input view. The occluded regions are different in each of the views, which can be used to the advantage of view synthesis algorithm to fill in some holes.

In the most common approach, multiple versions of the requested virtual view are synthesized, based upon another input view, cf. Fig. 1.46. Those multiple versions of the virtual view are then merged together with various blending methods in order to reach a single output view without disocclusions. For example, the colors from input

FIG. 1.46

Virtual view synthesized from left (A) and right (B) input view. Disoccluded regions are marked in *black*.

Ballet sequence © Microsoft Research, rendering—courtesy of Poznań University of Technology.

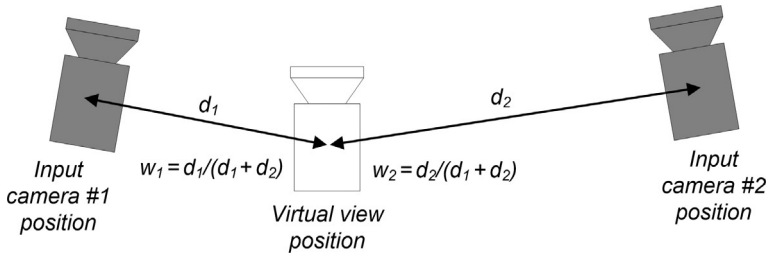

FIG. 1.47

Blending of pixels from two input views with weights calculated based upon distances in camera positions.

views can be linearly blended with weights w_i, calculated based on the distances d_i between the position of a given input view i and the requested virtual view:

$$w_i = 1 - \frac{d_i}{\sum_{k=1}^{N} d_k}, \tag{1.20}$$

where N is the number of input cameras used for the generation of variants of the requested output virtual view. Fig. 1.47 presents an example with two input cameras and weights w_1 and w_2. The result of such formulated view blending is presented in Fig. 1.48 (left).

1.5.3 INPAINTING

Though blending of multiple views substantially reduces the problem of disocclusion in the final synthesized image, it may not solve it entirely. For example, the viewer

FIG. 1.48

Virtual view (left) obtained from blending of two input views presented in Fig. 1.47, and the same image with inpainting of holes (right).

can select a viewpoint pointing at something which was occluded in all of the views. In such cases the missing pixels have to be colored with a generated value. This process of filling such regions with content generated from the pixel's neighborhood is called *inpainting*.

Depending on the size and the number of holes, various inpainting algorithms can be used. The simplest employs techniques of finding the nearest available pixel and using its color to fill the entire hole region. For example, such a technique is used in the MPEG VSRS [55]. As it can be seen in Fig. 1.48 (right), due to the usage of inpainting the holes vanish, providing satisfactory results in the "Ballet" sequence [52,53].

A more advanced inpainting method includes interpolation [54,56,57] and texture synthesis [58]. In the latter technique, the holes are filled with content which tries to resemble not only the color of the nearest neighboring pixels, but also higher order features, texture, like edges, gradients. Fig. 1.49 presents results obtained with texture synthesis inpainting [58] (left) compared to the nearest color inpainting in VSRS [55] (right).

1.5.4 VIEW SYNTHESIS REFERENCE SOFTWARE

VSRS has been originally developed in 2008 by Nagoya University [59] during MPEG studies on new 3D video coding standards. Later, in 2013, it has been enhanced to better address the needs arising in applications of FTV and Virtual Navigation [55]. During these years, VSRS has been continuously used in research studies as a technique of reference as well as a starting point for the implementation of new scientific ideas. As a consequence, it is the most commonly used view synthesis algorithm in papers in the area. Currently, the development of VSRS is coordinated in the FTV Ad-hoc Group in MPEG [60].

FIG. 1.49

Texture synthesis inpainting [58]. (Left) Compared to nearest color inpainting in VSRS [55] (right).

Cited from P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, T. Wiegand, Depth image-based rendering with advanced texture synthesis for 3-D video, IEEE Trans. Multimedia 13(3) (2011) 453–465, https://doi.org/10.1109/TMM.2011.2128862.

For completeness, Table 1.4 presents a description of the VSRS configuration file format used in the MPEG exploration studies.

The features of VSRS include:

- Backward warping with hole filling.
- Support for full 3D space warping as well as fast 1D warping (for linear camera arrangements).
- View blending in linear distance-based weights.
- Inpainting based on nearest-pixel blending.
- Subpixel accuracy thanks to half-pixel and quarter-pixel precision of synthesis.
- Postprocessing tools for improvement of boundaries of synthesized objects.

VSRS can be used freely for scientific and standardization purposes within MPEG. It can be downloaded from the MPEG SVN repository at the following address:

http://wg11.sc29.org/svn/repos/Explorations/FTV .

1.6 COMPRESSION

1.6.1 INTRODUCTION

As described in Section 1.2, a multiview video is acquired with the use of multiple cameras. The video is transmitted from the cameras, and somewhere, in an outside broadcasting van, in a preprocessing computer cluster or elsewhere, is the starting point from which the video from all the cameras is transmitted together. Such a video from multiple cameras is called “multiview video” under the assumption that the video sequences from all the cameras are synchronized as discussed in Section 1.2.

Table 1.4 VSRS Configuration File Description

Parameter	Description
DepthType	0: Camera space, 1: world space
SourceWidth	Input frame width
SourceHeight	Input frame height
StartFrame	Start frame index in image YUV files
StartFrameD	Start frame index in depth YUV files
TotalNumberOfFrames	Total number of input frames
LeftNearestDepthValue	z_{near} for the left input view
LeftFarthestDepthValue	z_{far} for the left input view
RightNearestDepthValue	z_{near} for the left input view
RightFarthestDepthValue	z_{far} for the left input view
CameraParameterFile	Name of file with camera parameters as described in Fig. 1.31 in Section 1.3
LeftCameraName	Name of real left camera, as specified in previous file
VirtualCameraName	Name of virtual camera, as specified in previous file
RightCameraName	Name of real right camera, as specified in previous file
LeftViewImageName	Name of left input image YUV file
RightViewImageName	Name of right input image YUV file
LeftDepthMapName	Name of left input image YUV file
RightDepthMapName	Name of right input image UYV file
OutputVirtualViewImageName	Name of output virtual view YUV file
SynthesisMode	0: General, 1: linear arrangement of cameras
ColorSpace	Internal color space: 0: YUV, 1: RGB
Precision	Upsampling precision: 1: integer-pixel, 2: half-pixel, 4: quater-pixel
Filter	Upsampling filter: 0: Bi-linear, 1: Bi-Cubic, 2: AVC
BoundaryNoiseRemoval	Postprocessing tool for removal of noise from synthesizer boundaries of objects: 0 or 1
ViewBlending	View blending disable (only in general mode): 0: weighted interpolation 1: use nearest
SplattingOption	0: disable; 1: enable for all pixels; 2: enable only for boundary pixels. Default: 2
BoundaryGrowth	A parameter to enlarge the boundary area with SplattingOption = 2. Default: 40
MergingOption	0: Z-buffer only; 1: averaging only; 2: adaptive merging using Z-buffer and averaging. Default: 2
DepthThreshold	A threshold is only used with MergingOption = 2. Range: 0–255. Default: 75
HoleCountThreshold	A threshold for number of holes

It would be a truism to say that the uncompressed multiview video needs extremely high bitrates for the transmission. For example, only 10 streams of consumer high-definition (HD) video (1920×1080 pixels in a frame, 25 frames/s) need a total bitrate of 6.22 Gbps even excluding all necessary auxiliary data, e.g., synchronization data. Therefore efficient compression of multiview video is of paramount importance. Multiview video compression is obviously related to general video compression technology developed mostly for video taken from a single camera, i.e., monoscopic video. The state-of-the-art technology of monoscopic video compression is very sophisticated, and therefore development of a new generation of video is inconceivably expensive. On the other hand, compression of multiview video is needed for various multiview video applications (cf. Section 1.1), i.e., for quickly growing but still niche markets. Therefore it would be too expensive to develop from scratch a compression technology just for multiview video. As a consequence, the multiview video compression exploits the technology of monoscopic video compression. A deeper description of the monoscopic video compression technology may be found elsewhere [61–64] and will be omitted here.

For compression of multiview video, the commonly used technology of monoscopic video compression may be used in two different approaches.

1. *Simulcast coding*. This term means that for each view, the video is encoded independently, i.e., the similarities among the views are not exploited by compression. Each video is compressed using the standard compression techniques developed for monoscopic video. Therefore the commonly used codecs, both hardware and software, are used. This approach is straightforward, easy to implement, and therefore cheap and eagerly used. A specific version of simulcast coding is widely used for stereoscopic video broadcasting and multicasting by television and IP networks. Nowadays, these "3D video" services use so-called frame-compatible transmission of stereoscopic video, where the left and the right views are appropriately subsampled and merged into single frames. In such a frame, the left half corresponds to the left view, while the right half corresponds to the right view ("side-by-side" format). Alternatively, the left and the right decimated views may be packed into the top and bottom parts of frames ("top-and-bottom" format). Some other ways of packing two views into single frames have been proposed and even standardized. The advantage of the "frame-compatible" formats is that the corresponding video may be compressed by standard mono video codecs, and the whole television transmission infrastructure needs no modification for stereoscopic video. The only specific information usually added to the bitstream is the metadata that signals the presence and the format of multiview video packed into frames. As the considerations of this chapter are focused on video with a higher number of views, the compression of stereoscopic video in the "frame-compatible" formats will not be further considered here.
2. *Multiview and 3D video coding*. These approaches exploit the similarities between views, therefore stronger compression is possible as compared to the

simulcast coding. The multiview video coding exploits strong similarities between horizontally aligned blocks of samples in the neighboring views. The 3D video coding additionally takes advantage from more sophisticated relations between the views and depth maps, thus achieving even more bitrate reduction than the purely multiview techniques. Both multiview and 3D video coding techniques are developed on the top of the standard monoscopic video coding techniques. They use the coding tools from the respective monoscopic video compression techniques, but they also use some additional coding tools that exploit the similarities between views. The 3D video coding techniques may additionally use the tools capable to exploit the spatial information about the scene in order to further reduce the bitrate. Application of these additional tools implies that the respective codecs are not just the popular monoscopic codecs but their specific modifications. Such extended multiview or 3D video codecs are still rarely used in practice, but the prospective applications in VR, virtual navigation, SMV and lightfield technology (see Section 1.1) are likely to stimulate increasing interest in their usage.

Video coding, either monoscopic, multiview, or 3D, may be lossless or lossy. The term "lossless coding" refers to such compression where the samples of the decoded video exhibit exactly the same values as the respective samples in the original video. Lossless compression of video is relatively rarely used and considered, so multiview or 3D lossless compression is still practically beyond the scope of research, and will be not considered here. Therefore the further considerations of this section are limited to lossy coding that always results in some distortions of the decoded video. For lossy coding, there exists a trade-off between the quality of the decoded video and the bitrate of the compressed bitstream, cf. Fig. 1.50. For commercial broadcasting, in

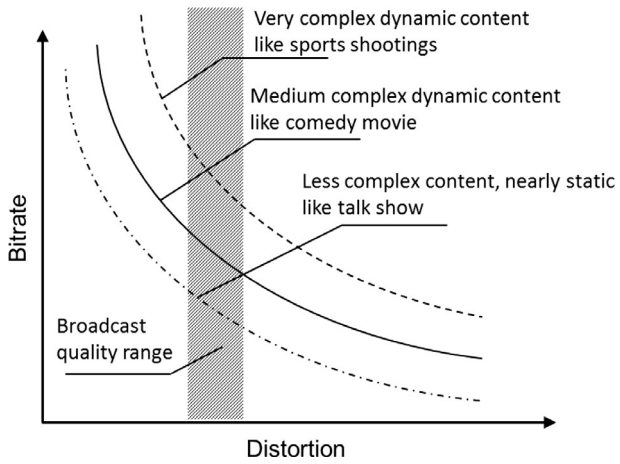

FIG. 1.50

The "rate-distortion" lines in video coding.

television, as well as for over-the-top video, so-called broadcast quality is required. It means that the coding artifacts should be either invisible or hardly visible. The quality and bitrate are controlled by quantization steps of the quantizers deployed for the transformed samples of the prediction errors. Changing the value of the quantization step moves the operational point of the codec along a "rate-distortion" curve that corresponds to the current video. Such a curve strongly depends on video content as depicted in Fig. 1.50.

1.6.2 MONOSCOPIC VIDEO CODING AND SIMULCAST CODING OF MULTIVIEW VIDEO

During the last quarter of a century, consecutive generations of video coding technology have been developed and promoted to the international standards, such as MPEG-2 [65], AVC (advanced video coding) [66], and HEVC [67]. These consecutive video coding generations have been developed thanks to huge research efforts. For example, the development, optimization, standardization, and implementation of HEVC needed an effort measured in thousands of man-years. This estimation implies that the development of a multiview video codec from scratch would be definitely too expensive, as already mentioned.

Obviously, the previously mentioned three milestone standards of video coding do not represent the only existing video coding technologies. There are many others [68], but they are less important for multiview video coding, at least hitherto. When considering the three previously mentioned representative video coding standards, the following "rule" may be formulated [69,70], cf. Fig. 1.51:

(1) For each next generation, for a given quality level, a given video format and content complexity, the bitrate is halved.
(2) The temporal intervals of about 9 years were observed between each consecutive technology generations of video coding.

During a 9-year interval the available computational power is increased by a factor of about 20, according to Moore's law. After each such interval, this increase may be consumed by more sophisticated codecs of the next generation. The previously mentioned "rule of thumb" was observed during two cycles only. It is probably too short a time to establish a rule that may be used to forecast the future developments. Nevertheless, the ISO/IEC expert group on audio and video (MPEG) has recently started the exploration activities aimed at creating a new compression technology that should be capable of reducing the bitrates again, possibly by a factor of about two. Its name is not fixed yet, but the experts use the tentative name "Future Video Coding." The expectations are to have this technology standardized around 2020–21, which would be roughly compliant with the previously mentioned rule of thumb.

Each video coding standard is related to some reference software that provides nearly the highest compression possible for this standard defining the compressed bitstream semantics and syntax. Therefore the rate-distortion performance of the reference software is a good model for a given compression technology. It should be

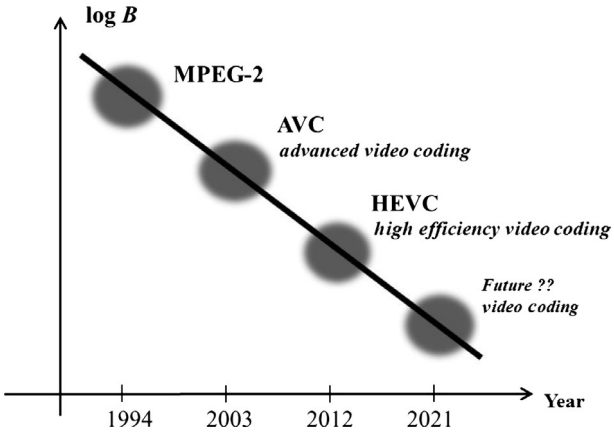

FIG. 1.51

The milestone generations of video coding. *B* stands for the least possible bitrate needed to achieve the assumed quality for the assumed complexity of video content and an assumed video format.

noted that for a given standard, the minimum bitrate required for a given quality level may change dramatically from the first encoders on the market until mature, sophisticated designs become available after some years (e.g., [71]). The latter ones mostly provide the compression similar to that provided by the reference software.

For a given content type, and for a given video format, the bitrate of the compressed bitstream may be roughly assessed assuming the required quality level and a mature codec implementation that reaches nearly the same compression as the standard reference software. For demanding complex dynamic content and assuming broadcast quality levels, for monoscopic video codecs the bitrate B may be very roughly estimated using the following equation [69,72,73]

$$B \approx A \cdot V \ (\text{Mbps}) \tag{1.21}$$

where A is a technology factor, where $A=4$ for MPEG-2, $A=2$ for AVC, $A=1$ for HEVC, $A=0.5$ for the prospective technology expected around year 2020–21, and V is video-format factor, where $V=1$ for the Standard Definition (SD) format, (either 720×576, 25 fps or 720×480, 30 fps, chroma subsampling 4:2:0, i.e., one chroma sample from each chroma component C_R and C_B per 4 luma samples), $V=4$ for the High Definition (HD) format (1920×1080, 25/30 fps, chroma subsampling 4:2:0), $V=16$ for the Ultra High Definition (UHD) format (3840×2160, 50/60 fps, chroma subsampling 4:2:0).

Eq. (1.21) defines the very rough estimate for the bitrate of a compressed bitstream, from which broadcast-quality video may be retrieved. Some examples of these estimations are presented in Table 1.5.

The last column of Table 1.5 shows that the state-of-the-art video codecs represent video using a surprisingly small numbers of bits. Even for demanding content,

Table 1.5 Video Compression for Main Broadcasting Formats Assuming Complex Video Content (f_o Denotes the Frame Rate)

Standard	Video Format	Approximate Bitrate (Mbps)	Approximate Average Number of Bits per Pixel of Color Video
MPEG-2	SD (720×576, $f_o = 25$ Hz)	4	0.40
AVC	HD (1920×1080, $f_o = 25$ Hz)	8	0.15
HEVC	UHD (3840×2160, $f_o = 50$ Hz)	16	0.04

the HEVC bitstreams can reach 1 bit in average for 25 color pixels! This result is even expected to be improved by reaching 1 bit in average for 50 color pixels for the forthcoming new standard expected until 2021. It is worth emphasizing that these numbers refer to all bits in the bitstream including all control data, motion vectors, etc. This means that for most blocks of samples no data is transmitted. The content of such blocks is indeed well predicted using sophisticated interframe and intraframe predictions. The description of the relevant algorithms and tools may be found in Refs. [74,75] for MPEG-2 [61,76], for AVC, and for HEVC [62–64].

Each consecutive video compression standard defines an increased number of coding tools. As it would be not economically efficient to implement all these tools in all codecs, the standards define profiles that correspond to sets of the coding tools. In fact, a profile is a specific subset of the bitstream syntax defined by the standard. Obviously, codecs of different profiles accept different sets of input video formats, provide different compression efficiency, are characterized by different complexity, etc. The previous considerations of this section assume the codec profiles widely used in video broadcasting, i.e., Main Profile for MPEG-2, High Profile for AVC, and Main Profile for HEVC. Somewhat higher bitrates of the HEVC bitstreams are necessary to handle 10-bit video samples using Main10 Profile of HEVC that is required by Wide Dynamic Range and Wide Color Gamut video as defined by the recently introduced Recommendation H.2020 of International Telecommunication Union (ITU) [77]. The issues of Wide Dynamic Range and Wide Color Gamut need also to be explored for future multiview video systems.

The conceptually simplest way to implement the simulcast coding of multiview video is to encode each view as an independent video stream with the time stamps included. In that way, the commonly used relatively cheap video codecs may be efficiently applied. The total bitrate B_m of the bitstreams is

$$B_m = N \cdot B, \tag{1.22}$$

where N is the number of views, B is the bitrate for a single view from Eq. (1.21).

This equation is similar to Eq. (1.7) in Section 1.2.3, except that here B stands for the bitrate of the compressed video from one view. Eq. (1.22) is written under the

View 1	View 2	View 3	View 4
View 5	View 6	View 7	View 8
View 9	View 10	View 11	View 12
View 13	View 14	View15	View 16

FIG. 1.52

"Frame-compatible" packing of 16 HD views into an "8K" UHD frame.

assumption that the bitrates B are similar for all the views, as it is often the case for multiview video. For example, according to the previously mentioned formulas, the bitrate for the HEVC simulcast of 10 views in the HD (1920×1080) format would be around 40 Mbps for very dynamic and demanding content.

Another option for simulcast compression of multiview video is "frame-compatible" coding, where several views are packed into a frame, similar to the transmission of stereoscopic video (cf. the previous section). Such a frame consists of the views taken in the same time instant, cf. Fig. 1.52. In that way, multiview video may be compressed by a standard monoscopic encoder. The advantage is also that no side synchronization information is needed. The limitation of this approach is related to the quantitative limitations of the encoders available on the market. They mostly allow for encoding of HD video or sometimes UHD video, although the current HEVC specification (3rd ed.) foresees the frame size (the picture size) up to 35.65 millions of samples. That frame size is sufficient for "8K" format of UHDT (7680×4320) and to accommodate up to 16 views in the HD (1920×1080) format. The "8K" codecs together with other equipment for "8K" ultrahigh definition television systems are currently under intensive development, and they may be an interesting option for compression of multiview video already around the year 2020. Unfortunately, the metadata for "frame-compatible" coding is yet not standardized for such applications.

It should be also emphasized that the bitrates discussed before are sufficient only for the video delivery to a final viewer, and are far insufficient for the production environment, where either high-fidelity versions of the video codecs or still image codecs may be used. Currently, the frequently used solutions exploit the JPEG2000 standard, or more exactly speaking the Motion JPEG2000 standard [78–82] where all the frames of video are encoded independently like still images. The bitrates of compressed HD video are mostly in the range between 50 and 100 Mbps. Applications of such a compression technology to multiview video would lead to very high total bitrates, and high-fidelity versions of the video codecs like HEVC High Tier or even All Intra are advantageous options.

Hitherto, we have considered simulcast coding of multiview video. Another issue is simulcast coding of multiview video with depth maps. For such purposes the depth representations are standardized in MPEG-C Part 3 [83]. A depth map may be treated as monochrome video, thus may be compressed using monoscopic video codecs (like

AVC or HEVC) configured to compress monochrome video, or using still image codecs (e.g., JPEG or JPEG2000) that are capable to compress monochrome images. For the scenario where both views and depth maps are encoded using the AVC codecs, the bit allocation between views and depth has been studied in detail in order to achieve the best quality of virtual views, e.g., in Ref. [84]. Such studies imply that the depth maps mostly consume 10%–30% of the total bitrate assuming that for each view the respective depth map is encoded.

Such video and image codecs like AVC and HEVC are developed for natural color pictures while depth is monochrome and featured by sharp edges and large flat areas. Therefore the more efficient approach is to use special depth compression techniques instead of standard video coding techniques. These special depth coding techniques include platelet-based [85], wedgelet-based [86], contour-based [87], synthesis-based [88], and other techniques.

1.6.3 MULTIVIEW VIDEO CODING

The main idea of the multiview video coding is to exploit the similarities between neighboring views. One view, called the base view, is encoded like a monoscopic video. During the encoding, the standard intraframe and temporal interframe predictions are used. The respective bitstream constitutes the base layer of the multiview video representation. The base view may be decoded from the base-layer bitstream using a standard monoscopic decoder. For the other views called the dependent views, in addition to intraframe and interframe predictions, the interview prediction with disparity compensation may be used, cf. Fig. 1.53. In such prediction, a block in a dependent view is predicted using a block of samples from a frame from another view in the same time instant. The location of this reference block is pointed out by the disparity vector. This interview prediction is dual to the interframe prediction, but the motion vectors are replaced by the disparity vectors, and the temporal reference frames are replaced by the reference frames from other views. Moreover, also other

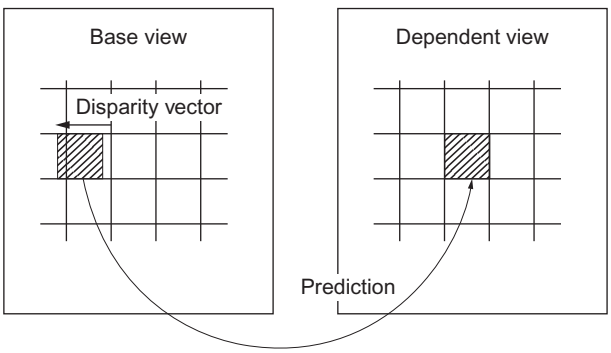

FIG. 1.53

Interview prediction with disparity compensation.

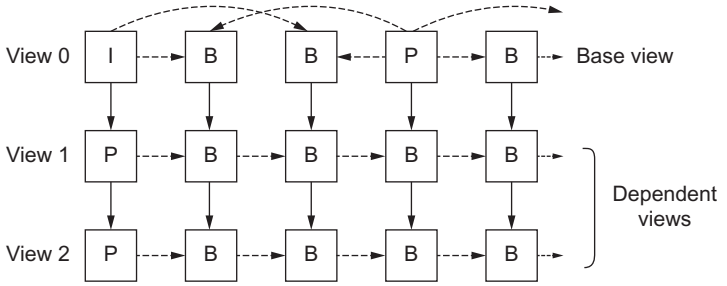

FIG. 1.54

Example of the frame structure in multiview video coding using interview prediction with disparity compensation: solid line arrows denote interframe predictions, while dashed line arrows correspond to temporal predictions. The letters I, P, and B denote I-frames (intraframe coded), P-frames (compressed using intra- and interframe coding), and B-frames (compressed with the additional use of a second interframe prediction or interview prediction).

coding tools have been proposed, e.g., compensation for illumination differences that allows to further increase compression efficiency [89].

In multiview video coding, the pictures are predicted not only from temporal interframe references but also from interview references. An example of a prediction structure is shown in Fig. 1.54.

Multiview video coding has been standardized as extensions to the MPEG-2 standard [74], the AVC standard [90], and the HEVC standard [91]. The multiview extension of AVC is denoted as MVC and that of HEVC as MV-HEVC. The respective profiles of standards are summarized in Table 1.6. These multiview extensions have been standardized in such a way that only minor modifications are needed to the monoscopic codec implementations. Therefore some more advanced techniques for multiview coding are not included into the standards. The extensions of the AVC and HEVC standards provide also the ability to encode the depth maps, where nonlinear representations are allowed [92].

The multiview coding provides the bitrate reduction in the order of 15%–30%, sometimes reaching even 50% as compared to the simulcast coding. These high bitrate reductions are achievable for video that is obtained from cameras densely located on a line, and then rectified in order to virtually set all the optical axes parallel and on the same plane. For sparse and arbitrary camera locations, the gain with respect to the simulcast coding reduces significantly.

1.6.4 3D VIDEO CODING

The distinction between multiview video coding and 3D video coding is not precise. The latter refers to compression of the multiview plus depth representations using more sophisticated techniques of interview prediction. A great diversity of 3D video

Table 1.6 Multiview Coding Profiles in the Video Compression Standards

Standard	Profile	Description
MPEG-2	Multiview	Not in use
AVC	Stereo High	Base layer compatible with High Profile Limited to stereo (two views only) Tools for interlaced video coding Adopted in consumer stereoscopic video recording (3D Blu-Ray) Some software video players provide support for this profile
	Multiview High	Base layer compatible with High Profile Arbitrary number of views No tool for interlaced video coding
	Multiview Depth High	Support for coding of depth maps together with multiview video Multiview video coding compatible with either stereoscopic high profile (two views) or multiview High Profile (more views allowed)
HEVC	Multiview Main	Base layer compatible with Main Profile Support for coding of multiview video together with depth maps

coding tools has been already proposed including prediction based on view synthesis, interview prediction by 3D mapping defined by depth, advanced inpainting, coding of disoccluded regions, depth coding using platelets and wedgelets, etc. [86,93–97]. Some of these tools have been already included into the standards of 3D video coding: 3D High Profile of AVC [66,98] and 3D Main Profile of HEVC [67,91]. The latter defines the state-of-the-art technology for compression of multiview video with accompanying depth.

3D-HEVC is an extension of the multiview coding in HEVC, i.e., MV-HEVC. Similarly, as in MV-HEVC, the standardization requirement was to reuse the monoscopic decoding cores for implementations. MV-HEVC, 3D-HEVC, and the scalable extension of HEVC share nearly the same high-level syntax of the bitstreams. Therefore it was decided that view encoding cannot depend on the corresponding depth. As compared to MV-HEVC, 3D-HEVC provides additional prediction types:

(1) Combined temporal and interview prediction of views that refers to pictures from another view and another time instant;
(2) View prediction that refers to a depth map corresponding to the previously encoded view;
(3) Prediction of depth maps using the respective view or a depth map corresponding to another view.

The compression gain of 3D-HEVC over MV-HEVC is around 2%–12% bitrate reduction. Nevertheless, the compression gains of both 3D-HEVC and MV-HEVC

are smaller when the cameras are not aligned on a line. For circular camera arrangements, in particular with the angles between the camera axes exceeding 10 degrees, the gain over simulcast falls below 15%, often being around 5%. This observation stimulated research on the extensions of 3D-HEVC that uses true 3D mapping for more efficient interview prediction [99,100]. Such extension of 3D-HEVC has been proposed in the context of transmission of the multiview plus depth representations of the dynamic scenes in the future free-viewpoint television systems [73].

The compression efficiency of simulcast HEVC, MV-HEVC, 3D-HEVC, and extended HEVC for arbitrary camera locations has been discussed in Ref. [100] using the coding results for 10 well-known test multiview sequences:

- with linear camera arrangements: Poznan_Street, Poznan_Hall 2 [23], Dancer [101], Balloons, Kendo [102], Newspaper [103],
- with circular camera arrangements: Poznan_Blocks [104,105], Big Buck Bunny Flowers [106], Ballet, Breakdancers [53].

The results obtained using the PSNR index for luma quality assessment for coding of three selected views are presented in Fig. 1.55 where the bitrate reductions are calculated using the Bjøntegaard metric [107]. These results clearly illustrate the conclusions drawn previously for both MV-HEVC and 3D-HEVC. In particular, the results show that for cameras sparsely located on an arc (like for test sequences Poznan_Blocks and Big Buck Bunny/BBB_Flowers) the compression gains of MV-HEVC and 3D-HEVC over the simulcast HEVC are negligible.

Fig. 1.55 also shows some gains for more advanced codecs that were recently proposed but that are not standardized [100].

1.7 FUTURE TRENDS

This chapter has provided an overview of DIBR techniques using Multiview plus Depth, developed within MPEG throughout the last decade in the context of FTV. The 3D scene is captured with multiple cameras from which depth is estimated, relying on algorithms that have been an active field of research over the last couple of decades.

3D video coding is currently a research topic for several groups around the world, and also future standardization activities are expected. Recently, MPEG-FTV, the body within MPEG (Moving Picture Experts Group of ISO and IEC) exploring FTV studies possible 3D-HEVC extensions for efficient coding of multiview video taken from arbitrary camera positions. Current industrial deployment of 3D-HEVC is rare, but growing interests in the applications hitherto mentioned in this chapter will stimulate applications of this compression as well as, probably, standardization of its more efficient extensions. It is also expected that the coding tools of 3D-HEVC together with possible improvements will be included, probably with some delay,

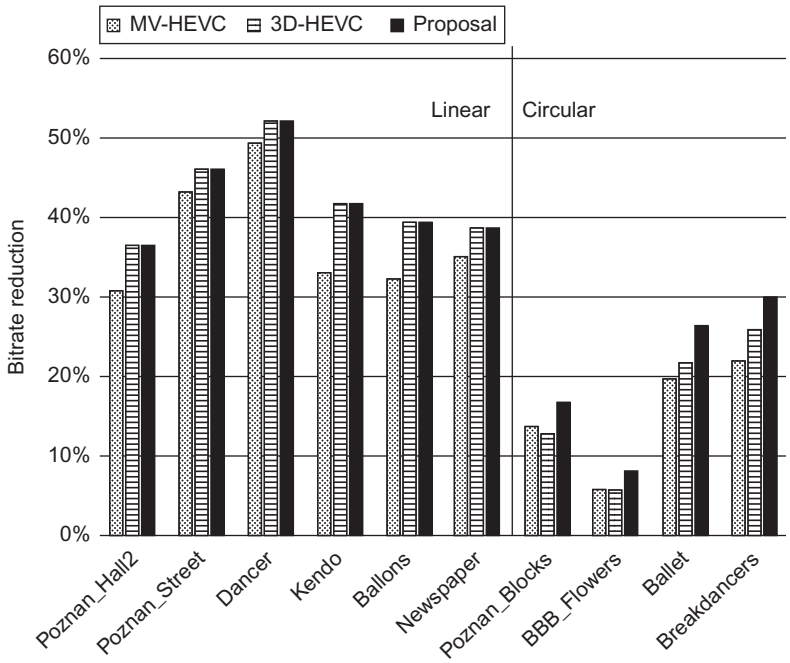

FIG. 1.55

Bitrate reduction against HEVC simulcast for various multiview test sequences with linear and circular camera arrangements along or around a scene. The data are provided for MV-HEVC and 3D-HEVC reference software and for the extension proposed in Ref. [100].

into the forthcoming future video coding standard that is expected to be ready around 2020–21 as its first version.

With the advent of VR, new challenges also arise at the horizon:

- Propose market-friendly solutions to capture the 3D scene at low cost with only a handful of cameras
- At least support real-time decoding and rendering at low latency
- Overcome cyber sickness, among others, by enabling 6-DoF user motion

Especially the latter will require virtual view synthesis techniques, going far beyond nowadays 3-DoF VR and 360 video streaming, which essentially select a viewport from a high-resolution texture.

At this point of the exploration and standardization discussions in JPEG and MPEG, it is believed that Light Fields embedding directional light information might be a solution to this 6-DoF Photorealistic/Cinematic VR. Other candidates like Point Clouds are also considered a viable solution.

Interestingly, Refs. [1,11] suggest many similarities between aforementioned data representations. Even though Light Fields, DIBR/Multiview plus Depth, and Point Clouds might—for the man in the street—look very different from each other, they are actually not.

The authors believe that there is an urgent need to bring experts from these different fields together, looking for a common denominator between these technologies, with the aim to reuse as much as possible existing coding solutions, slightly modified to the next-generation VR application needs.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Centre, Poland, within project OPUS according to the decision DEC-2012/05/B/ST7/01279, and within project PRELUDIUM according to the decision DEC-2012/07/N/ST6/02267, as well as project 3DLICORNEA funded by the Brussels Institute for Research and Innovation, Belgium, under Grant No. 2015-DS/R-39a/b/c/d.

The authors would also like to thank the fruitful discussions with members from the JPEG and MPEG standardization committees, and experienced colleagues having participated to the project DREAMSPACE funded by the European Commission.

GLOSSARY

3D mesh model	a 3D data representation model where the 3D shape of the object and its 2D texture are used in the rendering process
Coding	the process by which the redundancies in a data representation model are exploited to reduce the amount of information (bitstream) to represent the data
Depth image-based rendering	the rendering process using input images and depth maps to synthesize the virtual viewpoint
Point cloud	set of 3D data points in space, which essentially correspond to the vertices of a 3D mesh model, but without including any connectivity. In practice, point clouds should be dense to obtain good rendering quality
Rendering	the process by which some visual data is processed for visualization on a display
Super-multiview video	high density array of video views captured by a linear or curvilinear arrangement of conventional cameras

REFERENCES

[1] F. Pereira, E.A.B. da Silva, G. Lafruit, Plenoptic imaging: representation and processing, in: R. Chellappa, S. Theodoridis (Eds.), Academic Press Library in Signal Processing, vol. 6, Elsevier, Academic Press, New York, 2018, pp. 75–111.

[2] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, S. Sullivan, High-quality streamable free-viewpoint video, in: ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2015, vol. 34(4), August 2015 (Article No. 69).

[3] A. Jones, J. Unger, K. Nagano, J. Busch, X. Yu, H.-Y. Peng, O. Alexander, M. Bolas, P. Debevec, An automultiscopic projector array for interactive digital humans, in: Proceeding SIGGRAPH'15 ACM SIGGRAPH 2015 Emerging Technologies, 2015, https://vimeo.com/128641902 (Article No. 6).

[4] TimeSlice Films, Tim Macmillan Early Work 1980–1994, 2009, https://vimeo.com/6165108.

[5] TimeSlice Films, Sky High Jump, 2012, https://vimeo.com/49921117.

[6] M. Domański, A. Dziembowski, A. Grzelka, D. Mieloch, O. Stankiewicz, K. Wegner, Multiview Test Video Sequences for Free Navigation Exploration Obtained Using Pairs of Cameras, ISO/IEC JTC 1/SC 29/WG 11 Doc. M38247, Geneva, Switzerland, 2016, http://mpeg.chiariglione.org/news/experiencing-new-points-view-free-navigation.

[7] P. Goorts, S. Maesen, M. Dumont, S. Rogmans, P. Bekaert, Free viewpoint video for soccer using histogram-based validity maps in plane sweeping, in: Proceedings of The International Conference on Computer Vision Theory and Applications (VISAPP), 2014, pp. 378–386, https://www.youtube.com/watch?v=6MzeXeavE1s.

[8] ScanLab projects Point Cloud data sets, http://grebjpeg.epfl.ch/jpeg_pc/index_galleries.html.

[9] T. Gurdan, M.R. Oswald, D. Gurdan, D. Cremers, Spatial and temporal interpolation of multi-view image sequences, in: 36th German Conference on Pattern Recognition, Münster, September 2–5, 2014, http://www.tobiasgurdan.de/research/.

[10] A. Schenkel, Corrections géométriques et colorimétriques automatisées de modèles tridimensionnels de grande taille (Ph.D. Thesis), LISA Department, Université Libre de Bruxelles, 2016.

[11] Technical Report of the Joint Ad Hoc Group for the Digital Representations of Light/Sound Fields for Immersive Media Applications, ISO/IEC JTC1/SC29/WG1/N72033 & ISO/IEC JTC1/SC29/WG11/N16352, Geneva, Switzerland, June 2016, http://mpeg.chiariglione.org/sites/default/files/files/standards/parts/docs/w16352_2016-06-03_Report_JAhG_light-sound_fields.pdf.

[12] http://www.openexr.com/.

[13] 754-1985 – IEEE Standard for Binary Floating-Point Arithmetic, 754-2008 – IEEE Standard for Floating-Point Arithmetic.

[14] https://en.wikipedia.org/wiki/Kinect.

[15] Mesa Imaging, SR4000 Data Sheet, https://www.inf.u-szeged.hu/sites/default/files/ipcglab/docs/vision/SR4000_Data_Sheet.pdf.

[16] T. Grajek, R. Ratajczak, K. Wegner, M. Domański, Limitations of vehicle length estimation using stereoscopic video analysis, in: 20th International Conference on Systems, Signals and Image Processing, IWSSIP 2013, Bucharest, Romania, July 7–9, 2013, pp. 27–30.

[17] ST 259:2008: For Television – SDTV1 Digital Signal/Data – Serial Digital Interface, https://doi.org/10.5594/S9781614824077, ISBN 978-1-61482-407-7, 2008.

[18] https://en.wikipedia.org/wiki/LANC.
[19] M. Tanimoto, Ray Capture Systems for FTV, Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), Asia-Pacific, 2012, ISBN: 978-1-4673-4863-8.
[20] F. Zilly, Method for the Automated Analysis, Control and Correction of Stereoscopic Distortions and Parameters for 3D-TV Applications (Doctoral Thesis), 2015.
[21] I. Feldmann, M. Mueller, F. Zilly, R. Tanger, K. Mueller, A. Smolic, P. Kauff, T. Wiegand, HHI Test Material for 3D Video, ISO/IEC JTC1/SC29/WG11, Doc. m15413, France, Archamps, 2008.
[22] M. Domański, K. Klimaszewski, J. Konieczny, M. Kurc, A. Łuczak, O. Stankiewicz, K. Wegner, An experimental free-view television system, in: 1st International Conference on Image Processing & Communications (IPC), Bydgoszcz, Poland, September 2009, pp. 175–184.
[23] M. Domański, T. Grajek, K. Klimaszewski, M. Kurc, O. Stankiewicz, J. Stankowski, K. Wegner, Poznań Multiview Video Test Sequences and Camera Parameters, ISO/IEC JTC1/SC29/WG11, MPEG/M17050, m17050, Xian, China, October 26–30, 2009.
[24] J.J. McConnell, Computer Graphics: Theory into Practice, Jones & Bartlett Learning, Jones and Bartlett Publishers, Inc, 2006, p. 120. ISBN: 0-7637-2250-2.
[25] C.D. Brown, Decentering distortion of lenses, Photogramm. Eng. 32 (3) (1966) 444–462.
[26] http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.
[27] Z. Zhang, A Flexible New Technique for Camera Calibration, MSR-TR-98-71, Microsoft Research, 1998.
[28] G. Lafruit, K. Wegner, M. Tanimoto, FTV Software Framework, ISO/IEC JTC1/SC29/WG11 MPEG2015/N15349, Poland, Warsaw, 2015.
[29] https://en.wikipedia.org/wiki/ICC_profile.
[30] J. Stankowski, K. Klimaszewski, O. Stankiewicz, K. Wegner, M. Domański, Preprocessing Methods Used for Poznan 3D/FTV Test Sequences, ISO/IEC JTC1/SC29/WG11 MPEG 2010/M17174, m17174, Kyoto, Japan, 2010.
[31] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: the KITTI dataset, Int. J. Rob. Res. 32 (11) (2013) 1231–1237.
[32] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vis. 47 (1) (2002) 7–42.
[33] M.D. Nguyen, Stereo Depth Estimation With Inter-View Consistencies (Master Thesis), Faculty of Science and Bio-Engineering Sciences, Department of Computer Science, Vrije Universiteit Brussel, 2015.
[34] R.A. Hamzah, H. Ibrahim, Literature survey on stereo vision disparity map algorithms, J. Sens. 2016 (2016) 23. http://www.hindawi.com/journals/js/2016/8742920/ (Article ID 8742920).
[35] M. Dumont, P. Goorts, S. Maesen, G. Lafruit, P. Bekaert, Real-time edge-sensitive local stereo matching with iterative disparity refinement, in: 11th International Joint Conference, ICETE 2014, Vienna, Austria, August 28–30, 2014, pp. 435–456.
[36] D.M. Nguyen, J. Hanca, S.-P. Lu, P. Schelkens, A. Munteanu, Accuracy and robustness evaluation in stereo matching, in: SPIE Optical Engineering and Applications, Proceedings vol. 9971, Applications of Digital Image Processing XXXIX, 2016. https://doi.org/10.1117/12.2236509.
[37] B. Ceulemans, M.D. Nguyen, G. Lafruit, A. Munteanu, New Depth Estimation and View Synthesis, ISO/IEC JTC1/SC29/WG11, MPEG2016/ M38062, San Diego, 2016.

[38] P. Tan, P. Monasse, Stereo disparity through cost aggregation with guided filter, Image Process. On Line 4 (2014) 252–275. https://doi.org/10.5201/ipol.2014.78.
[39] K. Zhang, J. Lu, G. Lafruit, Cross-based local stereo matching using orthogonal integral images, IEEE Trans. Circ. Syst. Video Technol. 19 (2009) 1073–1079.
[40] S.N. Sinha, Graph Cut Algorithms in Vision, Graphics and Machine Learning an Integrative Paper, Graph Cut Algorithms in Vision, Graphics and Machine Learning, Microsoft Technical Report MSR-TR-2004-152, 2004, https://www.microsoft.com/en-us/research/publication/248342/.
[41] V. Kolmogorov, R. Zabih, Graph cut algorithms for binocular stereo with occlusions, in: N. Paragios, Y. Chen, O. Faugeras (Eds.), Handbook of Mathematical Models in Computer Vision, Springer, New York, 2006.
[42] K. Wegner, O. Stankiewicz, M. Tanimoto, M. Domanski, Enhanced Depth Estimation Reference Software (DERS) for Free-Viewpoint Television, ISO/IEC JTC1/SC29/WG11, Doc. M31518, 2013.
[43] E.B. Sudderth, W.T. Freeman, Signal and image processing with belief propagation, IEEE Signal Process. Mag. 25 (2) (2008) 114–141.
[44] M.F. Tappen, W.T. Freeman, Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters, in: Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 2, October 2003, pp. 900–906.
[45] R. Collins, A space-sweep approach to true multi-image matching, in: Proceedings of the CVPR, 1996, pp. 358–363.
[46] P. Goorts, Real-Time, Adaptive Plane Sweeping for Free Viewpoint Navigation in Soccer Scenes (Ph.D. Thesis), Hasselt University, 2014.
[47] R.C. Bolles, H.H. Baker, D.H. Marimont, Epipolar-plane image analysis: an approach to determining structure from motion, Int. J. Comput. Vis. l (1987) 7–55.
[48] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, M. Gross, Scene reconstruction from high spatio-angular resolution light fields, in: ACM Transactions on Graphics (TOG) Conference Proceedings of SIGGRAPH, vol. 32(4), July 2013 (Article No. 73).
[49] L. Jorissen, P. Goorts, S. Rogmans, G. Lafruit, P. Bekaert, Multi-camera epipolar plane image feature detection for robust view synthesis, in: Proceedings of the 3DTV-Conference: The True Vision – Capture, Transmission and Display of 3D Video (3DTV-CON), 2015.
[50] L. Jorissen, P. Goorts, G. Lafruit, P. Bekaert, Multi-view wide baseline depth estimation robust to sparse input sampling, in: Proceedings of the 3DTV-Conference: The True Vision – Capture, Transmission and Display of 3D Video (3DTV-CON), 2016.
[51] E.E. Sutherland, R. Sproull, R.A. Schumacker, A characterization of ten hidden-surface algorithms, ACM Comput. Surv. 6 (1) (1974) 1–55.
[52] Break-Dancers and Ballet Sequence: http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/.
[53] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Trans. Graph. 23 (3) (2004) 600–608.
[54] A. Dziembowski, A. Grzelka, D. Mieloch, O. Stankiewicz, K. Wegner, M. Domański, Multiview synthesis – improved view synthesis for virtual navigation, in: Picture Coding Symposium, PCS 2016, Nuremberg, Germany, 2016.
[55] K. Wegner, O. Stankiewicz, M. Tanimoto, M. Domanski, Enhanced View Synthesis Reference Software (VSRS) for Free-Viewpoint Television, ISO/IEC JTC1/SC29/WG11, Doc. M31520, 2013.

[56] K. Wegner, O. Stankiewicz, M. Domański, Depth Based View Blending in View Synthesis Reference Software (VSRS), ISO/IEC JTC1/SC29/WG11 MPEG2015, M37232, Geneva, Switzerland, 2015.
[57] K. Wegner, O. Stankiewicz, M. Domański, Novel depth-based blending technique for improved virtual view synthesis, in: IEEE International Conference on Signals and Electronic Systems, Krakow, Poland, 2016.
[58] P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, T. Wiegand, Depth image-based rendering with advanced texture synthesis for 3-D video. IEEE Trans. Multimedia 13 (3) (2011) 453–465, https://doi.org/10.1109/TMM.2011.2128862.
[59] M. Tanimoto, T. Fujii, K. Suzuki, Reference Software of Depth Estimation and View Synthesis for FTV/3DV, ISO/IEC JTC1/SC29/WG11, MPEG2008/M15836, Busan, Korea, 2008.
[60] G. Lafruit, K. Wegner, T. Grajek, T. Senoh, K.P. Tamás, P. Goorts, L. Jorissen, B. Ceulemans, P.C. Lopez, Sergio García Lobo, Qing Wang, Joël Jung, Masayuki Tanimoto, FTV Software Framework, ISO/IEC JTC1/SC29/WG11 MPEG2014/N15349, Poland, Warsaw, 2015.
[61] I.E. Richardson, The H.264 Advanced Video Compression Standard, second ed., Wiley, Chichester, West Sussex, UK, 2010.
[62] V. Sze, M. Budagavi, G.J. Sullivan, High Efficiency Video Coding (HEVC), Algorithms and Architectures, Springer, Springer Heildelberg New York Dordrecht London, 2014.
[63] M. Wien, High Efficiency Video Coding, Coding Tools and Specification, Springer, Springer Cham Heildelberg New York Dordrecht London, 2015.
[64] I.E. Richardson, Coding Video: A Practical Guide to HEVC and Beyond, Wiley, Chichester, West Sussex, UK, 2016.
[65] Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC Int. Standard 13818-2: 2013 and ITU-T Rec. H.262 (V3.1), 2012.
[66] Coding of audio-visual objects, Part 10: Advanced Video Coding, ISO/IEC Int. Standard 14496-10: 2014 and Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 (V9) 2014.
[67] High Efficiency Coding and Media Delivery in Heterogeneous Environment: High Efficiency Video Coding, ISO/IEC Int. Standard 23008-2: 2015 and High Efficiency Video Coding, ITU-T Rec. H.265 (V3), 2015.
[68] K.R. Rao, Do Nyeon Kim, Jae Jeong Hwang, Video Coding Standards: AVS China, H.264/MPEG-4 Part10, HEVC, VP6, DIRAC and VC-1, in: Series Signals and Communication Technology, Springer, Netherlands, 2014. https://doi.org/10.1007/978-94-007-6742-3, ISBN: 9789400767416/9789400767423 (online).
[69] M. Domański, T. Grajek, D. Karwowski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, J. Stankowski, K. Wegner, Coding of multiple video+depth using HEVC technology and reduced representations of side views and depth maps, in: 29th Picture Coding Symposium, PCS 2012, Kraków, May 2012, pp. 5–8.
[70] M. Domański, A. Dziembowski, D. Mieloch, A. Łuczak, O. Stankiewicz, K. Wegner, A practical approach to acquisition and processing of free viewpoint video, in: 31st Picture Coding Symposium PCS 2015, Cairns, Australia, 31 May–3 June 2015, pp. 10–14.
[71] G. Davidson, M. Isnardi, L. Fielder, M. Goldman, C. Todd, ATSC video and audio coding, Proc. IEEE 94 (2006) 60–76.
[72] M. Domański, Approximate Video Bitrate Estimation for Television Services, ISO/IEC JTC1/SC29/WG11 Doc. MPEG M3671, Warsaw, June 2015.

[73] M. Domański, A. Dziembowski, T. Grajek, A. Grzelka, Ł. Kowalski, M. Kurc, A. Łuczak, D. Mieloch, R. Ratajczak, J. Samelak, O. Stankiewicz, J. Stankowski, K. Wegner, Methods of high efficiency compression for transmission of spatial representation of motion scenes, in: IEEE Int. Conf. Multimedia and Expo Workshops, Torino, 2015.
[74] B.G. Haskell, A. Puri, A.N. Netravali, Digital Video: An Introduction to MPEG-2, Chapman & Hall, New York, 1996.
[75] L. Torres, M. Kunt (Eds.), Video Coding, The Second Generation Approach, Kluwer, Boston, MA, 1996.
[76] J.-B. Lee, H. Kalva, The VC-1 and H.264 Video Compression Standards for Broadband Video Services, Springer, New York, 2008.
[77] Parameter values for ultra-high definition television systems for production and international programme exchange, Rec. ITU-R BT.2020-1, 2014.
[78] JPEG 2000 Image Coding System, Part 1: Core Coding System, second ed., ISO/IEC International Standard 15444-1, 2004 and ITU-T Rec. T.800, 2002.
[79] JPEG 2000 Image Coding System, Part 3: Motion JPEG 2000, ISO/IEC International Standard 15444-3, 2007 and ITU-T Rec. T.802, 2005.
[80] T. Acharya, P.-S. Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures, Wiley-Interscience, Hoboken, 2004.
[81] P. Schelkens, A. Skodras, T. Ebrahimi (Eds.), The JPEG 2000 Suite, Wiley, Chichester, 2009.
[82] D.S. Taubman, M.W. Marcelin, JPEG 2000 Image Compression Fundamentals, Standards and Practice, Kluwer, Boston, 2002.
[83] MPEG Video Technologies – Part 3: Representation of Auxiliary Video Streams and Supplemental Information, ISO/IEC Int. Standard 23002-3, 2007.
[84] K. Klimaszewski, O. Stankiewicz, K. Wegner, M. Domański, Quantization optimization in multiview plus depth video coding, in: IEEE International Conference on Image Processing ICIP 2014, Paris, France, October 27–30, 2014.
[85] Y. Morvan, D. Farin, P.H.N. de With, Platelet-based coding of depth maps for the transmission of multiview images, in: Proc. SPIE, Stereoscopic Displays and Applications, vol. 6055, San Jose, January 2006.
[86] P. Merkle, C. Bartnik, K. Müller, D. Marpe, T. Wiegand, 3D video: depth coding based on inter-component prediction of block partitions, in: 29th Picture Coding Symposium, PCS 2012, Kraków, May 2012, pp. 149–152.
[87] F. Jäger, Contour-based segmentation and coding for depth map compression, in: Proc. IEEE Visual Communications and Image Processing, Tainan, Taiwan, 2011.
[88] J.Y. Lee, H.W. Park, Efficient synthesis-based depth map coding in AVC-compatible 3D video coding, IEEE Trans. Circ. Syst. Video Technol. 26 (2016) 1107–1116.
[89] J.H. Hur, S. Cho, Y.L. Lee, Adaptive local illumination change compensation method for H.264/AVC-based multiview video coding, IEEE Trans. Circ. Syst. Video Technol. 17 (11) (2007) 1496–1505.
[90] A. Vetro, T. Wiegand, G.J. Sullivan, Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard, Proc. IEEE 99 (2011) 626–642.
[91] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, Y.-K. Wang, Overview of the multiview and 3D extensions of high efficiency video coding, IEEE Trans. Circ. Syst. Video Technol. 26 (1) (2016) 35–49.

[92] O. Stankiewicz, K. Wegner, M. Domański, Nonlinear depth representation for 3D video coding, in: IEEE International Conference on Image Processing ICIP 2013, Melbourne, Australia, September 15–18, 2013, pp. 1752–1756.
[93] Y. Chen, X. Zhao, L. Zhang, J.-W. Kang, Multiview and 3D video compression using neighboring block based disparity vector, IEEE Trans. Multimedia 18 (4) (2016) 576–589.
[94] M. Domański, O. Stankiewicz, K. Wegner, M. Kurc, J. Konieczny, J. Siast, J. Stankowski, R. Ratajczak, T. Grajek, High efficiency 3D video coding using new tools based on view synthesis, IEEE Trans. Image Process. 22 (9) (2013) 3517–3527.
[95] Y. Gao, G. Cheung, T. Maugey, P. Frossard, J. Liang, Encoder-driven inpainting strategy in multiview video compression, IEEE Trans. Image Process. 25 (2016) 134–149.
[96] K. Müller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F.H. Rhee, G. Tech, M. Winken, T. Wiegand, 3D high-efficiency video coding for multi-view video and depth data, IEEE Trans. Image Process. 22 (9) (2013) 3366–3378.
[97] F. Shao, W. Lin, G. Jiang, M. Yu, Low-complexity depth coding by depth sensitivity aware rate-distortion optimization, IEEE Trans. Broadcast. 62 (1) (2016) 94–102.
[98] M.M. Hannuksela, D. Rusanovskyy, W. Su, L. Chen, R. Li, P. Aflaki, D. Lan, M. Joachimiak, H. Li, M. Gabbouj, Multiview-video-plus-depth coding based on the advanced video coding standard, IEEE Trans. Image Process. 22 (9) (2013) 3449–3458.
[99] J. Stankowski, Ł. Kowalski, J. Samelak, M. Domański, T. Grajek, K. Wegner, 3D-HEVC extension for circular camera arrangements, in: 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 3DTV- Con 2015, Lisbon, Portugal, July 8–10, 2015.
[100] J. Samelak, J. Stankowski, M. Domański, Adaptation of the 3D-HEVC coding tools to arbitrary locations of cameras, in: International Conference on Signals and Electronic Systems, Kraków, 2016.
[101] D. Rusanovskyy, P. Aflaki, M.M. Hannuksela, UndoDancer 3DV Sequence for Purposes of 3DV Standardization, ISO/IEC JTC1/SC29/WG11, Doc. MPEG M20028, Geneva, Switzerland, 2011.
[102] M. Tanimoto, T. Fujii, N. Fukushima, 1D Parallel Test Sequences for MPEG-FTV, ISO/IEC JTC1/SC29/WG11, Doc. MPEG M15378, Archamps, France, 2008.
[103] Y.S. Ho, E.K. Lee, C. Lee, Multiview Video Test Sequence and Camera Parameters, ISO/IECJTC1/SC29/WG11 Doc. MPEG M15419, Archamps, France, 2008.
[104] M. Domański, A. Dziembowski, A. Kuehn, M. Kurc, A. Łuczak, D. Mieloch, J. Siast, O. Stankiewicz, K. Wegner, Poznan Blocks – A Multiview Video Test Sequence and Camera Parameters for Free Viewpoint Television, ISO/IECJTC1/SC29/WG11, Doc. MPEG M32243, San Jose, USA, 2014.
[105] M. Domański, A. Dziembowski, A. Kuehn, M. Kurc, A. Łuczak, D. Mieloch, J. Siast, O. Stankiewicz, K. Wegner, Experiments on acquisition and processing of video for free-viewpoint television, in: 3DTV Conference 2014, Budapest, Hungary, July 2–4, 2014.
[106] P. Kovacs, [FTV AHG] Big Buck Bunny light-field test sequences, ISO/IEC JTC1/SC29/WG11, Doc. MPEG M3571, Geneva, 2015.
[107] G. Bjøntegaard, Calculation of average PSNR differences between RD-curves, ITU-T Study Group 16 Question 6 (VCEG), Document VCEG-M33, Austin TX, 2001.

FURTHER READING

M. Domański, M. Gotfryd, K. Wegner, View synthesis for multiview video transmission, in: The 2009 International Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, USA, July 13–16, 2009, pp. 1–4.

K. Khoshelham, S.O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors (Basel) 12 (2) (2012) 1437–1454.