Real-Time 2D to 3D Conversion from H.264 Video Compression

Nir Shabat, Gil Shabat and Amir Averbuch

Abstract—We present a real-time procedure for an automatic conversion of 2D H.264 video into a 3D stereo. The proposed algorithm works with H.264 compressed videos by taking advantage of its advanced motion compensation (MC) features such as multiple references and quarter-pixel accuracy. The algorithm is based on incorporating our Depth-From-Motion-Compensation (DFMC) algorithm into H.264 decoding process. The DFMC algorithm uses the MC data in the compressed video to estimate a depth map for each video frame. These depth maps are then used to generate in real time a stereo video via a Depth-Image-Based-Rendering (DIBR).

Index Terms—depth estimation, 3d conversion, motion compensation, MPEG, H.264, real-time

1 Introduction

In recent years, we witness a sharp increase in the demand and availability of 3D capable devices such as TV sets and cellular phones. This rise is somewhat held back by the fact that the vast majority of video content available today is 2D video, and that 3D video recording in new productions is both expensive and complicated. In order to bridge the gap between the availability and the demand for 3D content, different stereo conversion techniques are being used for converting 2D video into 3D.

Most stereo conversion algorithms first estimate a depth map for each image in the video sequence, and then use depth-image-based rendering (DIBR) techniques to generate images for each eye. These algorithms try to facilitate different depth cues in the image sequence for inferring the depth for different areas in these images. When working with a single 2D image or a 2D video, where the view is monocular, the depth cues are sometimes referred to as *Monocular Depth Cues*. Some examples for monocular depth cues, which are used for depth estimation and perception, are structure [1], object familiarity [2], [3], [4], lightning [5], [6] and motion [7], [8], [9], [10], [11] (i.e. in video).

An important aspect for 2D to 3D stereo conversion algorithms lies in their performance and in their capability to perform the conversion in real-time. Using semi-automatic or computationally-intensive techniques in the process of depth map estimation can result in a high-quality 3D video, however, these techniques are not suitable for usage with either live broadcast or streaming content. It is evident that stereo conversion techniques for 3D capable TVs and cell phones must perform the stereo conversion in real-time.

N. Shabat, Google Tel Aviv, Electra Tower, 29th Floor, 98 Yigal Alon, Tel-Aviv, 6789141, Israel
A.Averbuch, G. Shabat, School of Computer Science, Tel Aviv University, Tel Aviv 69987, Israel

In this paper, we present a new algorithm for stereo convxersion of 2D video that is based on the H.264 video compression standard. We call this algorithm *Depth From Motion Compensation* (DFMC - see [12]). Similarly to many stereo conversions algorithms, the focus of our DFMC algorithm is in estimating a depth map to be subsequently used for stereo view synthesis.

The depth map estimation process of our algorithm is based on the Motion Compensation (MC) data, which is part of every H.264-based compressed video. The MC data, which is extracted during the video decoding process, is used to calculate the relative projected velocity of the elements in the scene. This velocity is used to estimate the relative depths of pixel patches in each picture of the video scene sequence. The depth estimation process is performed alongside the standard H.264 decoding process for increased efficiency, where the initial estimation of a patch depth is performed immediately after its decoding is completed.

In order to achieve spatial and temporal smoothness, we apply a joint bilateral filter on the estimated depth maps and then calculate a weighted average on the filtered result. The averaged depth maps are then used in a DIBR procedure to synthesize the resulting stereo images. We show how our stereo conversion procedure can be integrated into the normal flow of the H.264 decoding process. Figure 1.1 depicts the data flow diagram of a stereo conversion system that incorporates our DFMC algorithm.

We implemented our DFMC based stereo conversion procedure within the open source FFmpeg [13] H.264 decoder. We then applied it to several video sequences including nature scenes (“National Geographic - Amazing Flight Over the Grand Canyon”) and movie trailers (“Batman: The Dark Knight” and “X-Men: Days of Future Past”).

This paper has the following structure. Section 2 discusses related work on 2D to 3D conversion, focusing on depth-map-based procedures for stereo conversion. Sec-

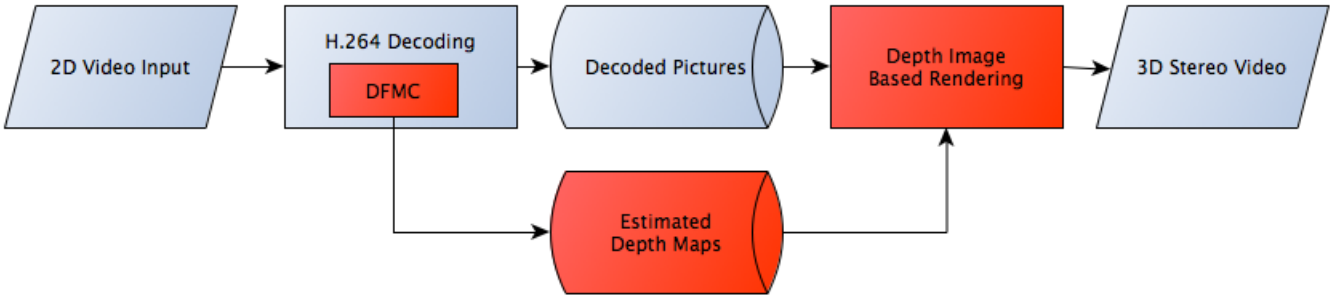


Fig. 1.1: Data flow diagram for the 2D to 3D stereo conversion procedure resulting from the incorporation of our DFMC algorithm into the H.264 decoding process. The blue elements are part of the H.264 standard decoding process, while the red elements are added to support the stereo conversion.

tion 3 describes necessary background in 2D to 3D stereo conversion such as DIBR, motion parallax and the H.264 video standard. Section 4 presents a detailed description of the DFMC algorithm. Finally, section 5 presents results (performance and quality) from the application of the algorithm to various video content.

2 Related Work

The area of 2D to 3D conversion (which bears many similarities to the area of 3D reconstruction), has been extensively studied over the years [14], [15], [16], [17]. The main task in the majority of 2D to 3D conversion techniques is in estimating the depth map for a given 2D image (or multiple depth maps for an image sequence). The depth map is generated by exploiting different depth cues in the 2D content. This depth map is then used to synthesize the views from the left and right eyes (or alternatively, to synthesize the view from one eye, using the source image as the view from the other). View synthesis is performed by employing depth-image-based rendering (DIBR) techniques.

There are many approaches for estimating the depth map of an image. Semi-automatic methods require the user to provide partial depth or disparity information, usually by marking scribbles on a few key frames [18], [19], [20]. This partial depth information is then propagated to the entire video scene by different methods. Another approach uses a shifted bilateral filter, which is a form of bilateral filtering, that takes motion into consideration [19]. A different path is taken in [18], where a SVM classifier (trained on the partial data) is employed to define constraints for an optimization problem. This optimization problem is then solved as a sparse linear system. Tracking is used in [21] to propagate objects' depth from key frames into other frames.

Another approach uses the structural and the geometric cues of an image to infer its depth map. The absolute mean depth of a scene is estimated in [1] by using features that are based on local and global spectral signatures in the image. In [22], the image is first classified into one of three structural categories (indoor, outdoor with geometric elements and outdoor without geometric elements), followed by the detection of vanishing points, before the actual depth is assigned. A two-phase approach is used in [23] where in the first phase different parts of the image are labeled semantically (with labels such as ground, sky, building, etc..). In the second phase, the depth of each pixel (or super-pixel) is estimated in relation to the label given to it in the first phase.

Recent methods approach the depth estimation problem from a machine learning point of view. A Markov Random Field (MRF) is used in [24], [25], [26], [27], [28] to model the relationship between the depth of an image patch and the depth of its neighboring patches. The MRF is trained on a database of images and their corresponding ground-truth depth maps. A data-driven approach is taken in [29] where a database of image-plus-depth pairs is employed. The k-nearest neighbors (k-NN) of the query image are found in the database. The Euclidean norm of the difference between histograms of oriented gradients (HOG) [30] is used as the distance measure. Then, the depth maps of these k neighbors are fused by a median filter to a single depth map. The resulting depth map is then smoothed by the application of a bilateral filter. A similar approach is described in [31], where the k-nearest candidate images are selected based on a combination of GIST [32] features and features derived from a calculated optical flow. The SIFT flows [33], which are computed from the input image to each of the k image candidates, are used in wrapping each of the candidates' depth maps. Finally, an objective function is defined for fusing the warped depth maps. This function is minimized using an iteratively reweighted least squares (IRLS) procedure. A supervised learning approach, which uses the random forest method, is applied in [34] to depth map estimation. The model is trained on feature vectors, which are based on a variety of depth cues, in order to get good results for unconstrained input videos.

Motion is often used when inferring the depth of an image sequence. A popular approach is to use the technique of structure from motion (SFM) [35], [36]. In SFM, a correspondence between subsequent video frames is

established. Then, it is used to compute the camera motion and the 3D coordinates of the matched image points. A SFM procedure is used in [7] to estimate the depth of feature points arriving at a sparse depth map. To achieve a dense depth map, a Delaunay triangulation is generated for each input image. Then, depth is assigned to a pixel of each triangle according to the depth of feature points covered by the triangle. SFM techniques are used in [37], [38] to estimate the camera parameters of each frame in the video. These camera parameters are then used in an energy function to enforce photo-consistency. Subsequently, loopy belief propagation [39] is employed in minimizing the energy function thus generating the initial depth map estimation. This initial estimation is then improved iteratively by minimizing another energy function using an optimization framework called bundle optimization. The colors of the input image are used in [9], [10] to enhance the motion estimated depth maps. A two-phase approach is employed by [9]. First, the pixels of the image are clustered based on their color using k-means, then, the clustering is refined using motion and edge information, where the pixels are classified into near and far categories. A coarse depth map is estimated in [10] by employing a block-matching algorithm for calculating the motion vectors (MVs) between an image and its reference image. The scaled size of each calculated MV is used as the depth of its block in the coarse depth map, which is later refined by utilizing image color.

In order to achieve a real-time performance, recent methods [11], [40], [41], [42], [43] utilize the MC data in videos compressed by either MPEG-2 or H.264/MPEG-4 AVC standards [44]. The scaled size of each MV is used as the depth of its associated macro block (MB) in [40] as an initial depth map, which is then smoothed using a nearest-neighbor method. The horizontal component of each MV is used in [41] as an initial depth estimation, which is then applied several corrections based on the camera's motion, displacement errors and object borders. The size of MVs is similarly used in [11] for a coarse depth map estimation, which is then refined by using a Gaussian mixture model (GMM) [45] to detect the moving objects. This motion based depth map is then fused with a geometric based depth map to produce the final result.

Our suggested procedure utilizes the idea of using motion information stored by motion-compensated-based video formats to builds upon it. We use the MC data of a frame not only to estimate depth within the frame, but also to estimate the depth in the referenced frames. The use of this "backward" prediction allows for a better depth estimation, since it allows depth estimation for intra-predicted pixel partitions. Our procedure also takes advantage of H.264 advanced features such as quarter-pixel precision of MV and motion predicted partitions as small as 4×4 pixels. We describe a complete framework for incorporating our stereo conversion procedure into an H.264 decoder that supports real-time 2D to 3D stereo conversion.

Fig. 3.1: An example of a color image (top) and its corresponding depth map (bottom). Areas in the color image, which are closer to the camera, appear brighter in its depth map

3 Stereo Conversion

3.1 3D Stereo Video

We usually view the world from two viewing angles due to the separation of the eyes. These two different viewing angles result in slightly different images seen by the left and right eye. This difference in images, and specifically the difference in the locations of objects between them, is called binocular disparity. Binocular disparity is used by the brain to extract depth information from these two images. The depth perceived by the brain due to this difference is called stereopsis. The vast majority of 3D techniques exploit stereopsis for creating the illusion of depth on 2D displays (such as TV sets) by presenting a different image to each eye. The use of stereopsis for this purpose is called stereoscopy. In order to create a perceived depth illusion for a 2D monocular image using stereopsis, one needs to generate two different images from the 2D image, which represent the two different viewing angles of the eyes.

3.2 Depth Maps

A depth map (or depth image) of an image is another image in which the intensity (or color) of each pixel represents the distance (or the depth) from a viewpoint of the corresponding pixel in the first image (see Fig. 3.1). Depth maps can be acquired directly by using tools such as a laser scanner, or be generated synthetically from

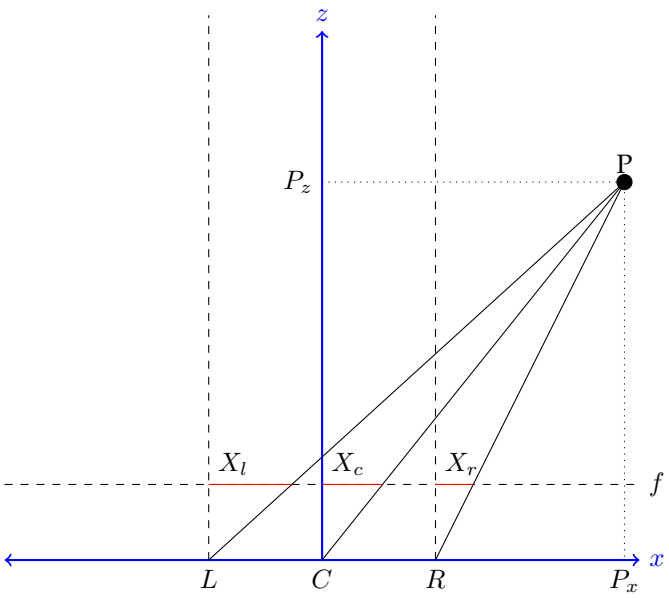


Fig. 3.2: The disparity of a point P between images taken from a camera with focal length f at points L, C and R

ordinary 2D content (from one or more viewpoints). Synthetic depth maps may either be produced in a manual process, or by an automatic (or semi-automatic) procedure. Depth maps have numerous applications, one of which is rendering a scene from different view points in a process which is called *Depth-Image-Based Rendering* (DIBR), as explained in section 3.3.

3.3 Depth-Image-Based Rendering (DIBR)

DIBR techniques use an image depth map to synthesize different images, pertaining to different viewing angles of a scene, from a single image of that scene. In 2D to 3D stereo conversions algorithms, these techniques are used to generate images for the left and right eyes from a monocular image and its depth map. Conceptually, DIBR is a two step process. Initially, the image is projected into 3D coordinates using the information of its depth map. Then, the 3D image is reprojected onto the 2D plane of a virtual camera. This process of projecting and reprojecting the image is called *image warping*. For the case of stereoscopic image creation, in which the virtual cameras are simple translations of the actual camera, image warping takes a simpler form.

Let P be a 3D point with coordinates (P_x, P_y, P_z), then we have (see Fig. 3.2):

$$X_l = \frac{f \cdot (P_x - L)}{P_z}, X_c = \frac{P_x f}{A_z}, X_r = \frac{f \cdot (P_x - R)}{P_z}$$

where f is the focal length of the cameras, L and R are the horizontal coordinates of the left and right eyes "virtual" cameras, respectively, and X_c, X_l, X_r are the horizontal distances between the cameras' centers and the projection of the point P.

Denoting $|L| = |R| = \frac{t_c}{2}$, where t_c is the interaxial distance (the horizontal distance between the two view ports), the disparity for the left and right images becomes

$$\Delta s = \pm \frac{t_c f}{2P_z}. \tag{3.1}$$

Given the focal length f, the interaxial distance t_c and a dense depth map (giving P_z at each pixel), we can use Eq. 3.1 to calculate the disparity of each pixel between the input image and the images for the left and right eyes needed for stereopsis. In the case of a viewer, watching a video on a 2D display, f is essentially the viewer's distance from the screen. t_c is usually chosen as the distance in humans between the eyes, which on average is 64mm.

3.4 Motion Parallax

Parallax is defined as the displacement or difference in the apparent position of an object viewed along two different lines of sight. Objects, which are closer to the viewer, have larger parallax than objects which are further away. This fact lets us use parallax in order to estimate objects' distance.

Motion parallax is the parallax created due to the movement of the viewer. Since objects, which are closer to the viewer, have larger parallax during the movement of the viewer, they appear to move faster than objects which are more distant (see Fig. 3.3).

In a monocular image sequence (e.g. video), this parallax can be used as a depth cue for estimating the relative pixels' depth. Consider a camera moving horizontally from a point C to a point R in Fig. 3.2. The disparity of point P between the two images, taken by the camera at points C and L, is given by Eq. 3.1. According to Eq. 3.1, bigger values of P_z will decrease the disparity Δs and vice versa. This means that we can use 3.1 to infer the relative pixels' depths from their disparity between the two images.

In order to use the disparity of a pixel between two images (taken during camera movement, i.e. from two different viewing angles) we need to match the pixel between those two images. This parallax can be used to estimate the relative depth of that pixel using Eq. 3.1.

3.5 Motion Compensation-Based Video

Modern video compression formats use an algorithmic technique called *Motion Compensation* to utilize the temporal redundancy in videos. These formats can describe (or predict) video frames in terms of other video frames, which are called reference frames, or just references (in this work we use the term frame loosely, referring to both full frames and frame fields). The references of a frame can either precede it in time, or belong to a future time (presentation-wise). The description of video frames in terms of other frames is called inter frame prediction, or inter-prediction, for short, and these frames are called inter predicted frames, or inter-frames. In contrast, frames, which are not described in terms of

Fig. 3.3: Images taken at constant time intervals from a camera moving at constant speed from right to left. The green box, which is closer to the camera, appears to be moving faster than the distant red box.

other frames, and for which decoding can be performed by using only information contained within them (i.e. without any reference frames), are called intra predicted frames, or intra-frames, for short. A series of subsequent frames, which can be decoded independently of the rest of the video, is called a *Group-of-Pictures* (GOP).

In order to describe a frame in terms of other frames, the frame is split into pixel blocks called *Macro Block*s (MB). These MBs can be further split into smaller blocks called *MB Partition*s. Each MB or MB partition of an inter-frame can be associated with (one or more) vectors called *Motion Vector*s (MVs). Each MV points to a location in a reference frame which is used to describe the MB (or MB partition) it is associated with. This allows the video encoder to only store the difference (or the error) between the inter-predicted MB (or MB partition) and its references, instead of the entire MB (or MB partition).

Video compression standards, employing MC techniques, usually use three types of frames: *I*-frames, *P*-frames and *B*-frames according to the prediction type used to encode the frame. *I*-frames are intra frames, *P*-frames are inter frames, which can reference other frames that precede them in time (i.e. backward references), and *B*-frames are inter frames, which can reference frames that both precede them or follow them in time (i.e. backward and forward references).

In order for a decoder to decode an inter-frame i, it needs to have access to all the frames referenced by i (in their decoded form). To achieve this, all frames referenced by the inter-frame i must arrive at the decoder prior to the arrival of i itself. The order by which the frames arrive at the decoder is called *Decoding Order* (see Fig. 3.5). The decoder places decoded frames in a buffer called the *Decoded Picture Buffer* (DPB). Reference frames are kept in the DPB by the decoder for as long as required in favor of inter prediction. The order by which the frames should be displayed (i.e. presented) is called *Presentation Order*, and it is usually different than the decoding order (see Fig. 3.4). In the MPEG2 and H.264 video standards, each video frame has two 90KHz time stamps associated with it. One time stamp indicates the time when the frame should be decoded, called *Decoding Time Stamp* (DTS), and one which indicates the time when the frame should be displayed, called *Presentation Time Stamp* (PTS).

In order for encoders of motion-compensated-based video standards to perform motion estimation by finding redundant image information *Block-Matching Algorithm*s (BMA) are employed. BMAs are used to find a match for a block of pixels (MB or MB partition) of a frame i in another frame j. The match is chosen based on some error metric, such as the mean squared error (MSE) or the sum of absolute differences (SAD). BMAs are computationally intensive algorithms, especially when performed on videos with high resolution (e.g. HD videos) or frame rate.

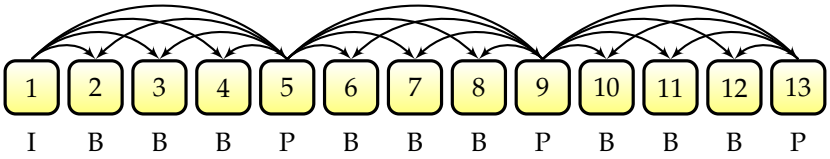


Fig. 3.4: A GOP of 13 frames in presentation order. The letter below each box denotes the frame's type and the number within each box denotes the presentation order count (POC). The arrows denote references where the arrow head points to the referencing frame and tail to the referenced frame.

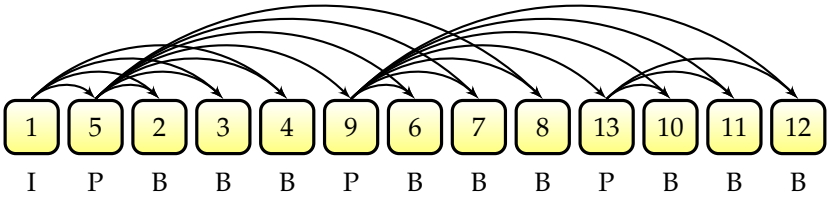


Fig. 3.5: Same GOP in Fig. 3.4, but in decoding order. Note that in decoding order, frames only reference frames which precede them.

3.6 The H.264/MPEG-4 AVC Video Compression Standard

H.264/MPEG-4 AVC is an advanced block-oriented motion-compensated-based video compression format [44], and the successor of the popular MPEG-2 format. H.264/MPEG-4 AVC offers several advancements over older video compression formats. Specifically, in regards to motion compensation (MC), it offers the following new features:

- Allowing up to 16 reference frames per encoded frame
- Allowing B-frames to be used as reference frames
- Variable block sizes, with blocks as big as 16×16 and as small as 4×4 samples
- The ability to use multiple MVs per MB
- Quarter-pixel precision for motion estimation

H.264 adds a type of intra-coded frame called an *Instantaneous Decoding Refresh* (IDR) frame. IDR frames are found at the beginning of each (closed) GOP. IDR frames signal that none of the pictures preceding them is needed (i.e. used as reference) for decoding subsequent pictures in the stream. These new enhancements of H.264 allow our proposed DFMC algorithm to perform the depth map estimation in a more accurate and comprehensive way than was possible with previous standards.

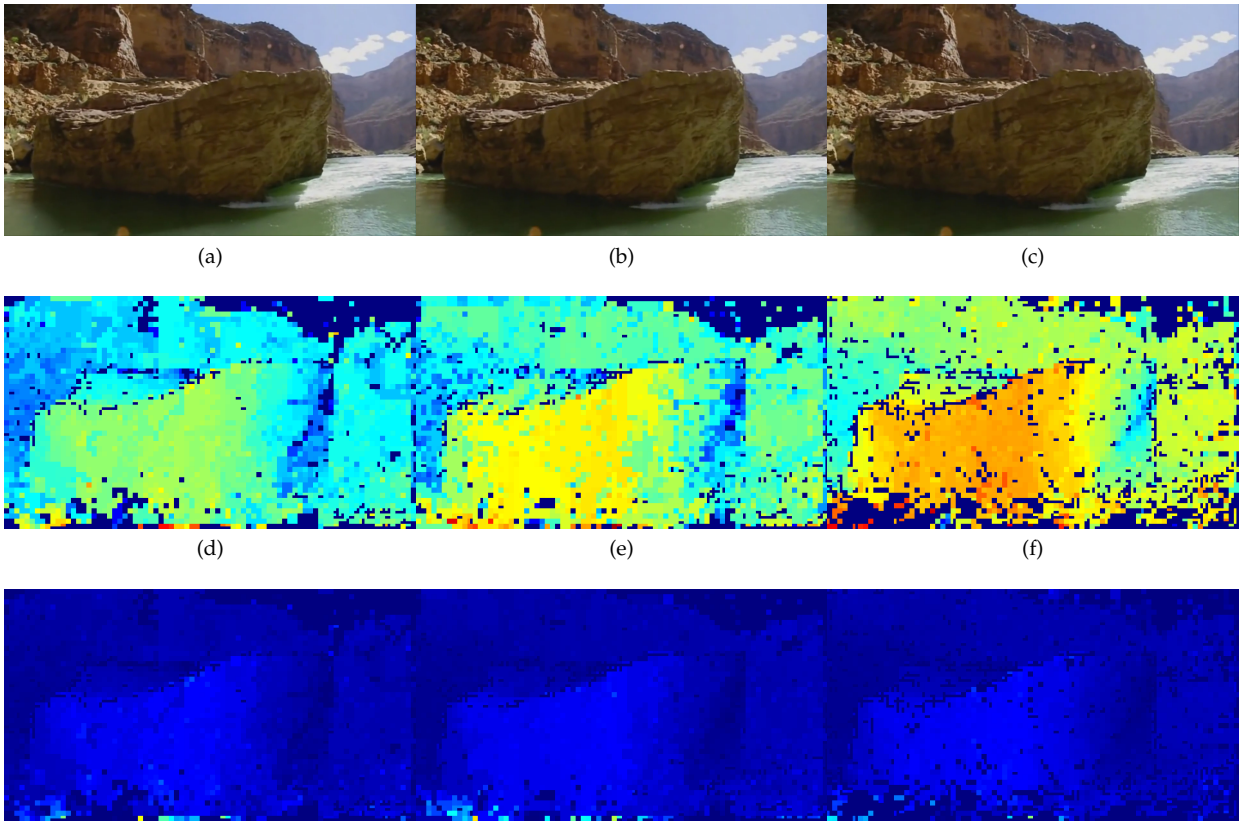
4 The Depth From Motion Compensation (DFMC) Algorithm

Our algorithm primarily uses motion to estimate a depth map for each video frame. The basic idea behind using motion to estimate depth is based on the fact that the projected object velocity on the screen is correlated with its relative distance from the camera. This is due to the fact that distant objects have small parallax in comparison to closer ones, see Fig. 3.3. The main objective of the algorithm is to produce dense depth maps which are suitable for 2D to 3D stereo conversion (e.g. via DIBR) in a computationally efficient way.

Since motion estimation is a computationally intensive process, we use the motion-compensating data that is extracted from the compressed video in the H.264 decoding stage. This makes it possible for the DFMC algorithm to be implemented on the decoder side, and to enable it to perform 2D to 3D video conversion in real-time. Moreover, our algorithm works with H.264 compressed videos, while taking advantage of its advanced MC features such as multiple references and quarter-pixel accuracy (see section 3.6). The algorithm in 4.1 describes the steps of our DFMC system and how it is incorporated into H.264.

4.1 Initial (Forward) Depth Estimation

Initial, also called forward, depth map estimation is achieved by assigning a depth value to each inter-predicted MB partition based on its associated MVs. Forward depth estimation for a MB partition is performed during the decoding process of the partition (see line 7 in Algorithm 4.1), since at this point we have immediate access to the partition's MVs, size and location. Similar to what was done in the related work by [11], [40], [41], [42], [43], we use the magnitude of the MVs to estimate the relative depth of an image pixel block. However, instead of directly taking the scaled size of each MV as the depth of a block, we normalize it with the difference in presentation time between the current frame and the frame referenced by that MV. We apply this normalization in order to account for a greater relative disparity of a frame's pixel block in comparison to other pixel blocks. This is due to the reference frame of the block's MV being further away in presentation time. This normalization is applied under

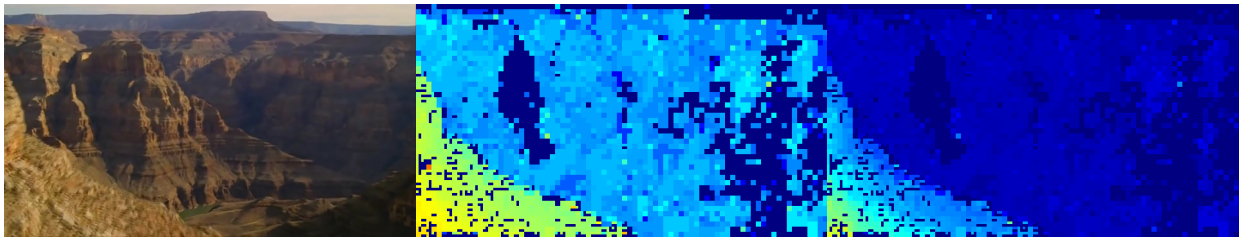


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 4.1: A sequence of three consecutive video frames from a National Geographic video titled "Amazing Flight Over The Grand Canyon" (top row), with their respective depth maps as produced by the first step of the DFMC algorithm (middle and bottom rows). The images in the second (middle) row are the frames' depth maps without time normalization. The images in the third (bottom) row are the time-normalized depth maps.



(a) (b) (c)

Fig. 4.2: A video frame from a National Geographic video of a flight over The Grand Canyon and its respective depth maps. The second image (middle) is the frame's depth map without time normalization. The third image (right), is the frame's time-normalized depth map.

the assumption that the camera's displacement between two consecutive video frames, which belong to the same video scene, is approximately constant. Figure 4.1 shows the resulting depth maps of 3 consecutive video frames with and without time normalization. One can see that the time-normalized depth maps are much more temporally consistent than their unnormalized counterparts. Figure 4.2 shows that the time-normalized depth map is more spatially consistent than its unnormalized version.

Denoting by $s\left(P\right)$ the time-normalized size of the MVs associated with the pixel block (MB or MB partition) of pixel P, then

$$s\left(P\right) = \frac{\sqrt{\mathrm{mv}_x^2 + \mathrm{mv}_y^2}}{\mathrm{mv}_{\Delta t}}$$

where mv_x and mv_y are the horizontal and vertical components, respectively, of the MVs associated with the block of pixel P, and $\mathrm{mv}_{\Delta t}$ is the difference in presentation time between the current frame and the reference frame calculated using the frames' PTS.

Algorithm 4.1: 2D to 3D conversion process with DFMC
Input: Next video frame F of video stream
Output: Side-by-side 3D version of input frame

```
1:  D ← Empty depth map
2:  for M ← macroblocks of F do
3:    if M is inter-predicted then
4:      for P ← pixel partitions of M do
5:        V ← MV of partition P
6:        RD ← depth maps of reference frames
7:        Update depth of P in D based on V
8:        Update depth of P references in RD based
          on V
9:      end for
10:   end if
11:   Decode M
12:   Add F to DPB
13:   Add D to DMB
14:   N ← next frame to display
15:   if N is an IDR frame then
16:     while DDB is not empty do
17:       F ← next frame in DDB
18:       D ← next DM in DMB
19:       Apply spatial hole filling, cross bilateral
          filtering and temporal smoothing to D
20:       L ← DIBR of left-eye image from F and D
21:       Output side-by-side video frame using L
          and F
22:     end while
23:   end if
24:   Add N to DDB
25: end for
```

To account for times where the camera is at a standstill, we define a lower bound $T_{\min}$. When the value of $s(P)$ is below $T_{\min}$, the depth of pixel P is discarded. In order to disregard MVs, which refer to partitions not belonging to the same object, we also define an upper bound $T_{\max}$, where $T_{\max} > T_{\min}$. When the value of $s(P)$ is above $T_{\max}$, we discard the estimation for pixel P.

Since H.264 supports weighted prediction (where a block is predicted using more than one reference block), our algorithm uses the MV, which references a frame that is closer in presentation time (according to the PTS), for the depth estimated frame.

The initial depth estimation for a pixel P, where $s(P)$ is within the bounds T_{min} and T_{max}, is given by

$$P_z = \text{depth}(P, c) = \max\left(\text{round}\left(\frac{s(P)}{c}\right), 1.0\right) \quad (4.1)$$

where $c > 0$ is a scaling factor, which is chosen empirically for the entire shot, or changed dynamically during the estimation process. In our experiments, we chose c dynamically for each GOP, so that the maximum estimated depth in that GOP is 1, i.e.,

$$c = \arg\max_{x} \max_{F \in G, P \in F} s(P)$$

where $F \in G$ denotes the frame F of a GOP G and $P \in F$ denotes the pixel P of frame F. Note that in order to dynamically calculate c for each GOP, we need to delay the output of our decoder by one GOP. In our experiments, we chose $T_{\min} = c/10$ and $T_{\max} = 30$ as lower and upper bounds, respectively.

4.2 Backward Depth Estimation

The initial depth map, estimated in the first step (see line 7 in Algorithm 4.1), provides a very partial depth estimation, since it only estimates the depth of pixels belonging to inter-predicted blocks. It does not provide estimation for pixels belonging to intra-predicted blocks since these do not have any MV associated with them. For example, I frames are strictly intra-coded and so every pixel in an I frame belongs to an intra-predicted MB or MB partition that has no associated MV. Since we want to use the precalculated MC data encoded in the video as much as possible when estimating pixels' depth, our algorithm also uses the MVs in a "backward" type, by using their size to assign depth to pixels belonging to the referenced blocks (see Fig. 4.3). This step is also performed during the decoding process of the MB partition (see line 7 in Algorithm 4.1), since at this point we can also quickly access the reference's depth map. The same pixel can be referenced several times by belonging to several referenced blocks or by belonging to a block being referenced multiple times. For such pixels, we use the MV that belongs to the closest (presentation-time-wise) referencing frame for that pixel. We then estimate its depth by using Eq. 4.1. This allows our DFMC Algorithm 4.1 to estimate the depth of many pixels that are not inter-predicted. Figure 4.4 shows that using backward estimation results in higher quality depth maps, which are more complete and less noisy, than the depth maps generated by using only forward estimation.

In order to incorporate this backward depth estimation in the decoder, it stores the partial depth map of each decoded frame in the DPB as long as that frame is still marked as a reference. Only when a frame is no longer used as a reference, the decoder outputs its depth map to the *Depth Map Buffer* (DMB). Since a frame can be fully decoded prior to the completion of its depth map estimation, we maintain a buffer of decoded frames called the *Depth Delay Buffer* (DDB). The DMB and DDB buffers are used to synchronize the depth estimation process with the decoding process. After the completion of a frame decoding process, we check if the next frame to display (according to the DDB) has a corresponding depth map in the DMB. If it does, the DIBR process uses the frame and its depth map to synthesize the stereo images (see Algorithm 4.1).

Note that in order to perform this backward estimation by the decoder, it must delay the output by at most one

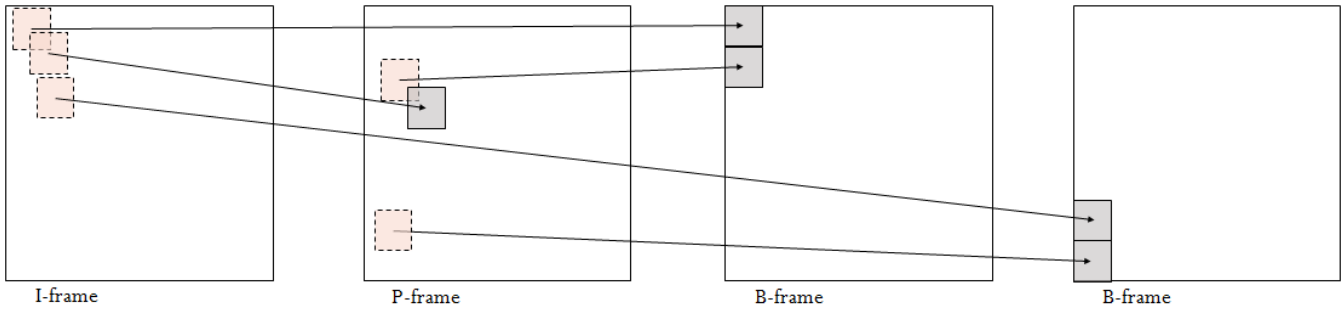


Fig. 4.3: The depth of pixels belonging to the I frame in the figure (dashed rectangles) is estimated using the MVs of the referencing P and B frames. The depth of intra-predicted pixels of the P frame in the figure are predicted using the MVs of the referencing B frames.

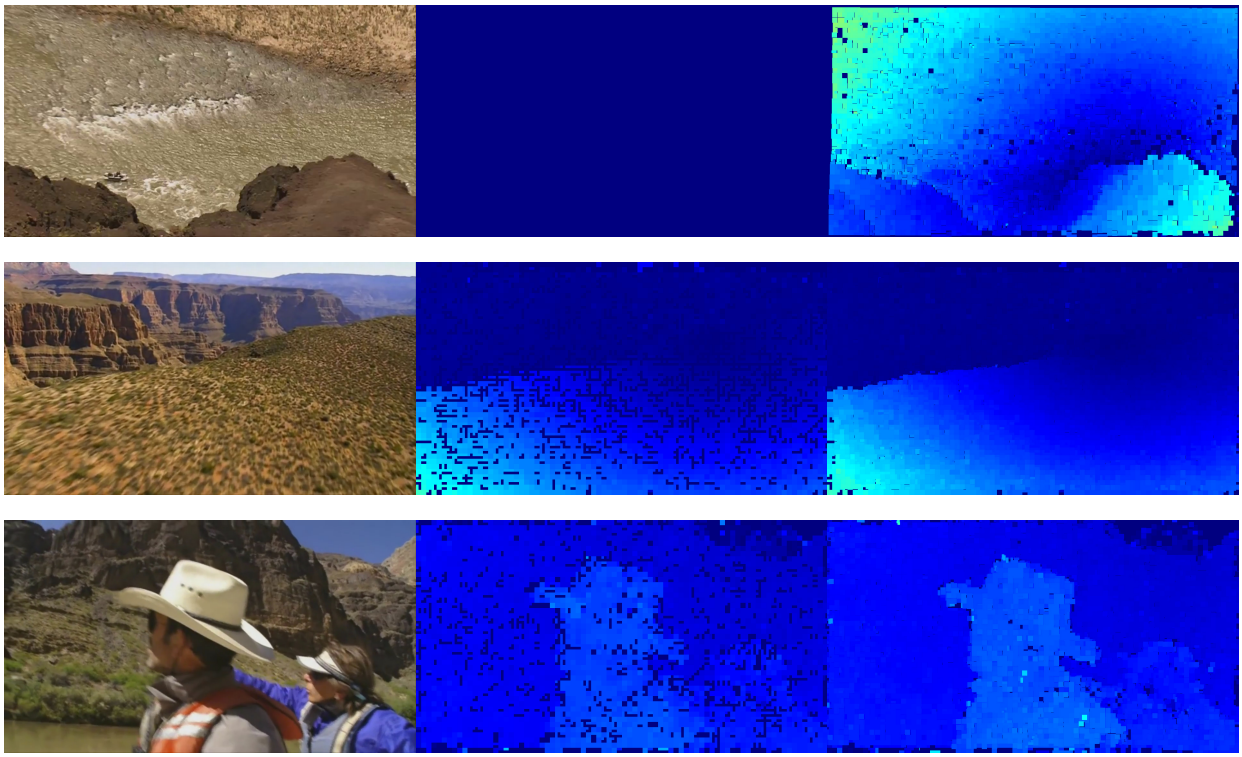


Fig. 4.4: Three frames from the National Geographic video titled "Amazing Flight Over The Grand Canyon". The first (leftmost) image in each row is the source frame, the second (middle) image is the frame's depth map using only forward estimation and the third (rightmost) image in each row is frame's depth map estimated using both forward and backward estimations. Note that the frame in the first (top) row is an I frame, therefore its forward estimated depth map is empty.

GOP in order to keep updating the depth map of the first frame in the GOP as new frames arrive into the DPB. In most cases, when the video is a broadcast stream, this does not pose a problem, since broadcast standards dictate that IDR pictures cannot be too far apart. For example, the ATSC Standard [46] requires AVC transport streams to contain at least one IDR frame every second from a video.

4.3 Depth Hole Filling

In the previous steps (forward and backward depth estimation), we discarded the estimated depth for pixels belonging to MB partitions with MV sizes lower than some predefined bound $T_{\min}$. In this step (see line 18 in Algorithm 4.1), we set the depth of these pixels as being equal to the depth of the same pixels in the preceding frame in presentation order. The assumption here is that when the camera or the objects, which these pixels belong to, are not moving, their depth should be the same as in the preceding frame. This step is performed

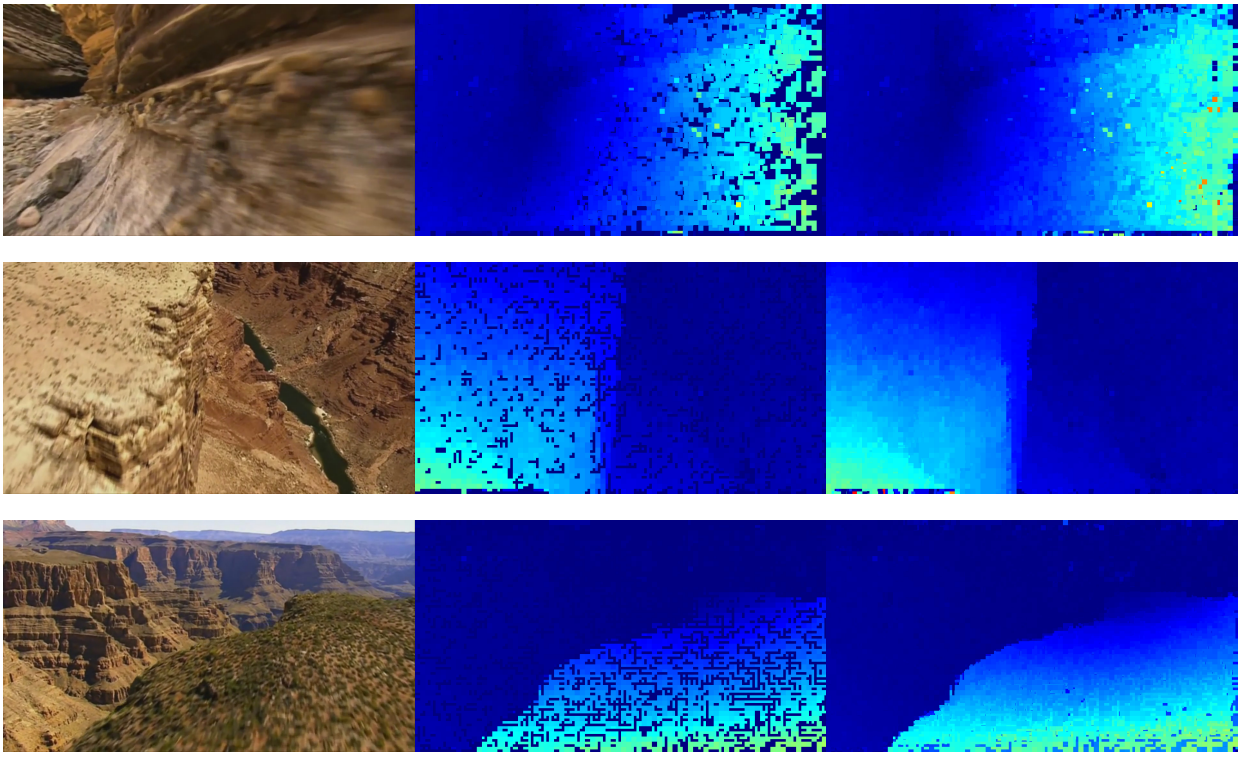


Fig. 4.5: Three frames from a National Geographic video of a flight over The Grand Canyon. The first (leftmost) image in each row is the source frame, the second (middle) image is the frame's depth map without the hole-filling step and the third (rightmost) image in each row is the frame's depth map after the application of the hole-filling step.

right before spatial and temporal smoothing. Figure 4.5 provides three examples of depth maps before and after the hole filling step. The hole-filled depth maps are less noisy than the original depth maps.

4.4 Temporal and Spatial Smoothness

It is important for a video scene's depth to have both spatial and temporal consistency, otherwise, the resulting 3D video will create discomfort for the viewer. Processing the depth maps for spatial and temporal consistency constitutes the final step prior to DIBR (see line 19 in Algorithm 4.1). To achieve spatial smoothness, our algorithm applies cross bilateral filtering [47] to the depth map, with a Gaussian based weight distribution over the RGB image. Formally,

$$D^{\mathrm{s}}(P) = \frac{1}{W(P)} \sum_{Q \in N(P)} G_{\sigma_s}(\|P-Q\|)\, G_{\sigma_r}(|I(P)-I(Q)|)\, D(Q)$$

where

$$W(P) = \sum_{Q \in N(P)} G_{\sigma_s}(\|P-Q\|)\, G_{\sigma_r}(|I(P)-I(Q)|),$$

and D^{s} is the spatially smoothed depth value of pixel P. D and I denote the depth map and the RGB image, respectively, $N(P)$ denotes the neighbors of pixel P, and G_{σ_s} and G_{σ_r} denote Gaussian distributions with an STD of σ_s and σ_r, respectively. The distance in values between two image pixels is defined as the Euclidean difference in RGB coordinates. Formally

$$|I(P)-I(Q)| = \sqrt{(P_r-Q_r)^2+(P_g-Q_g)^2+(P_b-Q_b)^2}.$$

In contrast to Gaussian filtering, bilateral filtering is edge preserving. Consequentially, it will not smooth depth values of edges in the image. Moreover, we take advantage of the color information by cross filtering the depth map with the input image. In our experiments, we used a kernel filter of size 5×5, $\sigma_s = 3$ and $\sigma_r = 0.15$.

For temporal smoothness, a weighted average scheme was used. The temporally smoothed depth map of the ith frame S_i is computed by

$$D_i^{\mathrm{s,t}} = \frac{2}{M(M+1)} \left(M \cdot D_i^{\mathrm{s}} + \sum_{j=1}^{M-1} (M-j) \cdot D_{i-j}^{\mathrm{s}} \right)$$

where $D_i^{\mathrm{s,t}}$ is the temporally-spatially smoothed depth map of frame i, D_i^{s} is the spatially smoothed depth map of frame i and M is the number of frames to use for temporal smoothing. We used $M = 3$ in our experiments.

4.5 Stereo Image Synthesis

Instead of using the source image and its depth map to synthesize both the left eye image and the right eye

image, we use the source image as the right eye image, and only synthesize an image for the left eye. One benefit of this approach is that generating one syntactic image requires less computations than generating two syntactic images. Moreover, it turns out that approximately two-thirds of the population is right-eye dominant [48], [49], [50], [51] meaning they have a tendency to prefer visual input from the right eye, which is another benefit for using a non-synthetic image as the right eye image.

To synthesize the left eye image from the source image and its depth map we employ a DIBR scheme. The value of a pixel P in the generated image is set to the value of a pixel Q in the source image that resides in the horizontally shifted coordinates of P. The number of shifted pixels to apply is a function of the depth of pixel P (based on Eq. 3.1). We denote by $x^\star$ the horizontal coordinate of the pixel which is the source pixel of (x, y) in the synthesized image. Then:

$$x^\star = \min\left(\text{round}\left(x - k \cdot D^{\text{s,t}}\left(x, y\right)\right), 0\right)$$

where $k > 0$ is a positive constant that depends on the viewer's distance from the display, interaxial distance and scene content, which is chosen empirically. The value of a pixel at coordinates (x, y) in the left eye image is thus given by $L\left(x, y\right) = I\left(x^\star, y\right)$.

5 Experimental Results

We implemented our DFMC algorithm and our DIBR procedure as part of the FFmpeg H.264 video decoder. FFmpeg [13] is a popular open source project which produces libraries for the decoding and encoding of multimedia data. We then applied our stereo conversion via the modified decoder to a variety of H.264 broadcast videos. Using a standard MacBook Pro with a 2 GB quad-core i7 Intel processor with 16 GB of RAM, we were able to perform the stereo conversions of each video stream in real-time. Figures 5.1 to 5.3 show a collection of video frames along with their corresponding depth maps as generated by the DFMC algorithm.

5.1 More Videos to Watch

Side-by-Side 3D video files generated by Algorithm 4.1 can be downloaded from our website at http://sites.google.com/site/nirnirs/. Watching this kind of videos requires a device which supports the display of SbS (Side-by-Side) video, such as a 3D television (3DTV). Some devices require that the viewer wear special glasses (e.g. shuttered or polarized glasses) when watching SbS videos.

Conclusion

We described a low complexity motion based algorithm for estimating depth maps for video frames. We also described a low complexity DIBR procedure for the synthesis of stereo images from these depth maps. We presented a complete framework for incorporating the depth estimation algorithm and the DIBR procedure into the standard H.264 decoding process. Finally, we showed that the resulting stereo conversion process is suitable for real-time 2D to 3D conversion of H.264-based broadcast content.

Acknowledgment

This research was partially supported by the Israeli Ministry of Science & Technology (Grants No. 3-9096, 3-10898) and by US - Israel Binational Science Foundation (BSF 2012282).

References

[1] A. Torralba and A. Oliva, "Depth estimation from image structure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 9, pp. 1226–1238, 2002.

[2] C. B. Hochberg and J. E. Hochberg, "Familiar size and the perception of depth," *The Journal of Psychology*, vol. 34, no. 1, pp. 107–114, 1952.

[3] W. C. Gogel, "The effect of object familiarity on the perception of size and distance," *The Quarterly journal of experimental psychology*, vol. 21, no. 3, pp. 239–247, 1969.

[4] W. C. Gogel and J. A. Da Silva, "Familiar size and the theory of off-sized perceptions," *Perception & Psychophysics*, vol. 41, no. 4, pp. 318–328, 1987.

[5] M. S. Langer and H. H. Bülthoff, "Depth discrimination from shading under diffuse lighting." *Perception*, vol. 29, no. 6, pp. 649–660, 1999.

[6] H. H. Bülthoff and H. A. Mallot, "Integration of depth modules: stereo and shading," *JOSA A*, vol. 5, no. 10, pp. 1749–1758, 1988.

[7] P. Li, D. Farin, R. K. Gunnewiek, and P. H. de With, "On creating depth maps from monoscopic video using structure from motion," *Proceedings of 27th Symposium on Information Theory in the Benelux*, pp. 508–515, 2006.

[8] W. Liu, Y. Wu, F. Guo, and Z. Hu, "An efficient approach for 2d to 3d video conversion based on structure from motion," *The Visual Computer*, pp. 1–14, 2013.

[9] Y.-L. Chang, C.-Y. Fang, L.-F. Ding, S.-Y. Chen, and L.-G. Chen, "Depth map generation for 2d-to-3d conversion by short-term motion assisted color segmentation," in *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1958–1961.

[10] L.-M. Po, X. Xu, Y. Zhu, S. Zhang, K.-W. Cheung, and C.-W. Ting, "Automatic 2d-to-3d video conversion technique based on depth-from-motion and color segmentation," in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1000–1003.

[11] X. Huang, L. Wang, J. Huang, D. Li, and M. Zhang, "A depth extraction method based on motion and geometry for 2d to 3d conversion," in *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, vol. 3. IEEE, 2009, pp. 294–298.

[12] N. Shabat, "Real-time 2d to 3d from h.264 video compression," M.Sc. thesis, Tel-Aviv University, Israel, 2014.

[13] F. Bellard, M. Niedermayer *et al.*, "Ffmpeg," *http://ffmpeg. org*, 2014.

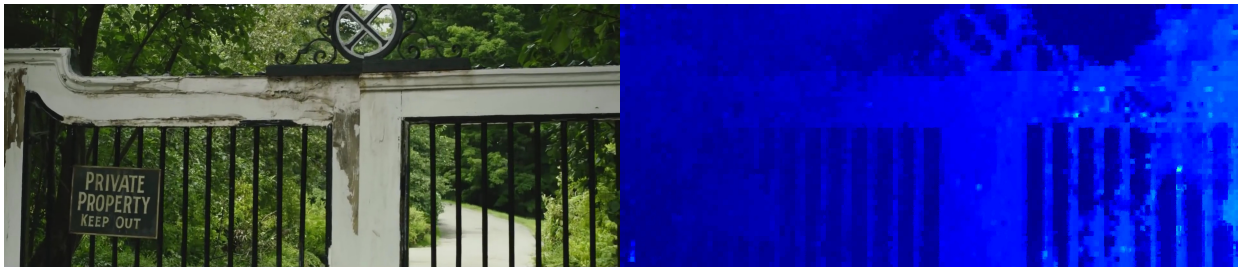
[14] Q. Wei, "Converting 2d to 3d: A survey," in *International Conference, Page (s)*, vol. 7, 2005, p. 14.

[15] W. J. Tam and L. Zhang, "3d-tv content generation: 2d-to-3d conversion," in *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006, pp. 1869–1872.

[16] S. Mohan and L. M. Mani, "Construction of 3d models from single view images: a survey based on various approaches," in *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*. IEEE, 2011, pp. 557–562.

[17] L. Zhang, C. Vázquez, and S. Knorr, "3d-tv content creation: automatic 2d-to-3d video conversion," *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 372–383, 2011.

[18] M. Guttmann, L. Wolf, and D. Cohen-Or, "Semi-automatic stereo extraction from video footage," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 136–142.



(a) Source frame (b) Estimated depth map

(c) Frame's Anaglyph

Fig. 5.1: Example result from the movie "X-Men: Days of Future Past"

[19] X. Cao, Z. Li, and Q. Dai, "Semi-automatic 2d-to-3d conversion using disparity propagation," *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 491–499, 2011.

[20] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross, "Stereobrush: interactive 2d to 3d conversion using discontinuous warps," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*. ACM, 2011, pp. 47–54.

[21] C. Wu, G. Er, X. Xie, T. Li, X. Cao, and Q. Dai, "A novel method for semi-automatic 2d to 3d video conversion," in *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2008*. IEEE, 2008, pp. 65–68.

[22] S. Battiato, S. Curti, M. La Cascia, M. Tortora, and E. Scordato, "Depth map generation by image classification," *Proceedings of SPIE*, vol. 5302, pp. 95–104, 2004.

[23] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1253–1260.

[24] A. Saxena, S. H. Chung, and A. Ng, "Learning depth from single monocular images," *Neural Information Processing Systems (NIPS)*, vol. 18, p. 1161, 2005.

[25] A. Saxena, J. Schulte, and A. Y. Ng, "Depth estimation using monocular and stereo cues." *IJCAI*, vol. 7, 2007.

[26] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Depth perception from a single still image," in *Proceedings of the 23rd national conference on Artificial intelligence (AAAI)*, vol. 3, 2008, pp. 1571–1576.

[27] ——, "Make3d: Learning 3d scene structure from a single still image," *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, vol. 31, no. 5, pp. 824–840, 2009.

[28] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 53–69, 2008.

[29] J. Konrad, M. Wang, and P. Ishwar, "2d-to-3d image conversion by learning depth from examples," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 16–22.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[31] K. Karsch, C. Liu, and S. B. Kang, "Depth extraction from video using non-parametric sampling," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 775–788.

[32] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[33] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 5, pp. 978–994, 2011.

[34] M. T. Pourazad, P. Nasiopoulos, and A. Bashashati, "Random forests-based 2d-to-3d video conversion," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 150–153.

[35] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[36] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[37] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Recovering consistent video depth maps via bundle optimization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[38] ——, "Consistent depth maps recovery from a video sequence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 6, pp. 974–988, 2009.

(a) Source frame (b) Estimated depth map

(c) Frame's Anaglyph

Fig. 5.2: Sample result from the movie "Batman: The Dark Knight"

[39] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *AAAI*, 1982, pp. 133–136.

[40] I. Ideses, L. P. Yaroslavsky, and B. Fishbain, "Real-time 2d to 3d video conversion," *Journal of Real-Time Image Processing*, vol. 2, no. 1, pp. 3–9, 2007.

[41] M. T. Pourazad, P. Nasiopoulos, and R. K. Ward, "An h. 264-based scheme for 2d to 3d video conversion," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 2, pp. 742–748, 2009.

[42] M. Pourazad, P. Nasiopoulos, and R. Ward, "Conversion of h. 264-encoded 2d video to 3d format," in *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*. IEEE, 2010, pp. 63–64.

[43] T.-I. Lin, C.-Y. Yeh, W.-C. Chen, and W.-N. Lie, "A 2d to 3d conversion scheme based on depth cues analysis for mpeg videos," in *Multimedia and Expo (ICME), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1141–1145.

[44] I. E. Richardson, *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[45] B. G. Lindsay, "Mixture models: theory, geometry and applications," in *NSF-CBMS regional conference series in probability and statistics*. JSTOR, 1995, pp. i–163.

[46] A. T. S. Committee *et al.*, "Atsc digital television standard," *Document A/72 Part 2, Feb*, 2014.

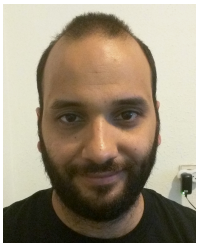
[47] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 839–846.

[48] I. Eser, D. S. Durrie, F. Schwendeman, and J. E. Stahl, "Association between ocular dominance and refraction." *Journal of refractive surgery (Thorofare, NJ: 1995)*, vol. 24, no. 7, pp. 685–689, 2008.

[49] W. H. Ehrenstein, B. E. Arnold-Schulz-Gahmen, and W. Jaschinski, "Eye preference within the context of binocular functions," *Graefe's Archive for Clinical and Experimental Ophthalmology*, vol. 243, no. 9, pp. 926–932, 2005.

[50] M. Reiss and G. Reiss, "Ocular dominance: some family data," *Laterality: Asymmetries of Body, Brain and Cognition*, vol. 2, no. 1, pp. 7–16, 1997.

[51] B. Chaurasia and B. Mathur, "Eyedness," *Cells Tissues Organs*, vol. 96, no. 2, pp. 301–305, 1976.



Nir Shabat works as a Software Engineer at Google. Nir was born in Tel Aviv, Israel. He received both his BS.c and MS.c degrees in Computer Science with honors from Tel Aviv University, in 2003 and 2014, respectively. His research interests include machine learning, big data processing and computer vision.

(a) Source frame (b) Estimated depth map

(c) Frame's Anaglyph

Fig. 5.3: Example result from the movie "Batman: The Dark Knight"

Gil Shabat Gil Shabat is a postdoctoral fellow in the Department of Computer Science in Tel Aviv University. Gil was born in Tel Aviv, Israel. He received his B.Sc, M.Sc and Ph.D degrees in Electrical Engineering all from Tel-Aviv university, in 2002, 2008 and 2015, respectively. Prior to his PhD research, Gil was working in Applied Materials as a computer vision algorithm researcher. His research interests include low rank approximations, fast randomized algorithms, signal/image processing and machine learning.

Amir Averbuch was born in Tel Aviv, Israel. He received the B.Sc and M.Sc degrees in Mathematics from the Hebrew University in Jerusalem, Israel in 1971 and 1975, respectively. He received the Ph.D degree in Computer Science from Columbia University, New York, in 1983. During 1966-1970 and 1973-1976 he served in the Israeli Defense Forces. In 1976-1986 he was a Research Staff Member at IBM T.J. Watson Research Center, Yorktown Heights, in NY, Department of Computer Science. In 1987, he joined the School of Computer Science, Tel Aviv University, where he is now Professor of Computer Science. His research interests include applied harmonic analysis, big data processing and analysis, wavelets, signal/image processing, numerical computation and scientific computing (fast algorithms).