

# Overview of efficient high-quality state-of-the-art depth enhancement methods by thorough design space exploration

Luc Vosters<sup>1,2</sup> · Chris Varekamp<sup>3</sup> · Gerard de Haan<sup>1,4</sup>

Received: 6 June 2014 / Accepted: 30 September 2015 / Published online: 27 October 2015  
© The Author(s) 2015. This article is published with open access at Springerlink.com

**Abstract** High-quality 3D content generation requires high-quality depth maps. In practice, depth maps generated by stereo-matching, depth sensing cameras, or decoders, have low resolution and suffer from unreliable estimates and noise. Therefore, depth enhancement is necessary. Depth enhancement comprises two stages: depth upsampling and temporal post-processing. In this paper, we extend our previous work on depth upsampling in two ways. First we propose PWAS-MCM, a new depth upsampling method, and we show that it achieves on average the highest depth accuracy compared to other efficient state-of-the-art depth upsampling methods. Then, we benchmark all relevant state-of-the-art filter-based temporal post-processing methods on depth accuracy by conducting a parameter space search to find the optimum set of parameters for various upscale factors and noise levels. Then we analyze the temporal post-processing methods qualitatively. Finally, we analyze the computational complexity of each depth upsampling and temporal post-processing method by measuring the throughput and hardware utilization of the GPU implementation that we built for each method.

**Keywords** Depth enhancement · Depth upsampling · Benchmark · Temporal filter · Image plus depth · Parameter space search · Design space exploration

## 1 Introduction

### 1.1 3D-TV

Following the successful introduction of color four decades ago, and high definition in the last decade, many researchers are currently exploring opportunities for 3-dimensional television (3D-TV) systems [16]. In such systems, depth maps play an important role for both 3D content generation and transmission.

In 3D content generation, depth maps can be obtained from a depth sensing (ToF) camera or an infrared-based structured-light sensor such as Primesense [29], that is rectified and upsampled using a color RGB camera as guiding signal [6, 11, 47]. Furthermore, depth can be computed from the 3D depth cues that are present in monoscopic video [49]. The resulting depth map is then used to generate virtual views from the original color image with depth image-based rendering [13, 48]. Alternatively, when two viewpoints of the same scene are available depth can be computed with a stereo matcher [34].

### 1.2 3D standards

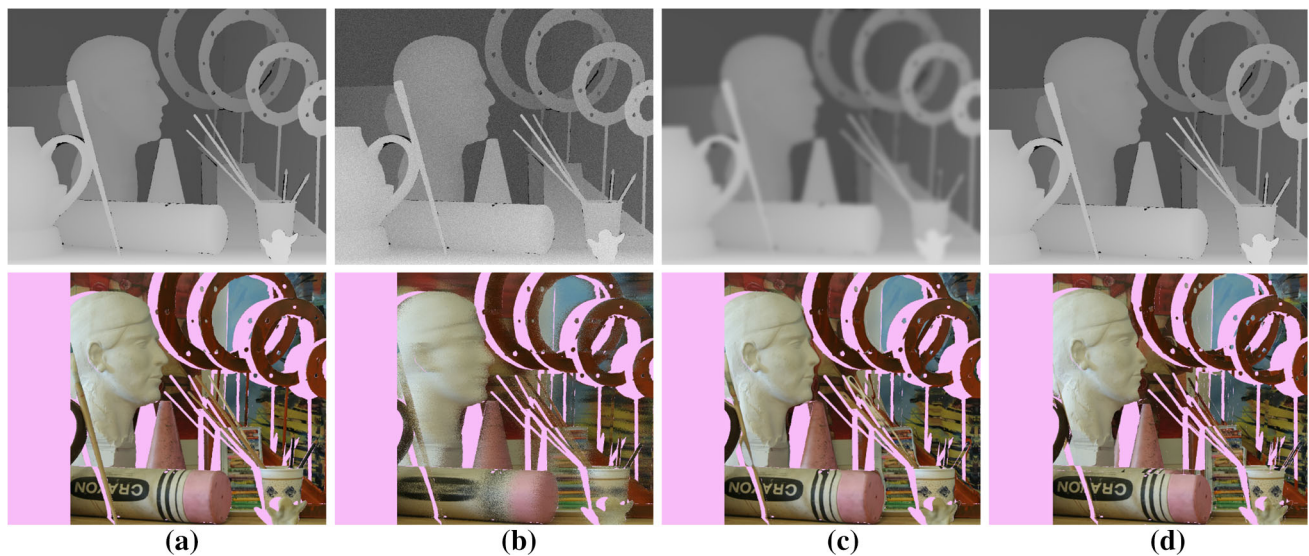
For the transmission of 3D content, Fehn et al. [13] first proposed the 2D-plus-depth format. In this format monoscopic video and per pixel depth information are encoded as separate streams and jointly transmitted. Since depth maps are spatially smooth everywhere, except at sparsely

---

✉ Luc Vosters  
l.p.j.vosters@tue.nl

Gerard de Haan  
g.de.haan@philips.com

<sup>1</sup> Electronic Systems Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands  
<sup>2</sup> Axon Digital Design, 5126 BB Gilze, The Netherlands  
<sup>3</sup> TP Vision Netherlands (at the time of writing), 5616 LZ Eindhoven, The Netherlands  
<sup>4</sup> Philips Research Eindhoven, 5656 AA Eindhoven, The Netherlands



**Fig. 1** Artifacts in depth image-based rendering. The *first* and *second* row show the left disparity map, which has been degraded, and the corresponding reconstructed *left color image*, respectively. The ground-truth depth and *color images* of this stereo pair can be downloaded from the Middlebury stereo vision website [33]. We obtained the reconstructed views by warping the *right color image* to the *left color image* with the degraded *left disparity map*. We marked the occluded areas in *pink*. These areas are computed from the

ground-truth *left* and *right* disparity images. **a** Ground truth depth map and interpolated view. **b** Noisy depth map (PSNR 24.1 dB) and interpolated view. Texture in sharp details have disappeared. **c** Blurred depth map (gaussian filter with standard deviation 10) and interpolated view. Edges are blurred and distorted. **d** Misaligned depth map and interpolated view. The depth map was shifted to the *left* and to the *top* by 20 pixels. This causes ghosting and distorted objects

located depth transitions, they are strongly compressed. More recently other standards for multiview video coding plus depth (MVD) have been proposed in which depth maps are jointly encoded with their corresponding color images [37]. This provides more flexibility for free viewpoint television (FTV) and significantly reduces the transmission bandwidth.

### 1.3 Requirements for depth maps

For high-quality 3D content generation, a high-quality depth/disparity map is required.<sup>1</sup> We will use the terms depth and disparity interchangeably. A high-quality depth map should meet the following subjective requirements:

1. It should be smooth within the interior of objects.
2. Depth transitions should be sharp at object boundaries.
3. Depth transitions should be well aligned with object boundaries as visible in the color image.
4. It should have the same resolution as the color image it corresponds to.
5. It should be temporally stable.

<sup>1</sup> A depth map contains a measure for the distance of an imaged point (pixel) to its corresponding point in 3D space along the camera's optical axis. Conversely, a disparity map indicates the horizontal shift between two corresponding pixels in the two views of a parallel stereo camera. When the intrinsic and extrinsic camera parameters are known depth can be converted into disparity and vice versa.

Figure 1 shows what happens when requirements 1–3 are not satisfied. Furthermore, when requirement 4 is not met, we cannot render a new view. Cheng et al. [7] and Vosters et al. [44] have shown that spatiotemporal coherence is a key factor in generating high-quality perceptually satisfactory virtual images. Therefore, requirement 5 must be satisfied.

### 1.4 Depth errors

Unfortunately most depth maps do not meet these requirements. Depth maps from a ToF camera or a structured light depth sensor typically have a much lower resolution than HD. For instance, the Primesense sensor [29] outputs  $640 \times 480$  resolution depth maps, whereas a current HD TV operates at a resolution of  $1920 \times 1080$ . In addition, depth sensing cameras are typically sensitive to the reflectivity of objects. Their depth maps contain significant amounts of noise in areas corresponding to objects with low reflectance [2]. Furthermore, stereo matchers and 2D-to-3D converters compute the depth map at a low resolution to reduce the computational burden and achieve real-time frame rates. Their depth maps often have considerable areas with unreliable depth estimates, and depth edges may not be well aligned with the color image.

In 3D transmission and coding a low-quality depth map that is obtained by a depth sensing camera, 2D-to-3D conversion, or stereo-matcher, generally requires

upsampling and temporal post-processing prior to transmission. Temporal post-processing removes temporal fluctuations and thereby improves the coding efficiency, and prevents visual fatigue and eyestrain [8]. Compression introduces blocking artifacts, mosquito noise and edge blurring. These artifacts need to be removed at the receiver's decoder by a depth enhancement filter to obtain a high-quality virtual view with depth image-based rendering.

Removing compression artifacts from depth maps to improve the visual quality of rendered virtual views is a well studied topic. State-of-the-art solutions include [9] and [19]. These solutions enhance a decoded depth map that has the same resolution as the transmitted color image. Klimaszewski et al. [21] have shown that a higher compression rate without the loss of quality in the rendered virtual view can be attained by downsampling the depth map prior to encoding. This requires an upsampling filter after decoding. Upsampling filters for decoding that simultaneously sharpen depth edges have been proposed in [10, 18, 27, 35]. However, these filters are solely designed to remove compression artifacts, but do not take into account image noise and other inaccuracies that arise from depth estimators for 2D-to-3D conversion, stereo-matchers or depth sensing cameras. Since we focus on depth video post-processing rather than compression artifact reduction in particular, we will not include them in this paper.

In MVD coding, multiview depth and color images can be used to enhance the depth maps. For two views Mueller et al. [25] propose an adaptive joint trilateral weighted median filter that exploits multiview depth images by extending the joint bilateral filter with a confidence weight. This weight is computed from a left–right depth consistency check and cross-correlation in the color images. In [26] Mueller et al. extend their previous work in [25] with a hybrid recursive post-processing method and motion estimation to obtain spatiotemporally stable depth images. However, in this paper we limit our scope to one color view and its corresponding depth map, since for depth-sensing cameras, the 2D-plus-depth format and 2D-to-3D converters multiview information is unavailable. Hence, we will not include Mueller et al. in our benchmark.

### 1.5 Depth enhancement

Depth enhancement may be decomposed into two stages: a depth upsampling stage (DU) and a Temporal Post-Processing (TPP) stage. In the first stage the depth map of a single image is upsampled, denoised and aligned with the edges in the high-resolution color image. In the second stage, temporal stability is enforced with a temporal post-processing filter.

Depth upsampling is a widely studied topic for which many techniques have been proposed. These can be roughly classified into optimization, segmentation, and filter-based techniques. In optimization-based depth upsampling a cost function is formulated that generally consists of various energy terms which provide a tradeoff between depth alignment with the high-resolution color image, spatiotemporal coherence and fine detail preservation. These cost functions can be minimized with iterative optimization techniques such as Markov Random Fields (MRF) [28, 38], Conjugate Gradient (CG) [11], iterative cost volume filtering [47], and quadratic optimization [36]. These methods have a high computational complexity and require a large amount of memory. Hence they are not suitable for implementation in real-time 3D-TV systems. In segmentation-based depth upsampling methods (Soh et al. [38], Tallon et al. [39] and Kim et al. [20]), the color and bicubically upsampled depth images are jointly segmented and regions where depth and color are inconsistent, are corrected. Generally these approaches require an additional post-processing step to remove the false depth transitions, introduced by segmentation. Conversely, filter-based techniques avoid the high computational cost and memory requirements of iterative optimization, and allow for scanline-based implementations. Therefore, these methods are potential candidates for implementation in embedded real-time devices with limited resources for 3D-TV.

Temporal Post-Processing on the other hand has received significantly less attention in literature. Like depth upsampling, temporal post-processing methods can be classified in optimization and filter-based depth upsampling methods. Again we limit our scope only to filter-based methods. In spite of its importance for high-quality 3D video, there has never been a fair attempt to quantitatively assess the performance of temporal post-processing methods on large datasets. Therefore, in this paper we extend our previous work on depth upsampling in [45] to temporal post-processing, by thoroughly benchmarking and surveying the state-of-the-art filter-based temporal post-processing methods on depth accuracy (Eq. 20) and complexity for upscale factors  $U \in \{8, 4, 2\}$ , and two simulated Gaussian noise settings, i.e., ( $\xi \in \{0.05, 0.1\}$ ), in the noise model of Edeler et al. [12].

The remainder of the paper is organized as follows. In Sect. 2, we introduce a new efficient depth upsampling method, and we evaluate its performance on depth accuracy with the other efficient depth upsampling methods that we have evaluated in our previous work [45]. Section 3 describes the state-of-the-art methods for temporal post-processing. Then Sect. 4 describes the test set, the evaluation metrics and the benchmark setup. Next Sect. 5 presents the results of the benchmark for all state-of-the-art

temporal post-processing algorithms. Finally, Sect. 7 concludes the paper.

## 2 Depth upsampling

In our previous work [45], we gave an overview and evaluation of all relevant state-of-the-art efficient depth upsampling methods. In this section, we extend this work by proposing PWAS-MCM, which is a combination of two existing techniques: the Multiscale Color Measure (MCM) (Min et al. [24]) and PWAS (Garcia et al. [15]).

PWAS-MCM upsamples depth images with a factor that is a power of 2. First the low-resolution depth map  $d$  is sparsely upsampled with factor  $U$ . Then the sparse depth map is filled in  $L$  steps according to the algorithm described in Algorithm 1. Note that PWAS-MCM is recursive, i.e., at step  $l$  we include the depth values that have already been computed in steps  $l \geq l + 1$ . The color guidance image  $\mathbf{I}^l$  at step  $l$  is computed by convolving the input color image  $\mathbf{I}$  with Gaussian low-pass filter kernel  $G_{\sigma_{\text{LPF}}^l}$  which scale is given by

$$\sigma_{\text{LPF}}^l = \begin{cases} \sigma_{\text{LPF}}^1 \cdot l & \text{if } l = 1, \dots, L - 1 \\ 0 & \text{if } l = 0 \end{cases}, \tag{1}$$

where  $\sigma_{\text{LPF}}^1$  is a parameter that can be specified by the user.<sup>2</sup> The function for step 2 in Algorithm 1 is given by

$$f(\mathbf{p}, \mathbf{I}^l, B, D, R^l) = \frac{\sum_{\mathbf{q} \in N(\mathbf{p}, R^l)} B_{\mathbf{q}} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p}} - \mathbf{I}_{\mathbf{q}}) C_{\mathbf{q}}^l D_{\mathbf{q}}}{\sum_{\mathbf{q} \in N(\mathbf{p}, R^l)} B_{\mathbf{q}} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p}} - \mathbf{I}_{\mathbf{q}}) C_{\mathbf{q}}^l}, \tag{2}$$

where

$$C_{\mathbf{p}}^l = G_{\sigma_c}(\nabla^l D_{\mathbf{p}}), \tag{3}$$

$$\nabla^l D_{\mathbf{p}} = [D_{\mathbf{p}+\mathbf{u}_x^l} - D_{\mathbf{p}-\mathbf{u}_x^l}, D_{\mathbf{p}+\mathbf{u}_y^l} - D_{\mathbf{p}-\mathbf{u}_y^l}]$$

denotes the credibility weight,  $\mathbf{u}_x^l = [2^{l+1}, 0]^T$ ,  $\mathbf{u}_y^l = [0, 2^{l+1}]^T$ ,

$$N(\mathbf{p}, R) = \{\mathbf{q} \mid \|\mathbf{p} - \mathbf{q}\|_{\infty} \leq R \wedge \mathbf{q} \in \mathbb{Z}^2\}, \tag{4}$$

$$G_{\sigma}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}\|_2^2}{2\sigma^2}\right),$$

$d$  denotes the low-resolution input depth image,  $D$  is a high-resolution depth image that gets filled in  $L$  steps,  $R^{\text{du}}$  is the kernel radius for depth upsampling,  $\mathbf{I}_{\mathbf{p}}$  is a vector containing the R, G and B color channel of pixel  $\mathbf{p}$  in the input guidance image,  $B$  is a binary indicator image, and

$G_{\sigma_r}(\mathbf{x})$  and  $G_{\sigma_s}(\mathbf{x})$  denote the negative exponential range and domain filter kernel,<sup>3</sup> respectively.

---

**Algorithm 1** Multiscale color measure framework for PWAS-MCM.  $d$  is the low resolution input depth image,  $D$  is a high resolution depth image that gets filled in  $L$  steps,  $V_{\mathbf{p}}$  and  $B_{\mathbf{p}}$  are binary indicator images at pixel location  $\mathbf{p}$ ,  $\mathbf{I}^l$  denotes the color guidance image at step  $l$ ,  $R^l$  is the kernel radius at step  $l$ ,  $R^{\text{du}}$  is the kernel radius at step  $l = 0$ ,  $U$  is the upscale factor which must be a power of 2,  $\text{mod}$  is the modulo operator that when applied to vectors operates on each element independently, and  $\mathbf{I}^l = \mathbf{I} * G_{\sigma_{\text{LPF}}^l}$  denotes convolution of each independent R, G and B channel in the color guidance image with a Gaussian kernel of scale  $\sigma_{\text{LPF}}^l$  (Eq. 1).

---

```

1: procedure MCM(I, d)
2:    $\tilde{V}_{\mathbf{p}} \leftarrow 1, \forall \mathbf{p} \in \{\mathbf{k} \mid \mathbf{k} \bmod U = 0\}$ 
3:    $\tilde{V}_{\mathbf{p}} \leftarrow 0, \forall \mathbf{p} \in \{\mathbf{k} \mid \mathbf{k} \bmod U \neq 0\}$ 
4:    $L \leftarrow \log_2(U)$ 
5:    $l \leftarrow L - 1$ 
6:    $D_{\mathbf{p} \cdot U} = d_{\mathbf{p}}$ 
7:   while (  $l \geq 0$  ) do
8:      $\mathbf{I}^l \leftarrow \mathbf{I} * G_{\sigma_{\text{LPF}}^l}$ 
9:      $V \leftarrow \tilde{V}$ 
10:     $R^l \leftarrow 2^{l+1} R^{\text{du}}$ 
11:    for (  $\mathbf{p} \in \{\mathbf{k} \mid \mathbf{k} \bmod 2^l = 0\}$  ) do
12:       $D_{\mathbf{p}} \leftarrow f(\mathbf{p}, \mathbf{I}^l, B, D, R^l)$  ▷ See Eq. 2
13:       $\tilde{V}_{\mathbf{p}} \leftarrow 1$ 
14:    end for
15:     $l \leftarrow l - 1$ 
16:  end while
17:   $D^{\text{PWAS-MCM}} \leftarrow D$ 
18: end procedure

```

---

In contrast to JBU, PWAS and PWAS-MCM compute a credibility map that contains a credibility weight  $C_{\mathbf{p}}$  for each pixel  $\mathbf{p}$  in the depth image. This credibility map has low values on and around depth transitions and noise. Consequently, misaligned and noisy pixels around a depth transition are discarded. Hence, the output depth map features sharp edges and texture copying artifacts are suppressed. In addition, MCM prevents the aliasing artifact,<sup>4</sup> by prefiltering the color guidance image prior to each upsampling step.

<sup>3</sup> Note that in this paper we implement the negative exponential function as:  $G_{\sigma}(\mathbf{x}) = \exp\left(-\min\left(\frac{\|\mathbf{x}\|_2^2}{2\sigma^2}, \epsilon\right)\right)$ , where  $\epsilon$  is the maximum value of the argument in the negative exponential. In our implementation we set  $\epsilon = 708$ . We clip  $\frac{\|\mathbf{x}\|_2^2}{2\sigma^2}$  to prevent undefined output pixels, which occur when all weights in the kernel become zero due to underflow. Underflow does not occur in the regular joint bilateral filter since its center weight is always equal to 1; however, for depth upsampling and temporal post-processing this is generally not the case. Consider for example, the range kernel of Eq. 9. When  $\mathbf{q} = \mathbf{p}$ , then  $\mathbf{I}_{\mathbf{p},n}$  does not necessarily have to be equal to  $\mathbf{I}_{\mathbf{q},n-1}$  and therefore all kernel weights could still underflow.

<sup>4</sup> Aliasing artifacts in depth upsampling occur during the range weight computation when the color guidance image is sparsely sampled without being properly prefiltered first [32].

<sup>2</sup> Note that we clip the radius of the Gaussian low-pass filter at  $3\sigma_{\text{LPF}}^l$ .



**Table 1** Benchmark for depth upsampling obtained by a full parameter space search

$(U = 2, \zeta = 0)$			$(U = 2, \zeta = 0.05)$			$(U = 2, \zeta = 0.1)$		
Rank	Method	DA	Rank	Method	DA	Rank	Method	DA
1	PWAS-MCM	37.77	1	NAFDU	34.12	1	PWAS	33.37
2	PWAS	37.01	2	PWAS-MCM	34.10	2	PWAS-MCM	33.37
3	WMF	36.35	3	PWAS	34.10	3	NAFDU	33.30
$(U = 4, \zeta = 0)$			$(U = 4, \zeta = 0.05)$			$(U = 4, \zeta = 0.1)$		
Rank	Method	DA	Rank	Method	DA	Rank	Method	DA
1	PWAS-MCM	34.42	1	PWAS-MCM	32.35	1	PWAS-MCM	31.71
2	PWAS	34.16	2	PWAS	32.19	2	JBU-MCM	31.58
3	WMF	33.85	3	JBU-MCM	32.18	3	PWAS	31.57
$(U = 8, \zeta = 0)$			$(U = 8, \zeta = 0.05)$			$(U = 8, \zeta = 0.1)$		
Rank	Method	DA	Rank	Method	DA	Rank	Method	DA
1	PWAS-MCM	32.18	1	PWAS-MCM	30.64	1	PWAS-MCM	30.01
2	PWAS	31.93	2	JBU-MCM	30.52	2	JBU-MCM	30.00
3	JBU-MCM	31.69	3	WMF	30.33	3	NAFDU	29.81

Top 3 ranking of all efficient depth upsampling methods described in Vosters et al. [45], and PWAS-MCM. We rank all methods on depth accuracy (DA) for upscale factors  $U \in \{8, 4, 2\}$  and noise levels  $\zeta \in \{0, 0.05, 0.1\}$ . Depth accuracy is computed over all image regions as the PSNR between an upsampled and ground-truth depth image in dB. The images for  $\zeta = 0$  were taken from our experiment in [45] in which we filled in the black pixels in the depth image

Table 1 ranks for each tested upscale factor  $U \in \{8, 4, 2\}$  and two simulated noise levels  $\zeta \in \{0.05, 0.1\}$ , the top 3 methods described in Vosters et al. [45] including PWAS-MCM on depth accuracy<sup>5</sup> In contrast to our previous work in [45] where we simulate depth noise as spatially invariant white Gaussian noise, in this work we simulate noise with the time-of-flight (ToF) based noise model of Edeler et al. [12]. Edeler et al. assume that depth noise is spatially variant and is inversely proportionally to the intensity of the incident light on the sensor. This model is more accurate because it is based on the underlying physical noise model of a ToF depth sensor. We introduce Edeler et al.’s noise model in Sect. 4.1.

For each benchmark method we perform a parameter space search to find the optimum set of parameters that maximize the average depth accuracy over 12 test images for each upscale factor and noise level individually. The parameter space for PWAS-MCM is computed as the set of all possible parameter combinations of  $\sigma_r, \sigma_s, \sigma_c, B$  and  $\sigma_{LPF}^1$  within the ranges specified in Table III of Vosters et al. [45].

The results show that for all upscale factors and noise levels, PWAS-MCM has rank 1, except for  $(U = 2, \zeta \in \{0.05, 0.1\})$  where it has rank 2. Furthermore, Table 1 shows that for  $\zeta = 0$  the difference between PWAS-MCM and the rank 2 method PWAS is 0.76 dB for  $U = 2$ . This

<sup>5</sup> In our previous work, we compute the depth accuracy for an upsampled depth image as the PSNR between said image and its corresponding ground-truth depth image.

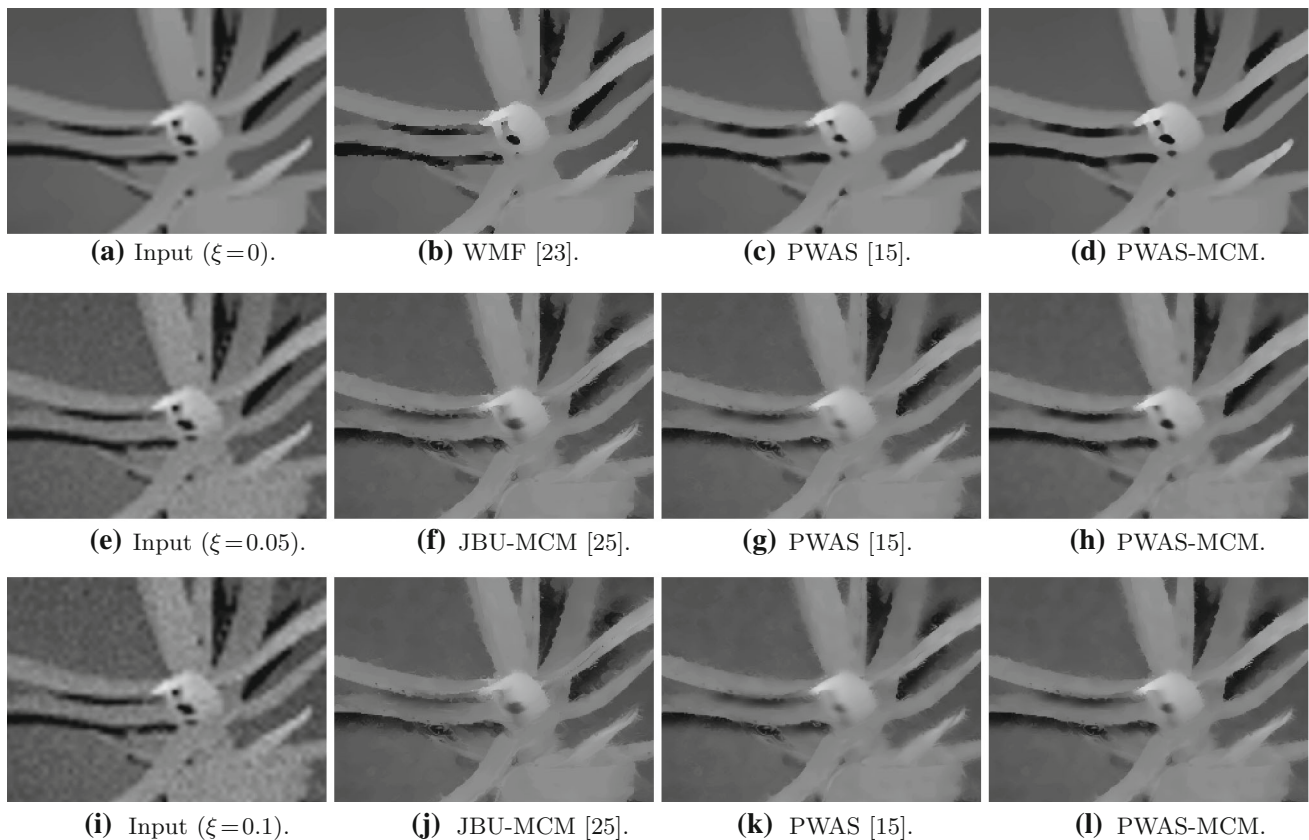
difference gradually gets smaller when  $U$  increases. In contrast, for  $\zeta \in \{0.05, 0.1\}$  the difference between all top three methods is small.

Due to the credibility weight, and recursive application of PWAS-MCM, misaligned and noisy pixels around depth transitions are ignored. Therefore, the output depth image is smooth, contains very few texture copy artifacts and has sharp edges. In addition, the prefilter in the MCM framework reduces aliasing artifacts in the upsampled depth image. This is shown in Fig. 2 in which we qualitatively compare PWAS-MCM, JBU-MCM, PWAS and WMF for  $U = 4$ , and  $\zeta \in \{0, 0.05, 0.1\}$  for the *aloe* image. The upsampled depth images for each method in Fig. 2 were computed with the set of parameters that gave the maximum average depth accuracy over all test images for each selected configuration of  $U$  and  $\zeta$ .

### 3 Temporal post-processing

Several methods exist for temporal post-processing. Choi et al. [8] propose a 3D-JBU filter (Sect. 3.1) which extends the kernel of a spatial (upsampling) filter to the temporal domain. In contrast with regular JBU (Kopf et al. [22]), the filter kernel of 3D-JBU comprises not only the current frame, but also the two previous frames.

Both Vosters et al. [44] and Richardt et al. [31] propagate the output depth of the previous frame to the current



**Fig. 2** Qualitative comparison of the top 3 depth upsampling methods in the ranking of Table 1 for  $U = 4$  and  $\xi \in \{0, 0.05, 0.1\}$  on the aloe image. This image shows that in PWAS and JBU-MCM guidance image textures get copied near depth edges, unlike PWAS-MCM. While WMF produces the sharpest edges, quantization artifacts appear. Overall, it can be seen that PWAS-MCM offers a

good tradeoff between sharp depth transitions, strong noise suppression and no texture copy artifacts. For a good comparison, these (vector) images should be viewed in the electronic version of the paper, in which they can be enlarged to their original resolution by zooming in

frame with a joint bilateral propagation filter (JBPF), and average the two with a falloff factor. The quality of the output depth map depends largely on the temporal propagation filter. In Sect. 3.2, we describe three methods for temporal propagation.

Alternatively, Fu et al. [14] propose a 3-tap recursive temporal filter to filter stationary image areas only. First they compute the structural similarity (SSIM) between pixel  $p$  in the color images of the current frame  $n$  and pixel  $p$  in frame  $n - 1$  and  $n - 2$ . Then the output depth at pixel  $p$  in the previous frame is multiplied by a weight, which is computed from the per pixel SSIM, and averaged with the depth at pixel  $p$  in the current frame. However, since this filter only has a 1 by 1 spatial aperture, it is not robust to noise. Moreover, the filter does not work when camera and/or object motion are present. Therefore, we will not include it in our temporal post-processing benchmark.

Furthermore, Kim et al. [20] estimate the motion of each pixel in the color image with a block matcher. Then if a pixel is stationary in both the previous and the current frame the depth of the previous frame is copied to that

pixel, otherwise the upsampled depth of the current frame is kept. This filter enforces temporal consistency only in stationary image regions, but fails during object and camera motion. Hence, we will not include it in our temporal postprocessing benchmark.

Min et al. extend their weighted mode filter to the temporal domain, by temporally filtering the histogram ([24], Eq. 5<sup>6</sup>) for each pixel with a temporally recursive motion-compensated 3-tap filter. However, this approach is far from practical, since it either requires memory for storing at least two 255-bin histograms for each pixel in an image, or recomputing the histograms on the fly. The former situation requires too much memory, while the latter is computationally just too complex.

In Sects. 3.1, 3.2, 3.3 and 3.4 we describe all methods that we include in our temporal post-processing benchmark of Sect. 5. Additionally, Table 2 gives an overview of these methods, their parameters and a reference to the equations that define them.

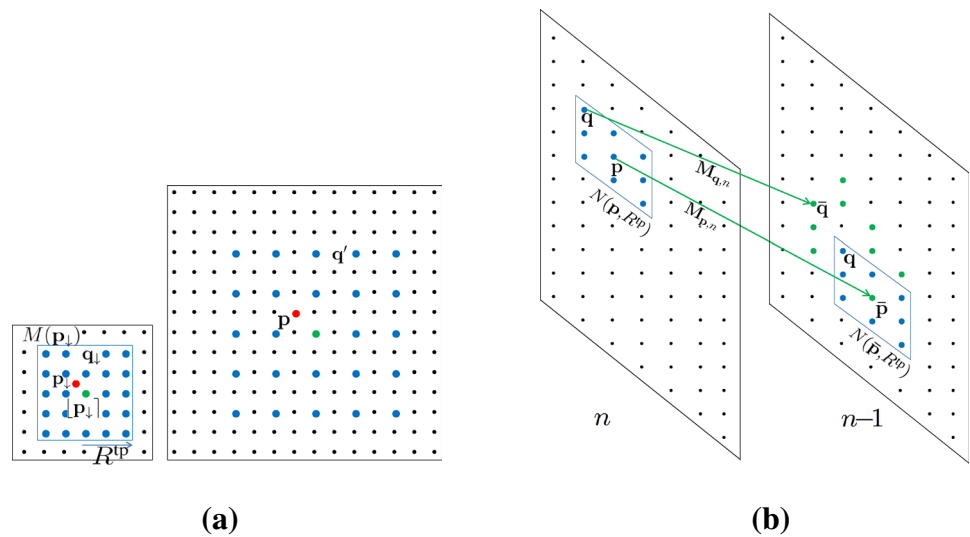
<sup>6</sup> See Eq. 7 in Vosters et al. [45] for the equivalent in our notation.

**Table 2** The various temporal post-processing methods that we have included in our benchmark together with their corresponding equation numbers (column 2) and parameters (column 3)

Method name	Equation nos.	Parameters	No. parameter configs.
3DJBU	5–7	$\sigma_s, \sigma_r, N_T$	6000
JP	8, 9	$\sigma_s, \sigma_r, \phi$	1000
JPMC-BROX	8, 10	$\sigma_s, \sigma_r, \zeta, \phi$	13,000
JPMC-3DRS	8, 10	$\sigma_s, \sigma_r, \zeta, \phi$	13,000
JPMC+-BROX	8, 13	$\sigma_s, \sigma_r, \sigma_d, \sigma_f, \phi$	100,000
JPMC+-3DRS	8, 13	$\sigma_s, \sigma_r, \sigma_d, \sigma_f, \phi$	100,000
MCPF-3DRS	16–18	$\sigma_s, \sigma_r, \sigma_d, \phi$	10,000
RICHARDT	14, 15	$\sigma_s, \sigma_r, \sigma_d, \sigma_f, \phi$	100,000
NULL	2	n.a.	1

A method that has the suffix BROX or 3DRS uses Brox et al.’s [3] or de Haan et al.’s [17] motion estimator. Column 4 shows for each method the total number of parameter configurations to test. This number can be computed by multiplying the number of values per parameter of all parameters for a designated method. The parameter range for each parameter is listed in Table 5

**Fig. 3 a** Operation of Eqs. 5 and 7 on the low and high-resolution grids for  $N_T = 1$ . **b** Depiction of how Eqs. 9–16 are defined on the grids of the current and previous frame



### 3.1 3D joint bilateral upsampling (3DJBU)

The 3D-JBU filter of Choi et al. [8] is given by

$$3DJBU_{\mathbf{p},n} = \frac{1}{\kappa_{\mathbf{p}}^{3DJBU}} \sum_{k=0}^{N_T} \sum_{\mathbf{q}_l \in M(\mathbf{p}_l)} G_{\sigma_s}(\mathbf{p}_l - \mathbf{q}_l) \cdot G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q}',n-k}) d_{\mathbf{q}_l,n-k}, \tag{5}$$

$$\kappa_{\mathbf{p}}^{3DJBU} = \sum_{k=0}^{N_T} \sum_{\mathbf{q}_l \in M(\mathbf{p}_l)} G_{\sigma_s}(\mathbf{p}_l - \mathbf{q}_l) \cdot G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q}',n-k}),$$

where  $n$  is the frame number and  $N_T$  is the temporal window size,

$$M(\mathbf{p}_l) = \{\mathbf{j} \mid \|\lfloor \mathbf{p}_l \rfloor - \mathbf{j} \|_{\infty} \leq R^{tp} \wedge \mathbf{j} \in \mathbb{Z}^2\}, \tag{6}$$

$$\mathbf{p}_l = \mathbf{p} / U,$$

$$\mathbf{q}' = \mathbf{q}_l \cdot U$$

where  $U$  is the upsample factor,  $\lfloor \cdot \rfloor$  is the round operator applied to each vector element, and  $R^{tp}$  is the temporal filter kernel radius. Note that  $\mathbf{p}_l$  may be non-integer, while  $\mathbf{q}_l$  is integer by definition. Figure 3a shows how Eqs. 5 and 6 are defined on the low and high-resolution grids. Finally, the output depth image at frame  $n$  is

$$D_{\mathbf{p},n}^{OUT} = D_{\mathbf{p},n}^{3DJBU}. \tag{7}$$

### 3.2 Joint bilateral depth propagation

Joint bilateral depth propagation methods apply a recursive temporal post-processing method to obtain a temporally stable depth map. They compute the output depth image at frame  $n$  as

$$D_{\mathbf{p},n}^{OUT} = D_{\mathbf{p},n}^{PWAS-MCM} \cdot (1 - \phi) + D_{\mathbf{p},n}^{TP} \cdot \phi, \tag{8}$$

where  $D_{\mathbf{p},n}^{\text{PWAS-MCM}}$  is the upsampled depth image computed by PWAS-MCM<sup>7</sup> (Eq. 2),  $D_{\mathbf{p},n}^{\text{TP}}$  is the temporally propagated version of  $D_{\mathbf{p},n-1}^{\text{OUT}}$ ,  $\phi \in [0, 1]$  is a constant, and TP is one of the temporal propagation methods in {JP, JPMC, JPMC+} which we will describe next.

### 3.2.1 Joint propagation (JP)

For key-frame propagation Varekamp et al. [42] compute a temporal prediction of the next depth image from the previous one with a joint bilateral propagation filter (JP) given by

$$D_{\mathbf{p},n}^{\text{JP}} = \frac{1}{\kappa_{\mathbf{p}}^{\text{JP}}} \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}) D_{\mathbf{q},n-1}^{\text{OUT}},$$

$$\kappa_{\mathbf{p}}^{\text{JP}} = \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}),$$
(9)

where  $D_{\mathbf{q},n-1}^{\text{OUT}}$  is the output depth image at the previous frame, and  $N(\mathbf{p})$  is as defined in Eq. 4, except that  $R^{\text{du}}$  should be replaced by the temporal filter kernel radius  $R^{\text{TP}}$ .

### 3.2.2 Joint propagation motion compensation (JPMC)

Cao et al. [4] proposed to extend Eq. 9 with motion compensation according to

$$D_{\mathbf{p},n}^{\text{JPMC}} = \frac{1}{\kappa_{\mathbf{p}}^{\text{JPMC}}} \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}^*(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}) D_{\mathbf{q},n-1}^{\text{OUT}},$$

$$\kappa_{\mathbf{p}}^{\text{JPMC}} = \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}^*(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}),$$

$$G_{\sigma_r}^*(\mathbf{x}) = \begin{cases} \exp(-\|\mathbf{x}\|_2^2 / (2\sigma^2)) & \text{if } \|\mathbf{x}\|_2^2 < \zeta^2 \\ 0 & \text{otherwise,} \end{cases}$$
(10)

where

$$\mathbf{p} = [\mathbf{p} + \mathbf{M}_{\mathbf{p},n}]$$
(11)

is the motion-compensated version of pixel  $\mathbf{p}$ , and  $\mathbf{M}_{\mathbf{p},n}$  is the motion vector estimated at pixel  $\mathbf{p}$  in image  $\mathbf{I}_n$  with image  $\mathbf{I}_{n-1}$  as reference. When  $\|\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}\|_2^2 \geq \zeta^2$ ,  $\forall \mathbf{q} \in N(\mathbf{p})$ , all kernel weights are zero, hence the output pixel is undefined. Cao et al. do not provide a solution for this particular case. Therefore, in this case we set

$$D_{\mathbf{p},n}^{\text{JPMC}} = D_{\mathbf{q},n-1}^{\text{OUT}},$$
(12)

where

<sup>7</sup> We selected PWAS-MCM as depth upsampling method, because it achieves the best performance over all upsampling factors and noise levels in our parameter space analysis of Sect. 2.

$$\hat{\mathbf{q}} \in \left\{ \mathbf{k} \mid \left( \|\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{k},n-1}\|_2^2 \leq \|\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{l},n-1}\|_2^2 \right) \wedge \mathbf{k}, \mathbf{l} \in N(\bar{\mathbf{p}}, R^{\text{TP}}) \right\}.$$

### 3.2.3 Joint propagation motion compensation plus (JPMC+)

Richardt et al. [31] go one step further, and compensate each kernel pixel for motion. Their temporal propagation filter is given by

$$D_{\mathbf{p},n}^{\text{JPMC+}} = \frac{1}{\kappa_{\mathbf{p}}^{\text{JPMC+}}} \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}) \cdot G_{\sigma_d}(D_{\mathbf{p},n} - D_{\mathbf{q},n-1}) G_{\sigma_f}(\mathbf{M}_{\mathbf{q},n}) D_{\mathbf{q},n-1}^{\text{OUT}},$$

$$\kappa_{\mathbf{p}}^{\text{JPMC+}} = \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n-1}) \cdot G_{\sigma_d}(D_{\mathbf{p},n} - D_{\mathbf{q},n-1}) G_{\sigma_f}(\mathbf{M}_{\mathbf{q},n}),$$
(13)

where  $\mathbf{p}$  and  $\mathbf{q}$  are the motion-compensated version of pixel  $\mathbf{p}$  and  $\mathbf{q}$ , respectively (Eq. 11), in the previous frame. Richardt et al. added the additional term  $G_{\sigma_d}(D_{\mathbf{p},n} - D_{\mathbf{q},n-1})$ , to prevent blurring across depth edges.

## 3.3 Richardt (RICHARDT)

Richardt et al. [31] propose a depth enhancement scheme that upsamples a low-resolution depth image, followed by a spatiotemporal filtering scheme according to

$$D_{\mathbf{p},n}^{\text{OUT}} = D_{\mathbf{p},n}^{\text{SF}} \cdot (1 - \phi) + D_{\mathbf{p},n}^{\text{JPMC+}} \cdot \phi,$$
(14)

where

$$D_{\mathbf{p},n}^{\text{SF}} = \frac{1}{\kappa_{\mathbf{p}}^{\text{SF}}} \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\gamma(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n})).$$

$$G_{\sigma_d}(D_{\mathbf{p},n}^{\text{MRFI}} - D_{\mathbf{q},n}^{\text{MRFI}}) D_{\mathbf{q},n}^{\text{MRFI}},$$

$$\kappa_{\mathbf{p}}^{\text{JPMC+}} = \sum_{\mathbf{q} \in N(\mathbf{p}, R^{\text{JP}})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\gamma(\mathbf{I}_{\mathbf{p},n} - \mathbf{I}_{\mathbf{q},n})).$$

$$G_{\sigma_d}(D_{\mathbf{p},n}^{\text{MRFI}} - D_{\mathbf{q},n}^{\text{MRFI}}),$$

$$\gamma = \max(0, \min(2 - \|\mathbf{M}_{\mathbf{q},n}\|_2 / \sigma_f, 1)),$$
(15)

is a filter that is applied to the upsampled depth image,  $D_{\mathbf{p},n}^{\text{JPMC+}}$  is the temporal prediction given by Eq. 13,  $D_{\mathbf{p},n}^{\text{MRFI}}$  is a depth image that is upsampled by the multi-resolution fill-in (MRFI) approach that was proposed by Richardt et al. [31],<sup>8</sup> and  $\sigma_f$  is the same Gaussian scale parameter as in Eq. 13.

<sup>8</sup> See Table 3 in Vosters et al. [45] for a description of the MRFI algorithm in our notation.



In contrast with the depth enhancement scheme in Eq. 8, Richardt’s scheme employs an additional spatial filter on the upsampled depth image prior to adding it to the temporal prediction of the previous frame. Moreover, instead of PWAS-MCM, Richardt et al. propose MRFI to upsample the low-resolution depth image.<sup>9</sup>

In time-of-flight cameras fast motion leads to increased noise levels. Therefore, the range kernel in the spatial filter of Eq. 15 is augmented with  $\gamma$ . The result is that in image areas with fast motion, i.e.,  $\gamma \rightarrow 0$ , the range kernel of the spatial filter will effectively be disabled, hence depth will be smoothed across color edges which leads to additional noise suppression [31].

### 3.4 Motion-compensated post-filtering (MCPF)

Whereas Cao et al. and Richardt et al. apply motion-compensated filters for depth propagation, Lie et al. [23] take a different approach in their key-frame propagation framework. Instead of applying motion-compensated propagation filters, they first apply block motion compensation to compensate the upsampled previous depth image with the motion vectors that are estimated on the current guidance image taking the previous guidance image as a reference

$$D_{\mathbf{p},n}^{MC} = D_{\mathbf{p},n-1}^{DU}, \tag{16}$$

where  $\mathbf{p}$ , is the motion-compensated pixel  $\mathbf{p}$  in the previous frame  $n - 1$  (Eq. 11). The motion vectors are computed with a blockmatcher.<sup>10</sup> Then the propagated depth image is post-filtered according to

$$D_{\mathbf{p},n}^{MCPF} = \frac{1}{\kappa_{\mathbf{p}}^{MCPF}} \sum_{\mathbf{q} \in N(\mathbf{p}, R^{ip})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p}} - \mathbf{I}_{\mathbf{q}}) G_{\sigma_d}(D_{\mathbf{p},n}^{MC} - D_{\mathbf{q},n}^{MC}) D_{\mathbf{q},n}^{MC}, \tag{17}$$

$$\kappa_{\mathbf{p}}^{MCPF} = \sum_{\mathbf{q} \in N(\mathbf{p}, R^{ip})} G_{\sigma_s}(\mathbf{p} - \mathbf{q}) G_{\sigma_r}(\mathbf{I}_{\mathbf{p}} - \mathbf{I}_{\mathbf{q}}) G_{\sigma_d}(D_{\mathbf{p},n}^{MC} - D_{\mathbf{q},n}^{MC}).$$

This post-filter is a trilateral filter which contains in addition to the Gaussian range and domain kernel, a third Gaussian depth kernel. This kernel computes a weight based on the difference in depth between a kernel pixel  $\mathbf{q}$  and its center  $\mathbf{p}$ . Finally, the output depth image can be obtained by substituting  $D_{\mathbf{p},n}^{MCPF}$  for  $D_{\mathbf{p},n}^{TP}$  in Eq. 8.

<sup>9</sup> MRFI is analyzed and benchmarked in more detail in section 2.6 of Vosters et al. [45].

<sup>10</sup> In our implementation we use 3DRS.

### 3.5 Baseline (NULL)

We include a baseline method in our benchmark that employs depth upsampling only, without temporal propagation. As depth upsampling method we select PWAS-MCM (Sect. 2). Then the output depth image at frame  $n$  becomes

$$D_{\mathbf{p},n}^{OUT} = D_{\mathbf{p},n}^{PWAS-MCM}. \tag{18}$$

## 4 Benchmark setup

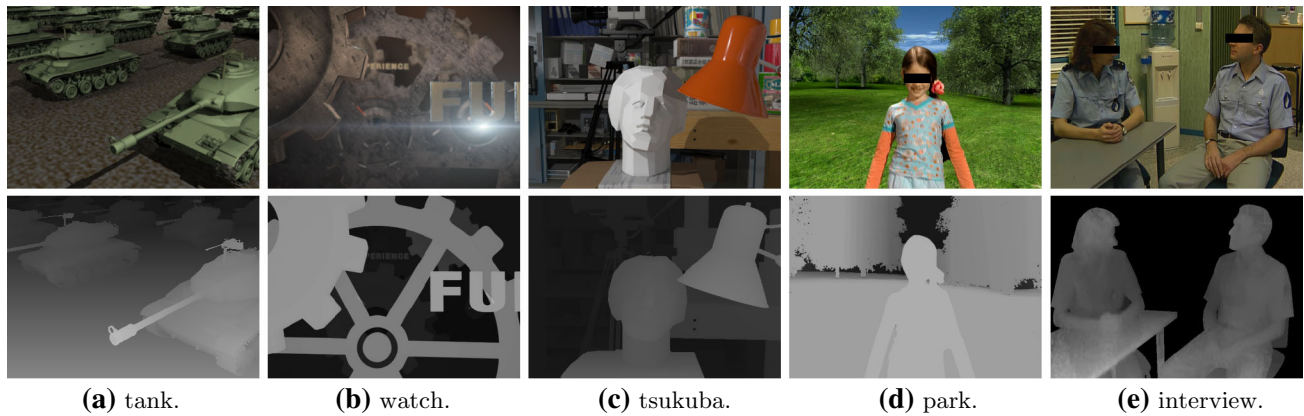
In Sect. 5, we benchmark all temporal post-processing methods shown in Table 2 for various upscale factors and noise levels. In this section, we describe the benchmark setup including the collection and preparation of the test sequences (Sect. 4.1), the evaluation metric (Sect. 4.3), the parameter ranges (Sect. 4.4) and the simulation platform (Sect. 4.5).

### 4.1 Test sequences

We have collected a set of 5 test sequences *Tank*, *Watch*, *Tsukuba*, *Park* and *Interview*. The original *Tank* and *Tsukuba* sequences are publicly available at [30] and [41], respectively. *Watch* has been produced and is owned by Dimenco. It can be downloaded from our website [43] and may be used, published and distributed solely for academic research purposes. *Park* and *Interview* have been extensively used in [4], and the original sequences can be downloaded as Sequence No. 3 and Sequence No. 4 from [5], respectively.<sup>11</sup> All videos except *Interview* are computer generated, hence ground-truth depth is available. *Interview* was shot with a Zcam<sup>TM</sup> by 3DV Systems, and post-processed and enhanced with a manual offline procedure to ensure that the depth discontinuities align with the object borders in the color image.

Since we do a parameter space search on video, the runtime of the experiment will become infeasibly high if we do not take any precautionary measures. To this end, we process and evaluate a fragment of 25 consecutive images with significant motion that we extract from each original test sequence. Each fragment roughly corresponds to 1 s of video. Since all images in a shot are statistically similar, and the average shot length in film and TV typically varies from 2 to 8 s, the fragments should be long enough to

<sup>11</sup> From this website only the original color images can be downloaded. To acquire the original ground-truth depth maps please contact the author of [4].



**Fig. 4** Snapshots of the cropped and resampled color and depth sequences used in the benchmark of Sect. 5. The resampled color and depth images of sequence **a**, **b** and **c**, and the depth images of **d** and **e** can be downloaded from our website [43]. Since we do not own the rights to **d** and **e**, we cannot distribute their color sequences. For a

download link of these sequences, we therefore refer to [5]. In order not to violate any personality rights, we have added censor bars to park and interview. Please note that these censor bars were not present in the benchmark

**Table 3** To avoid overestimating the border effect, for each upsample factor  $U$ , we crop the left, right, top and bottom image border for each test sequence with the amount indicated in the table, prior to computing the depth accuracy

$U$	Crop amount (#pix.)	% Border pix. after crop	% Border pix. 1080 p reference after crop	% Pix. in depth accuracy computation after crop
8	46	15.5	15.7	53.3
4	22	7.9	8.0	76.0
2	11	3.7	4.0	87.6

extrapolate the average depth accuracy of a whole shot. In addition, we resampled the source depth and color images of each test sequence<sup>12</sup> and cropped them to the target resolution of 400 by 300. Table 4 provides the frame numbers of the extracted video fragments, the resolution of the source images, the resampling factor, the crop amounts, and a short description of the scene in each test sequence. Additionally, Fig. 4 shows a snapshot of each sequence and its corresponding depth image. The color and depth images of the extracted fragments of sequence *Tank*, *Watch* and *Tsukuba*, and the depth images of *Park* and *Interview* can be downloaded from our website [43]. Since we do not own the rights to *Park* and *Interview*, we cannot distribute their color sequences. For a download link of these sequences, we therefore refer to [5].

We collected a diverse set of image sequences that contains textured and piecewise smooth image areas. Additionally, the test sequences contain independent object

motion as well as camera motion. Therefore, we believe that our test set is representative for general video material.

## 4.2 Noise model

No ground-truth depth references exist for depth maps generated by ToF cameras, stereo-matchers, range finders or 2-D to 3-D converters. Therefore, we have to simulate low-quality depth images by degrading the ground-truth source depth images in Table 4.

We degrade depth by downscaling the original ground-truth depth images with factors  $U \in \{8, 4, 2\}$  using a Lanczos2 resampling filter. Then we simulate depth noise with the time-of-flight (ToF) based noise model of Edeler et al. [12], who model noise as spatially variant Gaussian noise of which the standard deviation for pixel  $\mathbf{p}$  is given by

$$\sigma n(\mathbf{p}) = \sqrt{\frac{\xi}{I_{\mathbf{p}}^{\text{gray}}}}. \quad (19)$$

This model is more accurate than an independent identically distributed Gaussian noise model, since it is based on the underlying physical noise model of a ToF depth sensor. Note that the noise is added to the 8-bit representation of depth (not disparity), and that data outside the range  $[0, 255]$  are clipped.

<sup>12</sup> We have designed a Nyquist resampling filter with Matlab's filterbuilder. Furthermore, we have selected a stopband attenuation of 60 dB and a transition band of 20 % of the cutoff frequency. In addition, to remove ringing, we apply a Hamming window, and we clip the output of each polyphase filter to the dynamic range within the filter kernel.

**Table 4** Test sequence information

Sequence	Frame no.	Resolution	$\frac{IR}{DR}$	$CL$	$CR$	$CT$	$CB$	Description
Tank	20–44	400 × 300	$\frac{1}{1}$	0	0	0	0	CGI, fast camera motion, highly textured areas, fine details
Watch	45–70	1920 × 1080	$\frac{100}{319}$	92	110	1	38	CGI, independent camera and object motion, soft edges, glare
Tsukuba	335–359	640 × 480	$\frac{50}{71}$	7	44	1	38	CGI, fast camera motion, sharp depth and color edges
Park	95–120	960 × 450	$\frac{100}{159}$	83	121	1	39	CGI, fast camera motion, fast object motion
Interview	396–420	720 × 576	$\frac{25}{41}$	1	39	7	45	Real scene, no camera motion, independent object motion

All test sequences are resampled with interpolation factor  $I'$  and decimation factor  $D'$  and cropped by  $C_L, C_R, C_T, C_B$ , i.e., the number of pixels by which the left right, top and bottom image border are cropped, respectively

Note that  $C_R$  and  $C_B$  are large enough to remove the border effect of the polyphase filters that occurs on the right and bottom image borders, respectively

### 4.3 Depth accuracy

Unlike Min et al., who use the number of bad pixels, we quantify depth accuracy by computing the average PSNR in dB between an upsampled depth image and its corresponding ground-truth depth image over all images in a sequence as follows:

$$DA = \frac{10}{(N_{stop} - N_{start} + 1)} \sum_{n=N_{start}}^{N_{stop}} \log_{10} \left( \frac{D_{max}^2}{\sum_{p \in D_n^{OUT}} (D_{p,n}^{GT} - D_{p,n}^{OUT})^2} \right), \tag{20}$$

where  $D_{p,n}^{GT}$  is the ground-truth depth image,  $D_{max} = 255$  is the maximum depth value,  $N_{start} = 0$  and  $N_{stop} = 24$  are the start and stop frame number, respectively. We select PSNR as metric because large depth errors contribute more to the total error than small depth errors. In contrast, in the bad-pixel metric each depth error, regardless of its size, contributes equally to the total error. Therefore, in our optimization we prevent large depth errors that can cause annoying artifacts.

Unfortunately no quantitative metrics are available to assess the quality of a depth video and couple it to the perceptual quality of the rendered views. In Vosters et al. [45] we used in addition to depth accuracy, also the interpolation quality (IQ) metric. However, since most of our test sequences contain just a single view, we cannot compute IQ. Furthermore, the computation of IQ depends heavily on the textures in the input depth image. Moreover, in the computation of IQ we have to either exclude the occluded areas from the evaluation or apply hole filling. In the former case, we exclude large portions of the image near edges, while in the latter case IQ is biased by the hole filling algorithm. Therefore, in this paper we limit our scope to depth accuracy.

The influence of the border effect for an upsampling filter depends on the upsampling factor  $U$ , kernel radius  $R_{max} = 7$ , and image resolution.<sup>13</sup> As the image size gets

<sup>13</sup> For upsample factor  $U$  and maximum kernel radius  $R_{max}$  the width of the image border which pixels are affected by the border effect is  $U \cdot R_{max}$

smaller the percentage of pixels affected by the border effect gets larger. Since we are working with test sequences that have resolution of  $400 \times 300$ , the percentage of pixels affected by border effect particularly for  $U = 8$  is much larger than that of a 1080 p reference image. Consequently, in order not to overestimate the border effect we crop the left, right, top and bottom borders of the upsampled ( $DOUT$ ) and ground-truth ( $DGT$ ) depth images for each upscale factor with the amount indicated in Table 3 prior to computing depth accuracy. In this way, the percentage of pixels affected by the border effect in our test sequences will never exceed that of the 1080 p reference image. As a consequence, Table 3 column 5 shows that for a high upsample factor we unavoidably use a smaller percentage of the total number of pixels in the test sequence for evaluation than for a low upsample factor.

### 4.4 Parameter space analysis

We perform a parameter space search to find the optimum parameters for each upscale factor and noise level individually. For each method in Table 2 we find the optimum parameters  $\theta^*$  as

$$\theta^* = \arg \max_{\theta \in \Theta} \overline{DA}(\theta), \tag{21}$$

where  $\theta$  is a parameter vector taken from the parameter space  $\Theta$  of the designated method,

$$\overline{DA}(\theta) = \frac{1}{|S|} \sum_{s \in S} DA(s, \theta), \tag{22}$$

where  $DA(s, \theta)$  is the depth accuracy of sequence  $s$  obtained with parameter vector  $\theta$ , and  $S$  is the set of test sequences listed in Table 4. We construct each method’s parameter vector  $\theta$  as the concatenation of all parameters that belong to said method,<sup>14</sup> hence each element of  $\theta$

<sup>14</sup> For example, from Table 2 we see that the  $\theta = (\sigma_s, \sigma_r, \sigma_d, \sigma_f, \phi)$  for JPMC+-3DRS.

**Table 5** The range of each Gaussian scale parameter in column 2 is computed from Eq. 23 using the values of  $\alpha$ ,  $\sigma_0$  and  $k$  in columns 3, 4 and 5, respectively

Parameter	Range	$\alpha$	$\sigma_0$	$k$
$\sigma_r$	{4.5, 6.1, 8.2, 11.1, 14.9, 20.2, 27.2, 36.7, 49.6, 67.0}	0.35	1.5	3
$\sigma_s$	{1.0, 1.3, 1.7, 2.3, 3.0, 4.0, 5.3, 7.0, 9.2, 12.2}	0.32	0.5	2
$\sigma_d$	{1.5, 2.3, 3.4, 5.1, 7.6, 11.4, 17.1, 25.6, 38.4, 57.7}	0.5	1.5	1
$\sigma_f$	{1.2, 1.7, 2.5, 3.7, 5.3, 7.7, 11.2, 16.2, 23.4, 34.0}	0.45	0.6	2
$\zeta$	{0.3, 0.5, 0.8, 1.0, 1.3, 1.5, 1.8, 2.0, 2.3, 2.5, 2.8, 3.0, $10^5$ }	n.a.	n.a.	n.a.
$N_T$	{2, 3}	n.a.	n.a.	n.a.
$\phi$	{0.01, 0.050.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9}	n.a.	n.a.	n.a.

We compute the kernel radius for each method as  $R^{\text{du}} = R^{\text{lp}} = \min(\lceil 3\sigma_d \rceil, R_{\text{max}})$ , where  $R_{\text{max}} = 7$  is the maximum kernel radius

represents a parameter, and each parameter can only take values from the ranges specified in Table 5. Furthermore, we compute the parameter space  $\Theta$  for each method as the set of all permutations of the parameter values in vector  $\theta$ .

To restrict the runtime of the full parameter space search, we restrict the number of values per parameter to 10. In order to get a fair sampling of the parameter space for Gaussian kernels, we logarithmically sample  $\sigma_r$ ,  $\sigma_s$ ,  $\sigma_d$ , and  $\sigma_f$  according to:

$$\sigma(i, \alpha, \sigma_0, k) = \begin{cases} \sigma(i-1, \alpha, \sigma_0) \cdot (1 + \alpha) \cdot k^{(1/i)} & \text{if } i \in \{1, \dots, 9\}, \\ \sigma_0, & \text{if } i = 0 \end{cases}, \quad (23)$$

where  $\alpha$  is a scale factor,  $\sigma_0$  is the start value of the scale parameter and  $k$  is the dimensionality of the input vector.<sup>15</sup> The depth accuracy as a function of  $\sigma$  varies smoothly. Consequently, for linear sampling the relative difference between two consecutive  $\sigma$ s for small values of  $\sigma$  is too large, while for larger values of  $\sigma$  the difference is too small. In the former case the parameter space sampling is too coarse and an optimum may be missed, while in the latter case the sampling is too fine leading to a waste of computational resources. In contrast, by logarithmically sampling the parameter space, the Gaussian will cover  $1 + \alpha$  times more input samples for each new  $\sigma$ . Hence no computational resources are wasted, while still all variations in depth accuracy are captured.

Prior to temporal post-processing, the low-resolution depth images are upsampled with PWAS-MCM (Eq. 8) and MRFI (Eq. 14). For both upsampling methods we select for each upsample factor and noise level, the optimum parameter configurations found in the depth upsampling benchmark of Vosters et al. [45].

For JPMC, Cao et al. use a motion estimation algorithm based on block-matching [1]. For JPMC+, Richardt et al. use the fast optical flow algorithm of Brox et al. [3].

<sup>15</sup> For example,  $k = 3$  for  $\sigma_r$ , since  $G_{\sigma_r}$  accepts a 3-channel input vector.

However, the complexity of this optical flow algorithm is high.<sup>16</sup> A cheaper real-time option is the 3DRS motion estimator<sup>17</sup> proposed by de Haan et al. [17].

In the temporal post-processing benchmark, we will include JPMC and JPMC+ with both motion estimators. 3DRS produces a low-resolution motion vector field. For JPMC and JPMC+ we apply block erosion (de Haan et al. [17]) to upsample the motion vector field to the resolution of the guidance image. However, for MCPF, which performs block motion compensation, we upsample the low-resolution motion vector field with nearest neighbor.

We set the parameters of Brox et al.'s optical flow method to the default parameters in the GPU implementation of OpenCV. Furthermore, in 3DRS we set the penalties for the temporal and update candidates to  $0.625B^2$  and  $2B^2$ , respectively, where  $B = 8$  is the block size. This is the default setting in 3DRS. Additionally, for MCPF we have also experimented with block sizes  $B = 4$  and  $B = 16$ ; however, from the results of our parameter space analysis, we found that  $B = 4$  attains the highest depth accuracy for all upsample factors and noise levels. Therefore, in Sect. 5 we will only show the results for MCPF with  $B = 4$ .

Finally, in our implementations, we truncate all pixel  $\mathbf{j}$  that fall outside the image border, to the nearest integer pixel in the image, prior to using it for fetching depth ( $D_{\mathbf{j}}$ ), color ( $\mathbf{I}_{\mathbf{j}}$ ) etc.

<sup>16</sup> With OpenCV2.4.2's GPU implementation of Brox et al.'s optical flow algorithm, we obtain an average throughput of 0.9 mega pixels per second (MPPS) on an NVIDIA GTX570 GPU. This throughput is significantly lower than most of the efficient depth upsampling methods in Table 6 of Vosters et al. [45]. Hence Brox et al.'s optical flow method results in a significant overhead.

<sup>17</sup> We made an optimized GPU implementation of 3DRS with block erosion that attains a throughput of 810 megapixels per second (MPPS) on an NVIDIA GTX570 GPU. This is significantly higher than the throughput of Brox et al.'s optical flow method.



## 4.5 Simulation platform

We require a large amount of computational resources for the parameter space analysis, because the number of parameter configurations per method is very large (Table 2, column 4). In addition, we have to evaluate this number of parameter configurations for 25 frames, 3 upsample factors, 2 noise levels and 5 test sequences for each of the 9 methods in Table 2. Therefore, we run our experiment on a GPU cluster that consists of 2 PCs, each of which contains 4 NVIDIA GTX 570 GPUs and an Intel i7-960 CPU. Furthermore, we made an implementation of all benchmark methods, and the depth accuracy computation (Eq. 20) in CUDA C++ code.

We divide the computational load in batches. Each batch computes for a designated method, test sequence, upsample factor  $U$  and noise level  $\xi$ , the average depth accuracy of each parameter configuration for each frame and saves it to a text file. During the experiment we continuously run 4 of these batches in parallel on each PC. Furthermore, since the motion vectors and upsampled depth images are the same for all parameter configurations of the temporal filter, we precompute the motion vectors and upsampled depth images for each test sequence, upsample factor and noise level. In this way, we avoid wasting precious computational resources, and we further reduce the runtime of the experiment.

By taking advantage of the GPU cluster, we reduced the runtime of the experiment to 5 days as opposed to 40 days on a single GPU.

## 5 Results and discussion

### 5.1 Quantitative results

The results of the parameter space search described in Sect. 4 are summarized in Fig. 5 and Table 6. Columns 1–5 in Fig. 5 show the optimum depth accuracy  $DA(s, \theta^*)$  (Eq. 22) of all benchmark methods in a bar plot for each individual test sequence, upsample factor  $U$  and noise level  $\xi$ . Column 6 shows the optimum average depth accuracy  $\overline{DA}(\theta^*)$  (Eq. 22) over all image sequences. The bars in columns 1–5 have been ranked in ascending order on  $\overline{DA}(\theta^*)$ . Furthermore, Table 6 complements column 6 of Fig. 5 by showing  $DA(s, \theta^*)$  in 2 decimals, and by ranking all benchmark methods in descending order. In addition, it shows the optimum value of  $\phi$  (Eq. 8), which denotes the contribution of the temporal filter.

From Table 6 we see that when  $\xi = 0.05$  the difference between the baseline method NULL, that has no temporal post-processing, and the best performing temporal post-processing method is 1.07, 1.37 and 1.15 dB for

upsampling factors 2, 4 and 8, respectively. In contrast, for  $\xi = 0.1$  these differences are 1.54, 1.16 and 0.94 dB. Furthermore, we see that all temporal post-processing methods except 3DJBU consistently attain a higher ranking than the baseline method. Therefore, we can conclude that temporal filtering in addition to depth upsampling on average substantially improves the depth accuracy by more than 1 dB. Consequently, temporal filtering is important.

Overall JPMC+ ranks highest among all temporal post-processing methods. It is the best performing method for all depth upsampling factors and noise levels except for  $U = 4$  and  $\xi = 0.1$  where it is ranked second just after RICHARDT. This is because JPMC and MCPF only compensate the kernel position for the motion of the center pixel, while JPMC+ compensates every kernel pixel for motion individually. Table 6 also shows that MCPF consistently ranks higher than JPMC. Nevertheless, the difference between the motion-compensated methods is small, and in most cases less than 0.4 dB. The difference between the motion-compensated temporal post-processing methods gets smaller when the upsample factor increases.

On the other hand, the motion-compensated temporal post-processing methods (JPMC, JPMC+, RICHARDT, MCPF) consistently have a higher ranking than the non-motion-compensated methods (JP, 3DJBU) for all upsampling factors and noise levels. The difference in depth accuracy between the two is significant.

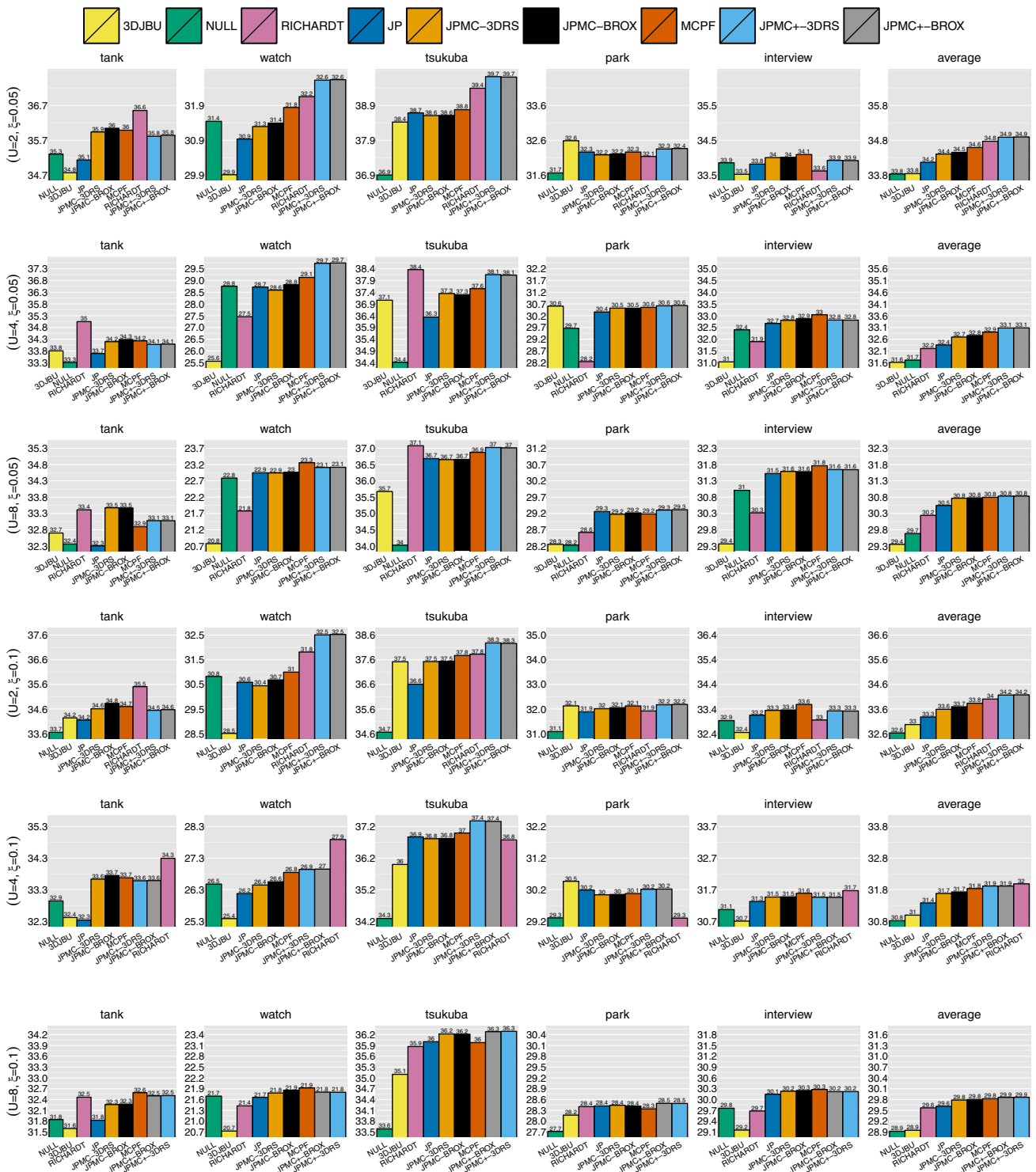
We see that the 3DJBU method performs poorly. It barely attains a higher depth accuracy than the baseline method NULL. For  $U \in \{4, 8\}$  and  $\xi = 0.05$  the depth accuracy of 3DJBU is even lower than that of NULL. 3DJBU performs poorly because of alias and texture copy artifacts that arise from a lack of prefiltering.

Table 6 shows that the differences in depth accuracy between methods that employ Brox et al's optical flow method and 3DRS are negligible. Consequently, instead of Brox et al's optical flow method, we can select the cheaper real-time 3DRS motion estimator with just a marginal loss in depth accuracy.

The good performance of RICHARDT for ( $U = 2$ ,  $\xi \in \{0.05, 0.1\}$ ) and ( $U = 4$ ,  $\xi = 0.1$ ) is due to the application of the additional spatial filter in Eq. 15. In this case, the additional noise suppression outweighs the poor performance of MRFI, RICHARDT's depth upsampling method. Interestingly, this does not hold for  $U = 8$  and ( $U = 4$ ,  $\xi = 0.05$ ), where RICHARDT even ranks below the non-motion-compensated method JP.

Table 6 shows that the temporal filtering strength  $\phi$  on average increases when  $U$  increases. A higher  $\phi$  leads to more temporal averaging and thus more noise suppression. However, we see that  $\phi$  is generally smaller for JP than for the other motion-compensated temporal post-processing methods. This is expected since JP's temporal prediction is





**Fig. 5** Result of the parameter space search described in Sect. 4. Each row shows a bar plot of the optimum depth accuracy ( $DA(s, \theta^*)$ , Eq. 22) of all methods for each image sequence individually for  $U \in \{2, 4, 8\}$  and  $\xi \in \{0.05, 0.1\}$ . The last column shows a bar plot of the optimum average depth accuracy over all image sequences ( $\overline{DA}(\theta^*)$ , Eq. 22). The bars in columns 1–5 have been ranked in ascending order on  $\overline{DA}(\theta^*)$ . For each row the tick scale and tick mark frequency

on the y-axis is the same; however, each axis has a different offset. Furthermore, we have rounded the depth accuracy to one decimal place, and plotted it right above each bar. To complement this figure, we show in Table 6 a ranking of all methods in descending order on  $\overline{DA}(\theta^*)$  in 2 decimals. In addition, this Table also shows the optimum value of  $\phi$  (Eq. 8), which denotes the importance of the temporal filter

**Table 6** Ranking of all methods on the optimum average depth accuracy over all image sequences  $\overline{DA}(\theta^*)$ , Eq. 22) in descending order, for upscale factor  $U \in \{2, 4, 8\}$  and noise levels  $\xi \in \{0.05, 0.1\}$

Rank	Method	$\overline{DA}(\theta^*)$	$\phi$
<i>(U = 2, <math>\xi = 0.05</math>)</i>			
1	JPMC+-BROX	34.90	0.50
2	JPMC+-3DRS	34.89	0.50
3	RICHARDT	34.77	0.50
4	MCPF	34.59	0.30
5	JPMC-BROX	34.45	0.30
6	JPMC-3DRS	34.40	0.30
7	JP	34.17	0.30
8	3DJBU	33.84	N/A
9	NULL	33.83	N/A
<i>(U = 4, <math>\xi = 0.05</math>)</i>			
1	JPMC+-BROX	33.08	0.60
2	JPMC+-3DRS	33.08	0.60
3	MCPF	32.90	0.50
4	JPMC-BROX	32.76	0.50
5	JPMC-3DRS	32.69	0.50
6	JP	32.36	0.30
7	RICHARDT	32.20	0.60
8	NULL	31.71	N/A
9	3DJBU	31.61	N/A
<i>(U = 8, <math>\xi = 0.05</math>)</i>			
1	JPMC+-BROX	30.83	0.60
2	JPMC+-3DRS	30.83	0.60
3	MCPF	30.80	0.60
4	JPMC-BROX	30.78	0.50
5	JPMC-3DRS	30.77	0.50
6	JP	30.54	0.50
7	RICHARDT	30.24	0.70
8	NULL	29.68	N/A
9	3DJBU	29.35	N/A
<i>(U = 2, <math>\xi = 0.1</math>)</i>			
1	JPMC+-BROX	34.17	0.60
2	JPMC+-3DRS	34.17	0.60
3	RICHARDT	34.00	0.60
4	MCPF	33.84	0.50
5	JPMC-BROX	33.69	0.50
6	JPMC-3DRS	33.59	0.50
7	JP	33.28	0.30
8	3DJBU	32.97	N/A
9	NULL	32.63	N/A
<i>(U = 4, <math>\xi = 0.1</math>)</i>			
1	RICHARDT	31.98	0.60
2	JPMC+-BROX	31.91	0.60
3	JPMC+-3DRS	31.91	0.60
4	MCPF	31.83	0.50
5	JPMC-BROX	31.73	0.50

**Table 6** continued

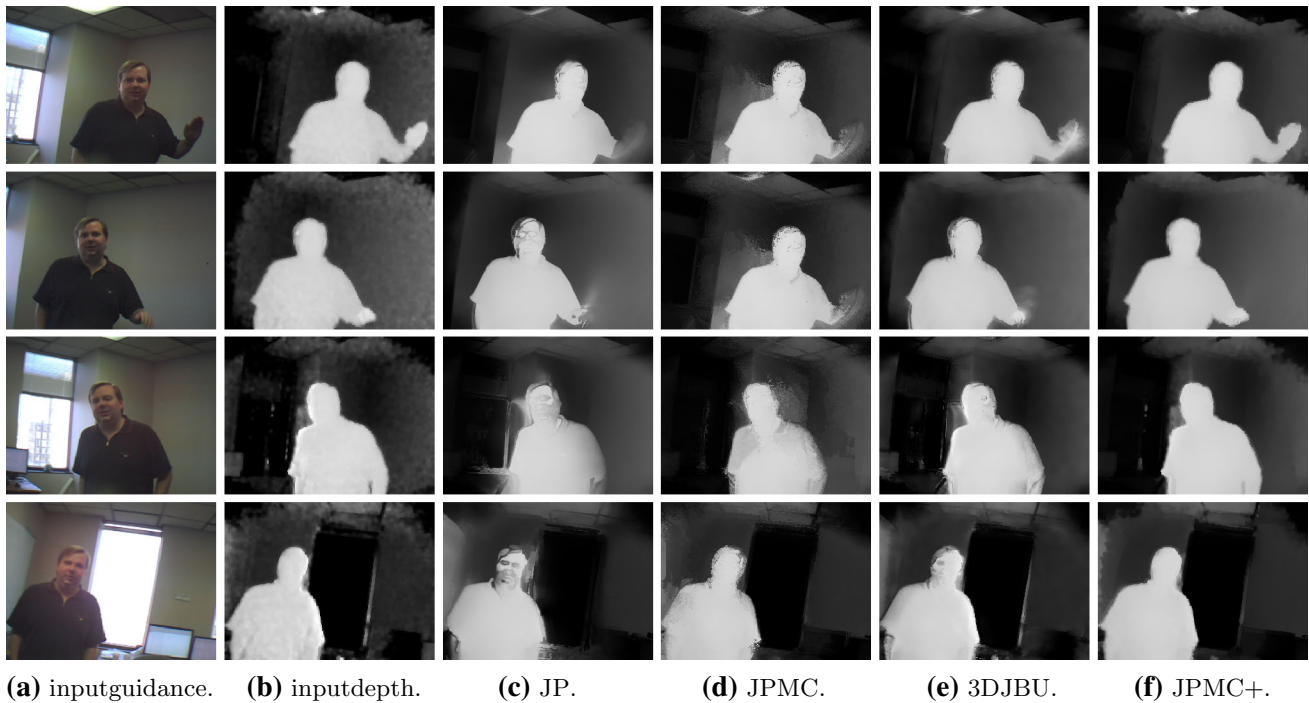
Rank	Method	$\overline{DA}(\theta^*)$	$\phi$
6	JPMC-3DRS	31.68	0.50
7	JP	31.38	0.50
8	3DJBU	30.99	N/A
9	NULL	30.82	N/A
<i>(U = 8, <math>\xi = 0.1</math>)</i>			
1	JPMC+-3DRS	29.86	0.60
2	JPMC+-BROX	29.86	0.60
3	MCPF	29.82	0.50
4	JPMC-BROX	29.80	0.60
5	JPMC-3DRS	29.79	0.60
6	JP	29.61	0.50
7	RICHARDT	29.57	0.70
8	3DJBU	28.95	N/A
9	NULL	28.92	N/A

The values for  $\overline{DA}(\theta^*)$  in this table correspond to column 6 in Fig. 5. For each method we also show the optimum value of  $\phi$  (Eq. 8), which denotes the importance of the temporal filter. For methods NULL and 3DJBU  $\phi$  is not applicable (N/A)

less accurate in image portions that contain motion. Consequently a high  $\phi$  will introduce more depth errors. While a large  $\phi$  increases stability, it also causes problems at scene changes. In that case a scene change detector is required that resets  $\phi$  to zero only at scene changes.

Moving objects in adjacent frames have the same textures; however, their depth may decrease/increase if the objects are moving towards/away from the camera. None of the temporal post-processing methods take this depth change into account, and as a consequence, they cannot properly track the depth change of corresponding depth areas that have been established by motion estimation. This causes depth errors and hence reduces the depth accuracy. Fu et al. [14] and Kim et al. [20] only partially solve this problem by filtering stationary image areas only. However, our test sequences have both camera and object motion and hence both approaches will not work. The inability to track depth changes in moving objects could explain why the improvement in depth accuracy of temporal post-processing methods is limited to on average 1dB. We consider the tracking of depth changes to be outside the scope of this paper and leave it as future work.

We should note that the depth accuracy metric does not explicitly take into account perceptual factors such as flicker which could significantly degrade the image quality of the rendered multiview video. As a consequence, when we qualitatively compare the output depth video of the baseline method NULL, which does not apply a temporal post-processing, with the other benchmark methods that do apply temporal post-processing, we see



**Fig. 6** Qualitative comparison of temporal post-processing methods on real depth data recorded by a time-of-flight (ToF) camera. Row 1, 2, 3 and 4 show frame no. 116, 143, 163 and 384, respectively, of

depth sequence  $ms$  which can be downloaded from [40] ( $ms$ ). All sequences have been processed with the same set of parameters:  $\phi = 0.9$ ,  $\sigma r = 0.018$ ,  $\sigma s = 12.1$ ,  $Rtp = 7$ ,  $\sigma d = 0.03$ ,  $\sigma f = 34$

that especially in the presence of noise depth flickering is far more suppressed by the methods that do apply temporal post-processing. Thus from a perceptual point of view temporal filtering may be more important than the depth accuracy metric in Fig. 5 suggests. However, as mentioned before no quantitative metrics are available that can measure perceptual quality of depth maps. In addition, a design space exploration based on a perception test is infeasible.

Looking at the scale offsets of each individual bar plot in a row of Fig. 5, we see that the spread in optimum depth accuracy  $DA(s, \theta^*)$  is high for each method. This is because the sample size is small, i.e., we just use 5 test sequences. Since the sample variance is inversely proportional to the sample size, we could reduce the spread by including more test sequences in our benchmark. However, very few sequences with ground-truth are available and moreover the runtime of the experiment would become too high. Nonetheless, we observe the same trends for all upsample factors and noise levels, namely:

1. Temporal post-processing leads to a substantial improvement in depth accuracy by roughly 1dB on average for all upsampling factors.
2. The highest depth accuracy is attained by JPMC+.
3. The motion-compensated temporal filters rank consistently higher than the non-motion-compensated filters.

4. The performance of the motion estimators Brox and 3DRS for is approximately equal for all temporal post-processing methods.

Additionally, Fig. 5 shows that the ranking on  $\overline{DA}(\theta)$  (column 6) largely resembles the ranking in the bar plots for each individual image (columns 1–5). Whenever the ranking in the individual bar plots for each image deviates from column 6, the differences in depth accuracy are generally negligible. Therefore, we believe that increasing the number of test sequences will not alter the ranking much.

## 5.2 Qualitative results on real depth data

Figure 6 qualitatively compares JP, JPMC, 3DJBU and JPMC on real depth images acquired by a time-of-flight (ToF) depth camera. For this experiment, we use the depth sequence  $ms$  provided by Wang et al. [46] which is available for download from [40]. This sequence consist of synchronized 320 by 240 RGB color and depth images with the same resolution that have been recorded by ZCam from 3DV Systems. The depth and color images have been internally aligned by ZCam and the depth has 256 levels. The sequence contains both fast camera and object motion.

Figure 6 shows that all methods significantly suppress the noise. However, JP and 3DJBU show severe texture copy artifacts, in which the texture of the input guidance image

reappears in the output depth image. Furthermore, we see that the non-motion-compensated temporal post-processing method JP suffers from severe ghosting in fast moving image areas. To a lesser extent ghosting is also visible in JPMC and 3DJBU. In contrast, JPMC+ does not suffer from ghosting at all because (1) it compensates each kernel pixel for motion and (2) it adaptively reduces the amount of temporal filtering in fast moving image portions. Moreover, the texture copy artifact does not appear in JPMC+.

## 6 Complexity

### 6.1 Throughput

The complexity in big-O notation is the same for each method, namely  $\mathcal{O}((2R + 1)^2)$ , where  $R$  is the kernel radius. Additionally, we measure the complexity by computing the number of megapixels per second<sup>18</sup> for the depth upsampling method PWAS-MCM (Sect. 2), each temporal post-processing method, and a pipelined implementation of PWAS-MCM with any of the temporal post-processing methods.

We implemented PWAS-MCM and all benchmarked temporal post-processing methods in CUDA C++ code, and ran it on our GPU cluster (Sect. 4.5). We aimed for an efficient and versatile implementation; however, we made no attempt to squeeze every last drop of performance out of the GPU. In contrast with the parameter space analysis in which we used double precision floating point arithmetic for accuracy, we use single precision arithmetic in our complexity analysis to get a higher throughput<sup>19</sup> at the cost of a lower accuracy.

In addition, we made multithreaded pipelined implementations of PWAS-MCM, the depth upsampling method, with any of the other temporal post-processing methods. For these implementations we ran PWAS-MCM and the temporal post-processing method in parallel on GPU1 and GPU2 in our cluster, respectively. Compared to a sequential implementation, these pipelined implementations require just twice the amount of resources, but will theoretically reduce the processing time per frame from  $T^{\text{du}} + T^{\text{tp}}$  to  $\max(T^{\text{du}}, T^{\text{tp}})$ , where  $T^{\text{du}}$  and  $T^{\text{tp}}$  are the processing times per frame for PWAS-MCM and one of the temporal post-processing methods, respectively. The measured times of  $T^{\text{du}}$  and  $T^{\text{tp}}$  include data transfer overhead between CPU and GPU. Furthermore, the measured processing times of the pipelined implementations that we refer to as DU+TPP, include the data transfer overhead from CPU to GPU and vice versa, as well as from GPU1 to GPU2. We compute the

<sup>18</sup> To give an indication, a 720 and 1080 p image contain 0.9 and 2.1 megapixels, respectively.

<sup>19</sup> The peak double precision floating point performance in GFLOPS is half that of the peak single precision floating point performance on the target NVIDIA GTX570 GPU.

throughput by dividing the number of pixels per frame by the average processing time per frame.

In Fig. 7 we present the complexity of each benchmark method in a bar plot for various kernel radii. This figure shows that the throughputs of JP, JPMC-3DRS, and MCPF are significantly higher than the throughput of PWAS-MCM for all kernel radii. When the kernel radii of PWAS-MCM and the temporal post-processing methods are larger or equal to 2, i.e.,  $R^{\text{du}} \geq 2$  and  $R^{\text{tp}} \geq 2$ , respectively, we see that the throughput of JPMC+3DRS, drops a little below the throughput of PWAS-MCM. In addition, Fig. 5 shows that JP has the highest throughput among all temporal post-processing methods. Furthermore, the throughputs of JPMC-3DRS and MCPF are approximately equal. In contrast, the throughput JPMC+3DRS is significantly lower due to the additional Gaussian kernel  $G_{\sigma_d}(\cdot)$  (Eq. 15), and the irregular memory access pattern that is caused by compensating each kernel pixel for motion.

While the overhead of the 3DRS motion estimator on the throughputs of JPMC-3DRS, JPMC+3DRS and MCPF is negligible, the throughputs of JPMC-BROX, JPMC+3DRS and RICHARDT are severely constrained by Brox et al.'s optical flow estimator.<sup>20</sup> As a result the throughputs of the Brox's based temporal post-processing methods are at least 35 to 9 times lower than their 3DRS-based counterparts.

Figure 7 also shows that for kernel radii larger than 2, 3DJBU has a significantly lower throughput than JP, JPMC-3DRS and MCPF. Moreover, for kernel radii larger than 3, even the throughput of JPMC+3DRS is substantially higher than 3DJBU. This is because the temporal filter aperture of 3DJBU comprises an additional frame.

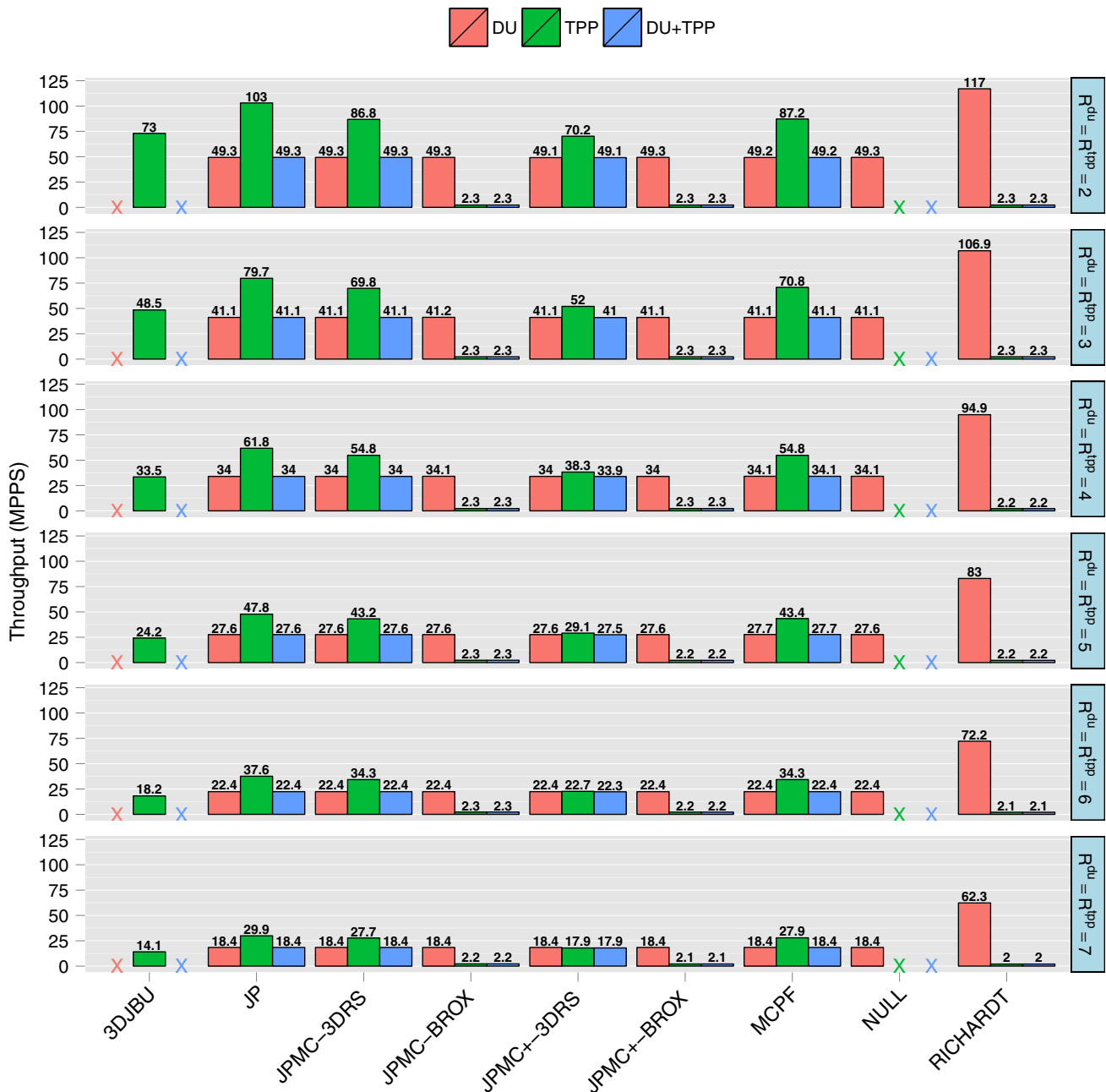
Finally, note that the throughput of the depth upsampling method of RICHARDT is higher than the throughput for PWAS-MCM in the other benchmark methods. This is because RICHARDT uses the cheap multi-resolution fill-in (MRFI) approach described in [45] rather than PWAS-MCM. However, the total throughput of RICHARDT is low since it applies Brox's optical flow estimator.

### 6.2 Performance limits

To improve the throughput of the depth enhancement methods, we must analyze whether their GPU implementations are bound by computation, memory bandwidth or latency. A method is bound by computation when the utilization of the computing hardware inside a streaming multiprocessor<sup>21</sup> is close to 100 %. A method is bound by

<sup>20</sup> The throughput of our implementation of 3DRS with block erosion is 810MPPS, while the throughput of OpenCV's 2.4.2's implementation of Brox et al.'s optical flow estimation algorithm with the default parameters attains not even 1MPPS.

<sup>21</sup> The computing hardware in a streaming multiprocessor consists of an arithmetic logic units, floating point units, load/store units, special function units and double precision units.



**Fig. 7** Complexity in megapixels per second (MPPS) as a function of the kernel radius. The red bar denotes the throughput of the depth upsampling (DU) method (Sect. 2) which for the temporal post-processing methods JP, JPMC, JPMC and JPMC+, MCPF and NULL is PWAS-MCM, except for RICHARDT that use the multi-resolution fill-in (MRFI) approach [31]. The green bar denotes the throughput of the temporal post-processing method (TPP), i.e., 3DJBU, JP, JPMC-3DRS, JPMC-BROX, JPMC+-3DRS, JPMC+-BROX, MCPF and RICHARDT. Finally, the blue bar represents the throughput of the

pipelined implementation (DU+TPP) of the depth upsampling method and temporal post-processing method. Since in 3DJBU upsampling and temporal post-processing is done at the same time, we replace the bars for depth upsampling and the pipelined implementation (DU+TPP) with a red and blue cross, respectively. Similarly for NULL, the temporal post-processing method does not exist, therefore we replace the bars for TPP and DU+TPP with a green and blue cross, respectively

memory bandwidth when the bandwidth, i.e., utilization, of the DRAM or shared or texture or L1 or L2 or cache memories is close to the maximum physical limits of the GPU

device. Finally, a method is latency bound when the latency of reads and writes to the DRAM memory cannot be hidden by the execution of instructions from other warps that have



**Table 7** Compute and memory utilization of the GPU implementations of the depth enhancement algorithms in this paper

Method	Compute utilization (%)	Memory utilization (%)	Limitation
3DJBU	82	75	Balanced
JP	45	95	Memory bound
JPMC	50	95	Memory bound
JPMC+	45	95	Memory bound
MCPF	45	95	Memory bound
RICHARDT	50	85	Memory bound
PWAS-MCM	42	85	Memory bound

We measured these figures on an NVIDIA GTX570 GPU

not yet stalled. As a consequence, the overlap between memory reads and writes and the execution of instructions by the cores in a streaming multiprocessor is low.

Table 7 shows the compute and memory utilization of the GPU implementations of the depth enhancement algorithms analyzed in this paper. All algorithms, except 3DJBU, are memory bound. Consequently, the only way to get a better performance is to either optimize the performance of the GPU implementation by optimizing the memory access pattern, or if this is not possible switch to a GPU device with a higher memory bandwidth. On the other hand, the 3DJBU kernel is balanced, meaning that the compute and memory utilization are approximately equal.

### 6.3 Performance on state-of-the-art GPU architectures

For kernel radii smaller than 4, we already achieve real-time performance for 720p video on an NVIDIA GTX570 GPU for JP, JPMC-3DRS and MCPF. The GTX570 has a peak bandwidth of just 152 giga byte per second (GB/s) and contains 480 cores. Since the depth enhancement algorithms are all memory bound, it is reasonable to assume that state-of-the-art GPU architectures such as the NVIDIA GTX 780 Ti, which features a peak bandwidth of 336.4 GB/s and 2880 cores, can easily attain at least a twofold increase in throughput by running the same code.

## 7 Conclusion

In this paper we proposed a novel filter-based depth upsampling method PWAS-MCM, which is a combination of PWAS [15], and the MCM framework of Min et al. [24]. We showed that PWAS-MCM attains the highest average depth accuracy over all test images in comparison with the other efficient depth upsampling methods in the benchmark of Vosters et al. [45].

Subsequently, we analyzed the performance of efficient filter-based temporal post-processing methods on video by conducting a parameter space search to find the optimum set

of parameters  $\theta^*$  and the optimum average depth accuracy  $\overline{DA}(\theta^*)$  over all test sequences for upscale factors  $U \in \{2, 4, 8\}$  and noise levels  $\zeta \in \{0.05, 0.1\}$  independently.

The results show that the temporal post-processing methods consistently attain a higher ranking than the baseline method without temporal post-processing and that the overall improvement in depth accuracy is on average roughly 1dB. Thus temporal post-processing is an important component of depth enhancement. However, the depth accuracy metric that we used does not explicitly take into account perceptual factors such as flicker. Thus from a perceptual point of view temporal filtering may be more important than the depth accuracy metric suggests.

Temporal post-processing methods cannot track depth changes of objects moving towards or away from the camera. Compensating kernel pixels for depth changes may potentially improve the depth accuracy even more. However, we consider the tracking of depth changes to be outside the scope of this paper and leave it as future work.

Furthermore, the results in Fig. 5 show that overall the motion-compensated temporal post-processing methods consistently attain a higher ranking than the non-motion-compensated methods. In addition, we found that the choice between the motion estimators 3DRS and Brox is not of critical importance.

We have also analyzed in Sect. 6, the complexity of PWAS-MCM and all other temporal post-processing methods that we included in our benchmark. In short, JP, JPMC-3DRS, and MCPF have significantly higher throughput for all kernel radii than PWAS-MCM. In addition, the throughputs of JPMC-BROX, JPMC+-BROX and RICHARDT are severely constrained by Brox et al.'s optical flow estimator, which impedes real-time performance. Furthermore, the throughput of 3DJBU is significantly lower than JP, JPMC-3DRS and MCPF. Finally, we showed that temporal post-processing methods can be pipelined with PWAS-MCM without decreasing the throughput of the total depth enhancement system.

In conclusion, JPMC-3DRS gives the best tradeoff between depth accuracy (Fig. 5; Table 6) and cost (Fig. 7). However, if resources are very scarce then the best option

is JP, which features the lowest complexity with just a limited decrease in depth accuracy.

For future work, we consider modifying the depth accuracy metric to penalize the temporal depth discontinuities that are not present in the ground-truth. Additionally, we plan to adapt  $\phi$  (Eq. 8) locally for each pixel, based on the depth difference between the propagated and upsampled depth.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Ahmad, I., Zheng, W., Luo, J., Liou, M.: A fast adaptive motion estimation algorithm. *IEEE Trans. Circuits Syst. Video Technol.* **16**(3), 420–438 (2006). doi:[10.1109/TCSVT.2006.870022](https://doi.org/10.1109/TCSVT.2006.870022)
- Bae, K., Kyung, K.M., Kim, T.C.: Depth upsampling method using the confidence map for a fusion of a high resolution color sensor and low resolution time-of-flight depth sensor. In: *Three-Dimensional Imaging, Interaction, and Measurement, Proceedings SPIE, Burlingame, California, USA*, vol. 7864, pp. 786,406–786,406–6 (2011)
- Brox, T., Bruhn, A., Papenbergh, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: *European Conference on Computer Vision, Prague, Czech Republic*, vol. 3024, pp. 25–36 (2004). <http://mb.informatik.uni-freiburg.de/Publications/2004/Br04a>
- Cao, X., Li, Z., Dai, Q.: Semi-automatic 2d-to-3d conversion using disparity propagation. *IEEE Trans. Broadcast.* **57**(2), 491–499 (2011). doi:[10.1109/TBC.2011.2127650](https://doi.org/10.1109/TBC.2011.2127650)
- Cao, X., Li, Z., Dai, Q.: Test sequences with ground-truth depth maps (2011). <http://media.au.tsinghua.edu.cn/2Dto3D/testsequence.html>
- Chan, D., Buisman, H., Theobalt, C., Thrun, S.: A noise-aware filter for real-time depth upsampling. *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, pp. 1–12. Marseille, France (2008)
- Cheng, C.M., Lin, S.J., Lai, S.H.: Spatio-temporally consistent novel view synthesis algorithm from video-plus-depth sequences for autostereoscopic displays. *IEEE Trans. Broadcast.* **57**(2), 523–532 (2011)
- Choi, J., Min, D., Kim, D., Sohn, K.: 3d jbu based depth video filtering for temporal fluctuation reduction. In: *International Conference on Image Processing, Hong Kong*, pp. 2777–2780 (2010)
- De Silva, D., Fernando, W., Kodikaraarachchi, H., Worrall, S., Kondo, A.: A depth map post-processing framework for 3d-tv systems based on compression artifact analysis. *Sel. Top. Signal Process. IEEE J. PP* (99):1 (2011). doi:[10.1109/JSTSP.2011.2165274](https://doi.org/10.1109/JSTSP.2011.2165274)
- Deng, H., Yu, L., Qiu, J., Zhang, J.: A joint texture/depth edge-directed upsampling algorithm for depth map coding. In: *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pp. 646–650 (2012). doi:[10.1109/ICME.2012.2](https://doi.org/10.1109/ICME.2012.2)
- Diebel, J., Thrun, S.: An application of markov random fields to range sensing. In: *Proceedings of Advanced Neural Information Processing Systems, Cambridge, MA*, pp. 291–298 (2005)
- Edeler, T., Ohliger, K., Hussmann, S., Mertins, A.: Time-of-flight depth image denoising using prior noise information. In: *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pp. 119–122 (2010). doi:[10.1109/ICOSP.2010.5656819](https://doi.org/10.1109/ICOSP.2010.5656819)
- Fehn, C.: A 3d-tv system based on video plus depth information. *Thirty-Seventh Asilomar Conference on Signals, Systems and Computers* vol. 2, pp. 1529–1533 (2003)
- Fu, D., Zhao, Y., Yu, L.: Temporal consistency enhancement on depth sequences. In: *Picture Coding Symposium (PCS), 2010*, pp. 342–345 (2010). doi:[10.1109/PCS.2010.5702503](https://doi.org/10.1109/PCS.2010.5702503)
- Garcia, F., Mirbach, B., Ottersten, B., Grandidier, F., Cuesta, A.: Pixel weighted average strategy for depth sensor data fusion. In: *International Conference on Image Processing, Hong Kong*, pp. 2805–2808 (2010)
- de Haan, G.: Television display processing: Past & future. *International Conference on Consumer Electronics*, pp. 1–2. Digest of Technical Papers, Las Vegas, NV (2007)
- de Haan, G., Biezen, P., Huijgen, H., Ojo, O.: True-motion estimation with 3-d recursive search block matching. *IEEE Trans. Circuits Syst. Video Technol.* **3**(5), 368–379, 388 (1993)
- Jager, F., Balle, J.: Median trilateral loop filter for depth map video coding. In: *Picture Coding Symposium (PCS), 2012*, pp. 61–64 (2012). doi:[10.1109/PCS.2012.6213286](https://doi.org/10.1109/PCS.2012.6213286)
- Jung, S.W.: Enhancement of image and depth map using adaptive joint trilateral filter. *IEEE Trans. Circuits Syst. Video Technol.* **23**(2), 258–269 (2013). doi:[10.1109/TCSVT.2012.2203734](https://doi.org/10.1109/TCSVT.2012.2203734)
- Kim, S.Y., Cho, J.H., Koschan, A., Abidi, M.: Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2358–2361 (2010). doi:[10.1109/ICPR.2010.577](https://doi.org/10.1109/ICPR.2010.577)
- Klimaszewski, K., Wegner, K., Domanski, M.: Influence of views and depth compression onto quality of synthesized. In: *ISO/IEC JTC1/SC29/WG11 MPEG/M16758, London, UK* (2009)
- Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. *ACM Trans. Graph.* **26**(3), 96 (2007)
- Lie, W.-N., Chen C.Y.: Motion picture depth information processing system and method. <http://www.google.com/patents/US20120194642> (2012)
- Min, D., Lu, J., Do, M.N.: Depth video enhancement based on weighted mode filtering. *IEEE Trans. Image Process.* **21**(3), 1176–1190 (2012)
- Mueller, M., Zilly, F., Kauff, P.: Adaptive cross-trilateral depth map filtering. In: *3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pp. 1–4 (2010). doi:[10.1109/3DTV.2010.5506336](https://doi.org/10.1109/3DTV.2010.5506336)
- Mueller, M., Zilly, F., Riechert, C., Kauff, P.: Spatio-temporal consistent depth maps from multi-view video. In: *3DTV Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pp. 1–4, (2011). doi:[10.1109/3DTV.2011.5877221](https://doi.org/10.1109/3DTV.2011.5877221)
- Oh, K.J., Yea, S., Vetro, A., Ho, Y.S.: Depth reconstruction filter and down/up sampling for depth coding in 3-d video. *Signal Process. Lett. IEEE* **16**(9), 747–750 (2009). doi:[10.1109/LSP.2009.2024112](https://doi.org/10.1109/LSP.2009.2024112)
- Park, J., Kim, H., Tai, Y.W., Brown, M., Kweon, I.: High quality depth map upsampling for 3d-tof cameras. In: *IEEE International Conference on Computer Vision*, pp. 1623–1630 (2011)
- Primesense. <http://www.primesense.com>
- Richardt, C.: Synthetic stereo videos with ground-truth disparity maps (2010). <http://richardt.name/publications/dcbgrid/datasets/>

31. Richardt, C., Stoll, C., Dodgson, N.A., Seidel, H.P., Theobalt, C.: Coherent spatiotemporal filtering, upsampling and rendering of rgbz videos. *Comput. Graph. Forum* **31**(2), 247–256 (2012)
32. Riemens, A., Gangwal, O., Barenbrug, B., Berretty, R.P.M.: Multistep joint bilateral depth upsampling. In: *Visual Communications and Image Processing 2009, Proceedings SPIE, San Jose, CA, USA, vol. 7257, pp. 72,570M–72,570M–12* (2009)
33. Scharstein, D., Szeliski, R.: Middlebury stereo vision page (2001). <http://vision.middlebury.edu/stereo/>
34. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* **47**(1–3), 7–42 (2002)
35. Schwarz, S., Olsson, R., Sjostrom, M., Tourancheau, S.: Adaptive depth filtering for hevc 3d video coding. In: *Picture Coding Symposium (PCS), 2012, pp. 49–52, (2012). doi: 10.1109/PCS.2012.6213283*
36. Schwarz, S., Sjstrm, M., Olsson, R.: Depth map upscaling through edge-weighted optimization. In: *Three-Dimensional Image Processing and Applications II, Proceedings SPIE, Burlingame, California, USA, vol. 8290, pp. 829,008–829,008–8* (2012)
37. Smolic, A., Mueller, K., Stefanoski, N., Ostermann, J., Gotchev, A., Akar, G., Triantafyllidis, G., Koz, A.: Coding algorithms for 3dtv—a survey. *IEEE Trans. Circuits Syst. Video Technol.* **17**(11), 1606–1621 (2007)
38. Soh, Y., Sim, J.Y., Kim, C.S., Lee, S.U.: Superpixel-based depth image super-resolution. In: *Three-Dimensional Image Processing and Applications II, Proceedings SPIE, Burlingame, California, USA, vol. 8290, pp. 82, 900D–82, 900D–10* (2012)
39. Tallon, M., Babacan, S., Mateos, J., Do, M., Molina, R., Katsaggelos, A.: Upsampling and denoising of depth maps via joint-segmentation. In: *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, pp. 245–249* (2012)
40. University of Kentucky: Time-of-flight depth sequences (2011). <http://vis.uky.edu/~gravity/Research/ToFMatting/ToFMatting.htm>
41. University of Tsukuba: Tsukuba stereo dataset (2012). <http://www.cvlab.cs.tsukuba.ac.jp/dataset/tsukubastereo.php>
42. Varekamp, C., Barenbrug, B.: Improved depth propagation for 2d to 3d video conversion using key-frames. In: *European Conference on Visual Media Production, London, UK, pp. 1–7* (2007)
43. Vosters, L.: Depth enhancement test sequences (2013). <https://sites.google.com/site/depthenhancement/>
44. Vosters, L., de Haan, G.: Efficient and stable sparse to dense conversion for automatic 2D-to-3D conversion. *IEEE Trans. Circuits Syst. Video Technol.* **23**(3), 373–386 (2013). doi:10.1109/TCSVT.2012.2203747
45. Vosters, L., Varekamp, C., de Haan, G.: Evaluation of efficient high quality depth upsampling methods for 3dtv. In: *Three-Dimensional Image Processing and Applications II, Proceedings SPIE, Burlingame, California, USA, vol. 8650, pp. 865,005–865,005–13, (2013). doi:10.1117/12.2005094*
46. Wang, L., Gong, M., Zhang, C., Yang, R., Zhang, C., Yang, Y.H.: Automatic real-time video matting using time-of-flight camera and multichannel poisson equations. *Int. J. Comput. Vis.* **97**(1), 104–121 (2012). doi:10.1007/s11263-011-0471-x
47. Yang, Q., Yang, R., Davis, J., Nister, D.: Spatial-depth super resolution for range images. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. Minneapolis, Minnesota, USA (2007)
48. Zhang, L., Tam, W.: Stereoscopic image generation based on depth images for 3d tv. *IEEE Trans. Broadcast.* **51**(2), 191–199 (2005)
49. Zhang, L., Vazquez, C., Knorr, S.: 3d-tv content creation: automatic 2d-to-3d video conversion. *IEEE Trans. Broadcast.* **57**(2), 372–383 (2011)

**Luc Vosters** received his BSc and MSc degree in electrical engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands in 2007 and 2009 respectively. He is currently a PhD candidate in the Electronic Systems group at the department of Electrical Engineering at the same university, on a grant from Axon Digital Design. His current research interests include algorithms for real-time video/image processing, image analysis, computer vision and high performance GPU computing.

**Chris Varekamp** obtained his MSc. in forestry (cum-laude) and Ph.D. (cum-laude) on interferometric radar observation of forests both from Wageningen University in The Netherlands. From 2001 till 2010 he worked as a senior scientist for Philips Research mainly focussing on the conversion of 2D video to 3D video for the development of auto-stereoscopic TV. From 2010 till 2013 he worked as a senior scientist for Philips Consumer Lifestyle and later for TP Vision, a joint venture of Philips and TPV where he investigated computer vision algorithms for camera-based control. Since 2013 he is again employed as a senior scientist at Philips Research. He has a broad research interest and expertise in remote sensing, geo-statistics, image analysis, computer vision and machine learning. He has contributed to several patent applications and has written various scientific publications on applications in remote sensing, geo-statistics and 3D autostereoscopic TV. In 2001 he received an award for ‘best PhD thesis 2001’ from the Remote Sensing and Photogrammetry Society in the UK. He has served as advisor and member of several Ph.D committees on the topics of remote sensing and computer vision respectively.

**Gerard de Haan** received B.Sc., M.Sc., and Ph.D. degrees from Delft University of Technology in 1977, 1979 and 1992, respectively. He joined Philips Research in 1979 to lead research projects in the area of video processing/analysis. From 1988 till 2007, he has additionally taught post-academic courses for the Philips Centre for Technical Training at various locations in Europe, Asia and the US. In 2000, he was appointed “Fellow” in the Video Processing and Analysis group of Philips Research Eindhoven, and “Full- Professor” at the Eindhoven University of Technology. He has a particular interest in algorithms for motion estimation, video format conversion, image sequence analysis and computer vision. His work in these areas has resulted in 3 books, 2 book chapters, 170 scientific papers and more than 130 patent applications, and various commercially available ICs. He received 5 Best Paper Awards, the Gilles Holst Award, the IEEE Chester Sall Award, bronze, silver and gold patent medals, while his work on motion received the EISA European Video Innovation Award, and the Wall Street Journal Business Innovation Award. Gerard de Haan serves in the program committees of various international conferences on image/video processing and analysis, and has been a “Guest-Editor” for special issues of Elsevier, IEEE, and Springer.