PERFORMANCE OPTIMIZATIONS FOR PATCHMATCH-BASED PIXEL-LEVEL MULTIVIEW INPAINTING

Shao-Ping Lu, Beerend Ceulemans, Adrian Munteanu, Peter Schelkens

Department of Electronics and Informatics,
Vrije Universiteit Brussel
Pleinlaan 2, Brussels, Belgium

Department of Future Media and Imaging,
iMinds V. Z. W.
G. Crommenlaan 8, Ghent, Belgium

e-mail: {splu, bceulema, acmuntea, pschelke}@etro.vub.ac.be

ABSTRACT

As 3D content is becoming ubiquitous in today's media landscape, there is a rising interest for 3D displays that do not demand wearing special headgear in order to experience the 3D effect. Autostereoscopic displays realize this by providing multiple different views of the same scene. It is however unfeasible to record, store or transmit the amount of data that such displays require. Therefore there is a strong need for real-time solutions that can generate multiple extra viewpoints from a limited set of originally recorded views. The main difficulty in current solutions is that the synthesized views contain disocclusion holes where the pixel values are unknown. In order to seamlessly fill-in these holes, inpainting techniques are being used. In this work we consider a depth-based pixel-level inpainting system for multiview video. The employed technique operates in a multi-scale fashion, fills in the disocclusion holes on a pixel-per-pixel basis and computes approximate Nearest Neighbor Fields (NNF) to identify pixel correspondences. To this end, we employ a multi-scale variation on the well-known PatchMatch algorithm followed by a refinement step to escape from local minima in the matching-cost function. In this paper we analyze the performance of different cost functions and search methods within our existing inpainting framework.

Index Terms— view synthesis, multiview inpainting, PatchMatch.

1. INTRODUCTION

3D video applications are becoming commonplace as the technology of 3D acquisition devices such as stereo cameras and depth cameras is maturing. The stereo format where users are required to wear special glasses is already widely adopted by both digital cinemas and broadcasting companies. The 3D experience offered by this type of 3D displays is however very limited in the sense that users cannot experience the motion parallax effect which is needed for a genuine 3D feel. Therefore many experts believe that the future of 3D lies in display devices that can offer this motion parallax and moreover without requiring users to wear any special headgear. This is the idea behind the so-called autostereoscopic displays [1]. Such displays are able to show different content, depending on the position of a viewer relative to the screen. This way, they can accommodate the motion parallax effect, thus providing a more realistic 3D feeling compared to simple stereo solutions.

In order to display high quality 3D content by making use of multiview video, a sufficient amount of different views needs to be available for the display. If the number of views is too low, users that move with respect to the screen will notice harsh discontinuous view changes instead of smooth natural looking transitions. However, in order to provide such smooth transitions, a large number of views are typically needed. This imposes an enormous burden on the acquisition, storage and transmission of multiview video data.

To reduce the amount of separate video streams that needs to be recorded and transmitted, one needs to synthesize many additional viewpoints based on a limited set of originally recorded views. This problem is already well-studied and resulted in so-called depth-based image rendering (DIBR) [2] techniques. The resulting synthesized images after 3D warping contain holes, corresponding to pixels of which the value is unknown. These holes originate either from inaccuracies in the process – the real 3D scene is continuous while its digital recordings are discrete – or from occlusions as parts of the scene that were not visible in the original view may become visible in the synthetic viewpoint. These holes need to be filled-in seamlessly so that the resulting synthetic frames look natural; furthermore, since playback of video should happen instantaneously, view synthesis should be performed in real-time.

The problem of filling in missing values in images is called inpainting and numerous such methods have been proposed in the literature in the past – e.g. [3-12]. Among them one distinguishes the class of so-called exemplar-based or texture-based methods [9, 12]. In this paper, we adopt our depth-based pixel-level inpainting approach for multiview video, originally proposed in [8].

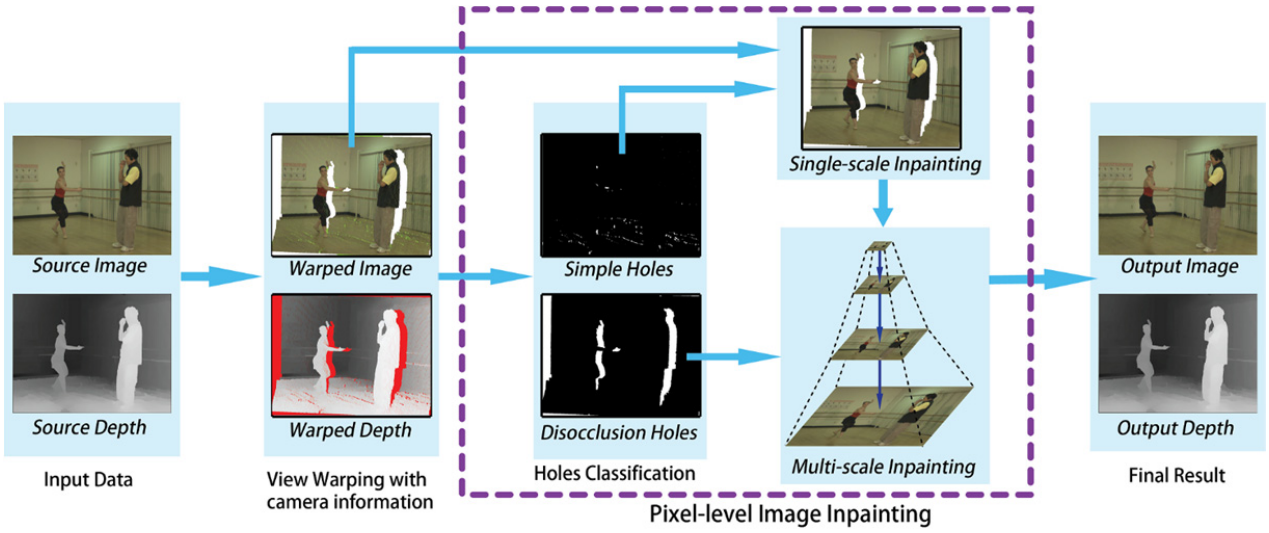


Fig. 1. Schema of our depth-based pixel-level inpainting method [8].

The considered technique operates in a multi-scale fashion and computes NNFs in order to identify matching candidates for each pixel to be inpainted. In this work, we propose an improvement on our earlier inpainting method [8] by considering additional spatial constraints as well as different search strategies in the approximate NNF computation.

The paper is structured as follows. In section 2, we give an overview of our inpainting method. Section 3 will explain the new additions to this framework and in section 4 we report the experimental results. The conclusions of our work are drawn in section 5.

2. INPAINTING METHOD

Fig. 1 depicts a schematic overview of our inpainting method of [8]. The first step comprises a classification stage where holes are declared to be either *simple holes* or a *disocclusion holes* simply by thresholding their surface. Simple holes are then directly inpainted, as explained next. A similar distinction is made in [13] where small holes coming from z-dimensional motion are interpolated while disocclusion holes are not addressed.

2.1. Pixel-level inpainting

To inpaint holes in an image, the inpainting algorithm needs to solve the problems of both structure propagation and preservation of local textures. Within the image I, we denote the holes and their boundaries as Ω and $\partial\Omega$, respectively. We will then search the known region $\Phi = I - \Omega$ for information to fill in the holes. The inpainting algorithm then iteratively fills in the hole boundary until the entire hole is inpainted. For any pixel $m \in \partial\Omega$ on the occlusion boundary and for a patch ψ_m centered around m, the optimal candidate is the pixel n that is the center of a patch ψ_n, identified as:

$$\psi_n = \underset{n \in \Phi}{\operatorname{argmin}}\, S(\psi_m, \psi_n) \qquad (1)$$

where $S(\cdot,\cdot)$ is a measure of similarity between two equally sized image patches. In our earlier work [8], this similarity measure was simply the Sum of Squared Differences (SSD) in *CIE*Lab* color space. In this work, we adopt the Sum of Absolute Differences (SAD):

$$S(\psi_m, \psi_n) = \frac{1}{\left|K_{\psi_m}\right|} \sum_{k \in K_{\psi_m}} \left|\psi_m(k) - \psi_n(k)\right|, \qquad (2)$$

where K_{ψ_m} is the set of pixel locations in ψ_m where the value is known and $\left|K_{\psi_m}\right|$ denotes the size of this set.

Note that our algorithm uses an exemplar-based approach to find plausible information to fill the holes, but performs the actual inpainting in a pixel-per-pixel fashion. This way, the algorithm will not suffer from artifacts coming from overlapping patches.

To find the optimal candidate patch to match the already known region around m, we employ a variation on the approximate NNF matching algorithm PatchMatch [7]. The algorithm is initialized with random candidates for each pixel and then proceeds iteratively using both candidate propagation and search to improve its current matches.

The candidate propagation step is built on the observation that if the best candidate for a pixel at location (u,v) is found, the best candidates for its neighbors $(u+1,v)$ and $(u,v+1)$ are likely to be also the neighbors of the best match for (u,v). Refining the random initialization only using this principle is prone to getting trapped in a local minimum. Therefore [7] also includes a random search step to avoid this effect.

2.2. Multi-scale disocclusion inpainting

If the simple holes are denoted by Ω_h, the remaining disocclusion holes are given by $\Omega_d = \Omega - \Omega_h$. To inpaint these, we construct an image pyramid:

$$P = \bigcup_k (\mathrm{I} - \Omega_d)\downarrow^k \tag{3}$$

where $k = \{0,1,\ldots,\mathrm{S}-1\}$ and $\downarrow^k$ denotes downsampling by a factor k, preceded by low-pass filtering to prevent aliasing. The pyramid consists of S resolution levels obtained by successive filtering and downsampling. Note that the simple holes in these images are already inpainted and only the disocclusion holes remain. We employ a multi-scale approach in order to avoid that our approximate NNF search gets trapped in a local minimum. In our earlier work [8] we showed that this approach indeed improves the visual performance compared to the single-scale version of our algorithm.

First the coarsest image P_{S-1} in the pyramid is inpainted using our pixel-level inpainting method, resulting in an image P'_{S-1}. If we upsample this image we obtain a prediction for the inpainted P_{S-2} image. We thus exploit this information in the true inpainting of P_{S-2}. This process is iteratively repeated until we reach the finest scale P_0 which is the output of our system.

3. COST FUNCTION

Note that equation (2) only considers the known pixels inside the patch. In the upsampled image there are no more unknown values. However, since this image is only a prediction of the actual one, we should control the degree of trust we invest in it; therefore, we adapt the SAD similarity measure in the following manner:

$$S^{(s)}(\psi_m,\psi_n) = \frac{1}{|\psi_m|}\left[\sum_{k\in K_{\psi_m}} |\psi_m(k)-\psi_n(k)| + \right.$$
$$\left. + \gamma^{(s)} \sum_{k\in U_{\psi_m}} |\psi_m(k)-\psi_n(k)|\right] \tag{4}$$

where K_{ψ_m} $\left(U_{\psi_m}\right)$ denotes the set of pixel locations where the value is known (unknown). $\gamma^{(s)}$ expresses the confidence we have in the upsampled inpainting result of the previous scale. Note that this makes the similarity measure scale dependent. When we inpaint the coarsest image in the pyramid and propagate this result to the next scale we cannot have too much confidence in these values. However, once we have refined this guess to P'_{S-2} we can have a little more confidence in the inpainted values. We therefore believe that the confidence values should increase as the algorithm progresses towards finer scales.

Furthermore, in multi-view inpainting there are some special constraints that can be built in the algorithm in order to guide the NNF search. For instance, compared to traditional image inpainting, multi-view inpainting is not limited to only use color information but can also consider scene depth information as input information, so we can further exploit the scene depth maps to refine the filling candidates. Furthermore, the disocclusion is normally caused by camera's horizontal shifting or because of foreground motion. The hole to be filled is more likely to belong to the background plane. Therefore, we introduce three additional factors, to which we refer as local depth coherency constraint, threshold constraint of reference distance and texture continuity factor. The cost function given by equation (4) is now extended to:

$$S^{(\mathrm{s})'}(\psi_m,\psi_n) = $$
$$= \; S^{(\mathrm{s})}(\psi_m,\psi_n)\cdot\delta_d^{(s)}(n)\cdot\delta_u^{(s)}(m,n)\cdot\delta_c^{(s)}(m) \tag{5}$$

In this equation, $\delta_d^{(s)}(n)$ is a local depth coherency indicator given by:

$$\delta_d^{(s)}(n) = \begin{cases} 1 & \text{if } d^s(n) < d^s_{\min} \\ e^{\frac{d^s(n)-d^s_{\min}}{d^s_{\max}-d^s_{\min}}} & \text{otherwise} \end{cases} \tag{6}$$

where $d^s_{\min}$ and $d^s_{\max}$ are the minimal and maximal depth values in the neighbor region of current hole.

$\delta_u^{(s)}(m,n)$ is a penalty to avoid choosing a candidate too far away, and the value is related to the Euclidean distance between the current position to be filled and the candidate pixel:

$$\delta_u^{(s)}(m,n) = \begin{cases} 1 & \text{if } l^s(n) < l^s_{\min} \\ e^{\left|l^s(m,n)-l^s_{\min}\right|} & \text{otherwise} \end{cases} \tag{7}$$

where $l^s(m,n)$ is the Euclidean distance between current point and the candidate, $l^s_{\min}$ is a fixed parameter and we set it as $l^s_{\min} = 30s$.

$\delta_c^{(s)}(m)$ is the texture continuity factor to produce continuous textures and we define it as:

$$\delta_c^{(s)}(m) = 1 + \left|\frac{V^s(m)-\overline{V}^s(m)}{\overline{V}^s(m)}\right| \tag{8}$$

where $V^s(m)$ is the offset between the current position m and its best candidate, $\overline{V}^s(m)$ is the mean offset of neighboring pixels which are inside the search window centered at position m. Note that such neighboring pixels have been inpainted in the current iteration and they are seen as known pixels for the current pixel.

4. SEARCH METHODS

In contrast to the original PatchMatch [7] algorithm, we perform deterministic rather than random searches to escape from local minima. PatchMatch iterates candidate propagation steps interleaved with random search steps. The

candidate propagation steps select a candidate patch for a pixel based on the candidates that were already found for its neighbors. The random search step then places a square window of size $d \times d$ around the center of the obtained candidate and randomly searches this window by iteratively choosing one random pixel followed by reducing the window size until it reaches the size of a single pixel.

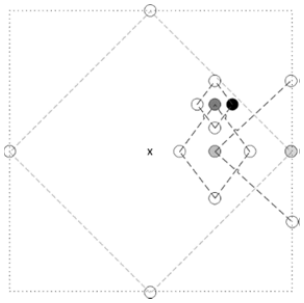
In our implementation we experiment with different search methodologies. More precisely, we employ logarithmic search, diamond search and full search in 2 dimensions. These search strategies are depicted in Fig. 2. Suppose that the candidate propagation step of PatchMatch yields a candidate patch centered in the pixel indicated by x in Fig. 2. This candidate is not necessarily the best match so it should be refined by considering other pixels in its vicinity. We center a search range with a certain radius d on x. Each pixel in this search range may be a better center for the best match, but considering them all is computationally expensive.

The circles represent the pixels that are evaluated by a particular method. In a diamond search of Fig. 2(a) only four pixels are evaluated in the first search step. The best of those four is selected and the neighbors of that pixel are evaluated. This is repeated iteratively. The logarithmic search evaluates all pixels that lie on the logarithmically spaced grid, as depicted in Fig. 2(b). We note that the original random search of PatchMatch is actually a special case of logarithmic search: for each "logarithmic radius" around x, illustrated by the dashed lines in Fig. 2(b), PatchMatch randomly selects a single pixel where the quality of the match is evaluated.

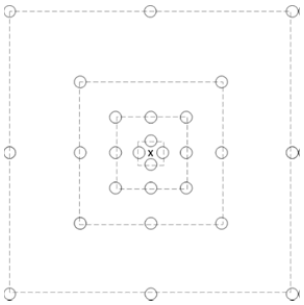
5. EXPERIMENTAL RESULTS

To validate the efficiency of different search strategies and the proposed matching function, we perform experiments on an MPEG reference MVD sequence, "Newspaper", in which the baseline between two adjacent cameras is 65 mm. In our experiments the input video is from camera 6 and the output is camera 4, which means the synthesized result is with twice the regular baseline.

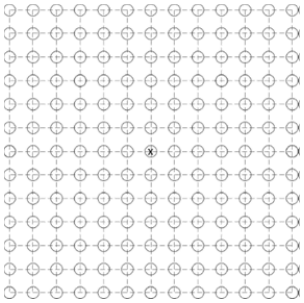
Firstly, experiments with different search strategies are performed in the proposed inpainting framework. As mentioned in Section 2, a candidate refinement search is introduced in order to escape from local minima after candidate propagation, and it is also one of the heaviest computing parts in the proposed algorithm. Here we employ Peak Signal-to-noise Ratio (PSNR), Structural Similarity (SSIM) [14] and Multi-scale Structural Similarity (MS-SSIM) [15] to evaluate the inpainting result. These objective metrics are conventionally being used in order to quantify view synthesis performance in the literature, e.g. [5]. As shown in Fig. 3, in general, for every search strategy the final synthesis result improves as search range increases. For a given search range, the 2D full search can produce superior results compared to the other techniques because it is more likely to find the best candidate.



(a) 2D diamond search



(b) 2D logarithmic search

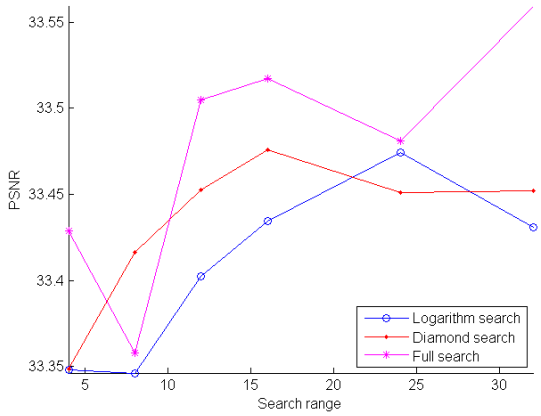


(d) 2D full search

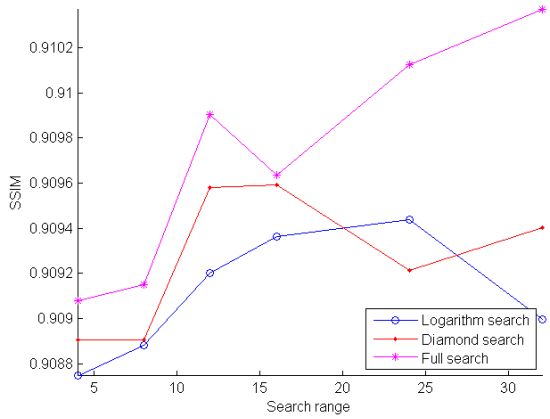
Fig. 2. Different search strategies.

Nevertheless, 2D full search is very slow because of the high number of pixels that need to be checked. 2D logarithmic search gets the lowest objective quality, one reason is that this search would be easily trapped into local minimal values. Compared to 2D logarithmic search, 2D diamond search can achieve good balance between the computation cost and final inpainting result.

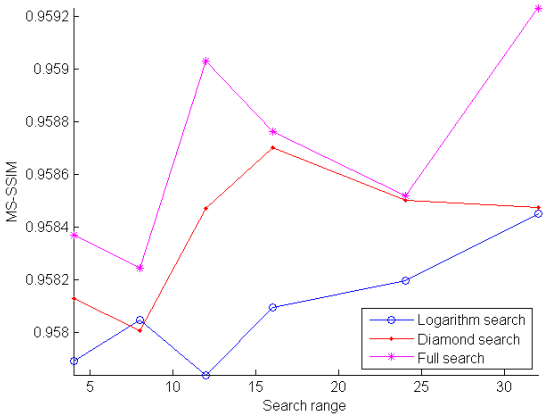
We also compare the proposed algorithm with our approach of [8] (Fig. 4 and 5). As shown in Fig. 4, the proposed algorithm can produce smoother results over time with higher and more stable PSNR values. Moreover, the MS-SSIM indicates that our proposed method is more effective to produce continuous structure textures. Regarding the subjective quality, the inpainting result in Fig. 5 shows that our algorithm can produce a better synthesis result than our previous method proposed in [8]. We note that our previous work [8] yields competitive results against reference techniques in the literature [4, 7, 9, 16].



(a) PSNR result

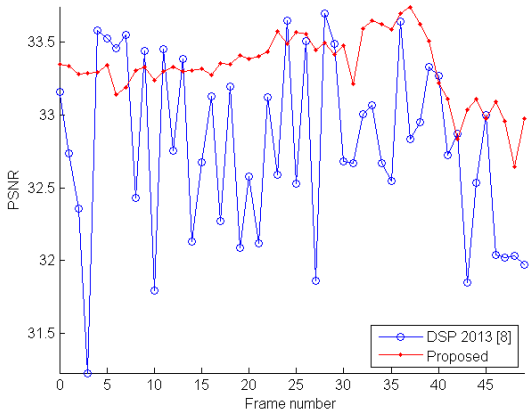


(b) SSIM result

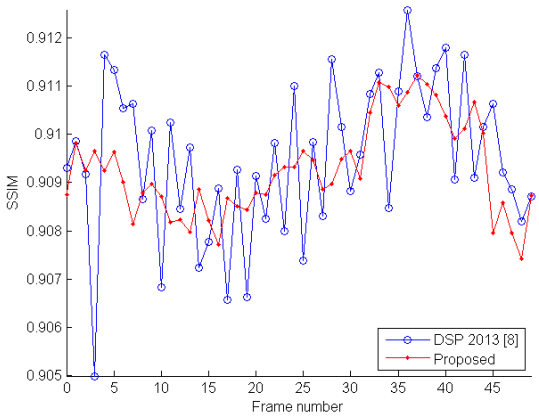


(c) MS-SSIM result

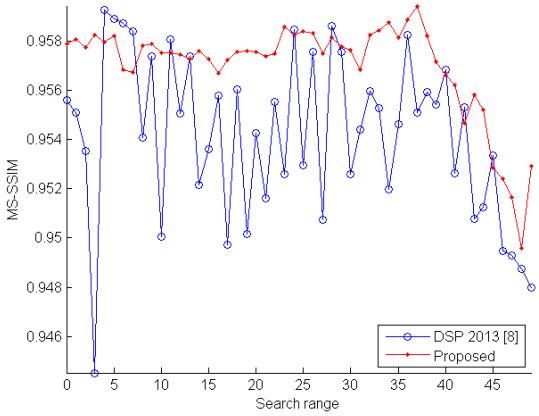
Fig. 3. Objective performance calculation by different search strategies and search ranges.



(a) PSNR result



(b) SSIM result



(c) MS-SSIM result

Fig. 4. Objective performance comparison of the proposed method and the reference technique of [8].

(a) Original occlusion regions.

(b) Inpainting result obtained with MPEG VSRS 3.5 [17].

(c) Inpainting result obtained with [8].

(d) The proposed algorithm.

Fig. 5. Inpainting results obtained with the proposed method and the reference techniques of [8] and [17].

6. CONCLUSIONS

We have presented a new approach for performing depth-based inpainting in multiview video. Our solution performs PatchMatch-based multilevel pixel-level inpainting employing candidates search and patch matching. An improved similarity measure for patch matching is introduced. Different search strategies and search ranges have also been investigated in the proposed framework. Experimental results reporting both subjective and objective evaluations show that the proposed inpainting algorithm can efficiently generate stable synthesis results.

ACKNOWLEDGEMENT

This work was supported by iMinds vzw and IWT in the context of the ASPRO+ project.

7. REFERENCES

[1] N. A. Dodgson, "Autostereoscopic 3D displays," *Computer*, vol. 38, no. 8, pp. 31-36, 2005.

[2] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," Proceedings of Electronic Imaging, pp. 93-104, 2004.

[3] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for novel view synthesis," Proceedings of IEEE International Workshop on Multimedia Signal Processing, MMSP, pp. 167-170, 2010.

[4] J. Gautier, O. Le Meur, and C. Guillemot, "Depth-based image completion for view synthesis," 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, pp. 1-4, 2011.

[5] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand, "Depth image-based rendering with advanced texture synthesis for

3-D video," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 453-465, 2011.

[6] V. Jantet, C. Guillemot, and L. Morin, "Joint projection filling method for occlusion handling in Depth-Image-Based Rendering," *3D Research*, vol. 2, no. 4, pp. 1-13, 2011.

[7] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 24, 2009, 2009.

[8] S. Lu, J. Hanca, A. Munteanu, and P. Schelkens, "Depth-based view synthesis using pixel-level image inpainting," IEEE International Conference on Digital Signal Processing, DSP, pp. 1-6, 2013.

[9] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200-1212, 2004.

[10] I. Ahn and C. Kim, "Depth-based disocclusion filling for virtual view synthesis," Proceedings of IEEE International Conference on Multimedia & Expo, ICME, pp. 109-114, 2012.

[11] S. Reel, G. Cheung, P. Wong, and L. S. Dooley, "Joint texture-depth pixel inpainting of disocclusion holes in virtual view synthesis," in *APSIPA ASC*. Kaohsiung, Taiwan, 2013.

[12] O. Le Meur, J. Gautier, and C. Guillemot, "Exemplar-based inpainting based on local geometry," *IEEE International Conference on Image Processing, ICIP*, 2011.

[13] Y. Mao, G. Cheung, A. Ortega, and Y. Ji, "Expansion Hole Filling in Depth-Image-Based Rendering using Graph-based Interpolation," *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, Vancouver, Canada, 2013.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

[15] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, vol. 2, pp. 1398-1402, 2003.

[16] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for novel view synthesis," IEEE International Workshop on Multimedia Signal Processing, MMSP, pp. 167-170, 2010.

[17] C. Lee and Y. S. Ho, "View Synthesis Reference Software (VSRS) 3.5," ISO/IEC JTC1/SC29/WG11, MPEG2008/M15851, 2008.