Depth image Compression Using Geometrical Wavelets

Majid Yaghouti Jafarabad, Vahid Kiani, Taha Hamedani, Ahad Harati

Robot Perception Laboratory, Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
ma.yaghouti@stu-mail.um.ac.ir, vahid.kiani@rocketmail.com, taha.hamedani@stu-mail.um.ac.ir, a.harati@um.ac.ir

Abstract— **Depth images are frequently used in robotic and 3D vision for different purposes like mapping or object recognition. Yet recently, they are encountered in many other areas such as free viewpoint and 3D television. Their innate redundancy especially in high frame rates and resolutions demands an effective compression algorithm or otherwise the required data rates grow prohibitively large.**

Standard lossy image and video compression methods, such as JPEG2000 and H264, remove high frequency and usually unimportant components of the signal in intensity images; which in the case of range or depth images accounts for edges and are essential for correct reconstruction of the scene geometry. Therefore, preserving geometrical properties of depth images should be the main objective in an effective compression algorithm. In this paper, Wedgelets, Platelets and Wedge-Platelets are proposed for depth image compression and are compared with JPEG2000 and H264. Moreover, for the first time, these methods are applied for compression of Kinect sensor depth images. Compared with previous works, it is shown that higher compression ratios up to 3dB can be achieved.

***Keywords-** Geometrical Wavelets, Wedgelet, Platelet, Wedge-Platelet, Depth Image Compression, Kinect, JPEG2000, H264.*

I. INTRODUCTION

Depth images are extensively used in robotic and 3D vision applications for different purposes like mapping or object recognition. With advent of Kinect sensor as a component of Microsoft X-Box 360, range and depth images are even more common place in such tasks and applications. Even such cheap sensor is capable of producing depth images at 30 frames per second. Therefore, accumulation of raw data soon exhausts any storage facility and compression is mandatory. On the other hand, a recent fast growth in 3DTV and free-viewpoint video fileds, introduces another instance of the depth image compression problem in a different setup. Hence, compression of depth images for transmission or storage is now an important and interesting research subject. In this paper, depth image is considered to be the z-map of the scene. In other words, each pixel in a depth image measures the z-coordinate of the corresponding 3D point in the scene or equivalently its distance to the xy-plane. It's worth mentioning that some texts use depth map to denote an image of distances to a single point, perhaps a camera, especially in robotic field. We use the word range image to denote this latter case and reserve depth image or depth map for the former case.

Main goal of traditional image compression techniques is preserving subjective quality; however, the aim of recently emerging depth image compression should be preserving geometrical characteristics of underlying scene. In other words, to obtain a reasonable and exact enough 3D reconstruction of the observed scene, one should preserve angle of corners and aspect ratio in addition of volume of the observed space. Polyhedral approximations seem a proper choice in many applications when indoor environments are considered.

In traditional image compression scope, wavelet transform is widely used because of its sparse and multi-resolution representation. By performing a wavelet transform, most of coefficients tend toward zero or a small enough number to be safely ignored. Hence, high compression rates for natural scenes may be achieved. However, classical wavelets are best suited for representing point singularities and not higher dimensional artifacts, i.e. geometries, encountered in multidimensional data. Edges in 2D images are among well-known examples of such geometrical artifacts. In natural scenes, edges usually have some regularity along their extension. But tensor product multidimensional generalization of 1D wavelet transform, i.e. separable multidimensional transform, has no way to benefit such regularity toward obtaining more sparse representation. Furthermore, indoor depth images exhibit distinct characteristics like smooth regions and simple texture-less structures in addition to sharp edges. Therefore, special purpose transforms which preserve these characteristics could be successfully used to compress depth maps. Wedgelets [2] and Platelets [7] are therefore a proper choice for depth image representation.

More than a decade ago, Donoho and Candes [1, 2] introduced an improved family of wavelets, now widely called geometrical wavelets, to better capture regularity of singular points in 2D images considering usually encountered dominant directionality. Geometrical wavelets have been extending rapidly in recent years [4,5,8-11,15] and come in two main groups: adaptive and non-adaptive.

Non-adaptive geometrical wavelets try to represent image data using a priori fixed set of basis functions which are usually obtained from classical wavelets by operations such as rotation or shear transformation. Ridgelets [1], Curvelets [3], Bandlets [4] and Contourlets [5] are among the most popular non-adaptive geometrical wavelets already used toward more sparse image representation.

In contrast, adaptive geometrical wavelets such as Wedgelets [2], Beamlets [6] and Platelets [7] use a customized dictionary of basis functions to represent each image. The dictionary is built by processing the image data which is to be represented and hence is adapted to the image at hand. In other words, adaptive geometrical wavelets exploit an optimal subset of basis functions to represent each image. A quad-tree is used to represent the given image using constructed atoms or words in the dictionary. The adaptation may result a better representation (smaller error), but the price to pay is higher computational complexity in discovering the quad-tree structure and building the dictionary.

Fundamental work on geometrical wavelets began by Donoho who introduced Wedgelets and Beamlets [2, 6]. Willett in [7] presented a generalization of wedgelets called platelets which is suitable for representing piecewise smooth functions and signals. Lisowska in [8] studied geometrical wavelets for intensity image compression and denoising. She extended wedgelets into smoothlets [9] suitable for representing images with smooth edges. Merkle in [10] used platelets for compression of depth images in multiple view video streams.

In this paper, we extend Merkle's work [10] by choosing basis functions more accurately, and applying these methods to indoor depth images of Kinect. Considering the fact that images of Kinect are usually quiet noisy, there are some negative points in the case of geometry coding; however, we show that our proposed methods can be successfully used for noisy depth image compression purposes. In addition, performance of different geometrical wavelets in compression of depth images has not been examined.

In the proposed methods a quad-tree based method is used to evaluate several geometrical wavelets in depth image compression. In next section, quad-tree based compression framework is described. Section 3 presents experimental results and discussions. In section 4, the paper is concluded and guidelines to future works are presented.

II. Compression Framework

In this section, different stages of compression framework are explained. First a complete quad-tree structure is generated for the whole image. Then four different modeling functions are used to estimate each block of image in each resolution. Thereafter, coefficients of modeling functions are quantized using a uniform quantization strategy. Finally the quad-tree is pruned and coded.

A. Image modeling

In this paper to estimate each block of image we use four modeling functions namely constant, wedge, linear and plate. A constant function estimates all pixels of the block with a single gray level and describes uniform parts of image very well. A wedge function divides each block into two parts and estimates each part with a single gray level. To do this, on the boundary of an N by N block, 4N-2 equidistant vertices could be considered on neighboring boundary pixels. By connecting every two vertices p1 and p2 a straight edge could be generated which divides the block into two disjoint parts called wedges. So every wedge and its complement can be described as:

$$w\left(x_1,x_2\right)=1_{\{x_2\le b(x_1)\}} \qquad \left(x_1,x_2\right)\in S \tag{1}$$

$$w'\left(x_1,x_2\right)=1_{\{x_2> b(x_1)\}} \qquad \left(x_1,x_2\right)\in S \tag{2}$$

where b presents an edge and S is a dyadic block of image. It should be mentioned that the edge $b_{p1,p2}$ and $b_{p2,p1}$ are two different edges. Hence, there are (4N-2)*(4N-3) possible edges in an N by N block.

Every wedgelet basis function consists of a wedge, its complement and two coefficients α and α' as below:

$$S = \alpha w + \alpha' w' \tag{3}$$

where w' is the complement of w in block S.

All possible wedges for blocks with sizes 2, 4, 8, 16, 32 and 62 are produced and for each block size a separate dictionary is constructed. Size of each dictionary is shown in table I. Basis functions of each dictionary will be used for estimating blocks of the same size.

Each linear basis function describes a dyadic block of image as below:

$$S(x,y) = Ax + By + C \tag{4}$$

TABLE I. Specification of Wedgelet Dictionaries

Block Size	*Dictionary Size*	*Bits*
2x2	7	3
4x4	75	7
8x8	499	9
16x16	2499	12
32x32	11107	14
64x64	46755	16

B. Quad-tree

Quad-tree is a tree data structure which is used in many computer graphic and machine vision algorithms. Root of quad-tree includes the whole image. Each node can divide into 4 smaller dyadic block and these partitioning can be continued until pixel level.

C. Estimation of Images

Every block of image is estimated by one of the basis functions (practically the most suitable one). In addition, coefficients of the appropriate basis function and its type are stored in the corresponding node of quad-tree. If the appropriate basis function is a constant function, only one coefficient must be stored. In fact, this coefficient which minimizes the MSE of such a block is mean of gray levels of that block. If the appropriate basis function is a wedge, two coefficients for that wedge and its complement must be stored. In order to find the best wedge, an exhaustive search is done among all wedges in dictionary to identify the wedge with minimum error. If the appropriate basis function is a linear function, 3 coefficients must be stored. Coefficients of the linear function are determined by least-squares minimization. Finally if the appropriate basis function is a plate, 6 coefficients must be stored in the corresponding node of quad-tree. In this case appropriate edge is found by minimizing estimation error on each wedge using same least-squares minimization process.

D. Rate-Distorion Optimization

Coefficients of basis functions are continuous so they must be quantized into discreet numbers during compression. For this purpose, all coefficients of quad-tree are quantized by a constant number of bits denoted by K. So there are 2^K quantization levels. It should be mentioned that we used uniform quantization.

Selection of basis function for each block is based on minimization of Rate-Distortion criterion. To do this, the very first stage is determining the bit usage for each basis function. For each constant basis function, only one gray level must be stored which needs K bits. To encode edge of each block we store index of that edge in the corresponding dictionary, instead of storing indices of start and end vertices of the edge as done by Merkle [11]. For each wedge function, 2 gray levels and index of wedge must be stored which consumes $2K+\log_2(|\text{Dictionary}|)$ bits. For each linear function, 3 coefficients must be stored which needs 3K bits. And finally for each plate, 6 coefficients and the index of used wedge must be stored which needs $6K+\log_2(|\text{Dictionary}|)$ bits. The best basis function for each node is determined using the cost function $C = D + \lambda R$.

where D denotes estimation error of type mean absolute error (MAE) and R denotes bit usage and λ is weighting parameter which balances the trade-off between error and bit usage. The basis function which minimizes the cost function C is selected as the best estimating function of block.

Rate-Distortion criterion is also used for pruning the tree and determining the best quad-tree which represents the image. Chou in [12] has proved that bottom-up tree pruning leads to an optimal quad-tree. According to this, by exploring the tree in bottom-up fashion cost of each node is compared with its four subtrees. If the cost of current node is less than its four subtrees, all four subtrees will be pruned. Otherwise, sum of cost of four subtrees is stored as cost of current node. There is also a bit to determine that current node is pruned or not. This bit is also considered in calculating cost of node. For each leaf node, P bits are needed to identify type of basis function. Value of P depends on variety of basis functions which are used.

E. Quantizer Selection

As mentioned in previous section, we quantize coefficients with constant number of bits K to optimize rate-distortion. The best number of quantization bits for a constant value of λ is determined as follow:

$$\tilde{k} = \underset{k=2,3,4,5,6,7,8}{\arg\min} \left(D(K) + \lambda R(K) \right) \tag{6}$$

where D(K) is error of estimated image using the tree with K bits for coefficients and R(K) is the bit usage of tree.

F. Compression of Quad-tree

In order to transform the quad-tree into a bitstream and compress it, we start from root of the tree. For each internal node, only one bit is written in file which shows that its subtrees has not been pruned. For each leaf node, at first one bit is written in file which shows that its subtrees has been pruned. Then, P bits are written to determine the type of basis function used for estimating the block. For constant function only a $\tilde{k}$ -bit coefficient, and for wedge function two $\tilde{k}$ -bit coefficients are written in the file. For linear function only a $\tilde{k}$ -bit coefficient (for constant C) is written and two coefficients A and B will be written at the end of file after arithmetic coding. For plate function, two $\tilde{k}$ -bit coefficients (constants C and C') are written and four coefficients A, A', B and B' will be written at the end of file after arithmetic coding. We use arithmetic coder because there are some blocks in depth images with similar linear or plate coefficients.

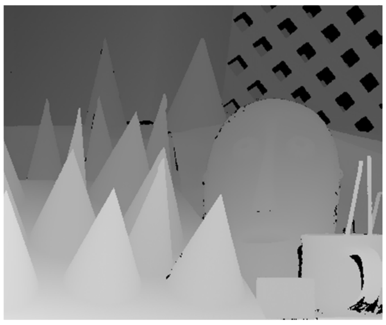
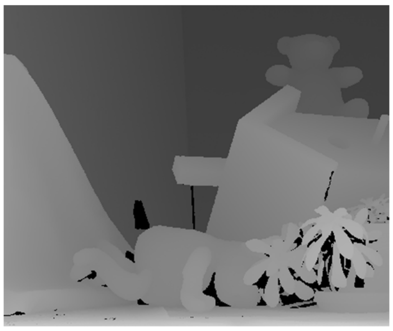

Figure 1. Test Depth images of Cones (Left), and Teddy (Right)

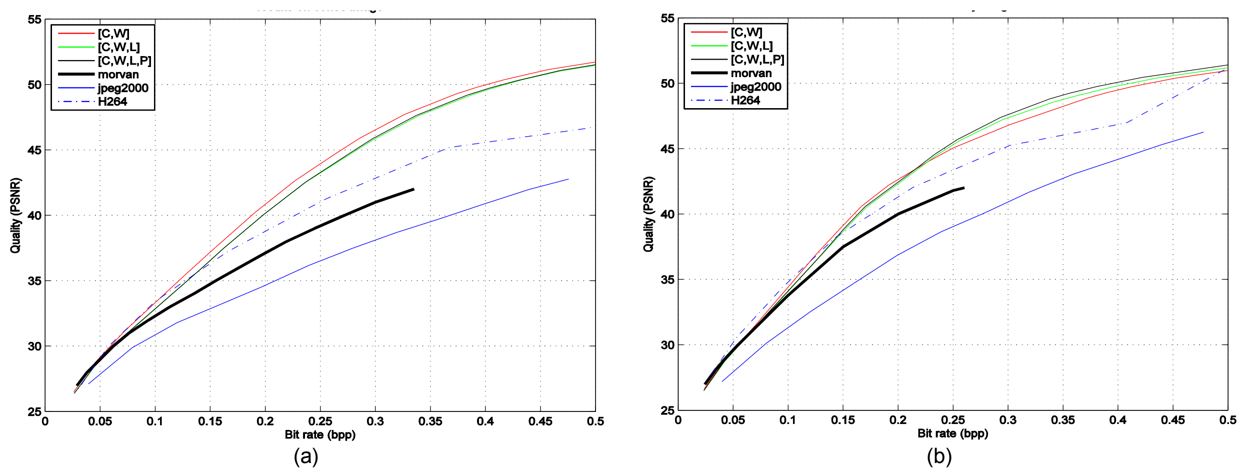

Figure 2. Quality of compression with different basis functions on (a) Cones image, and (b) Teddy image.

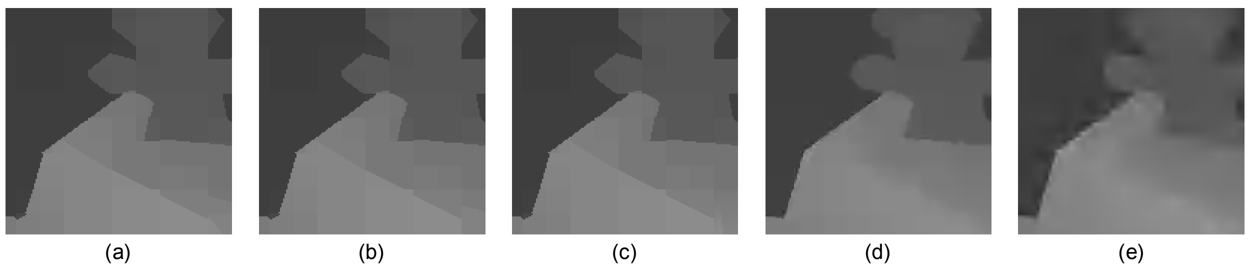

Figure 3. A part of compressed depth image of Teddy with bpp≈0.15 which compressed with (a) constant and wedge functions (b) constant, wedge and linear functions (c) constant, wedge, linear and plate functions (d) H264 (e) JPEG2000

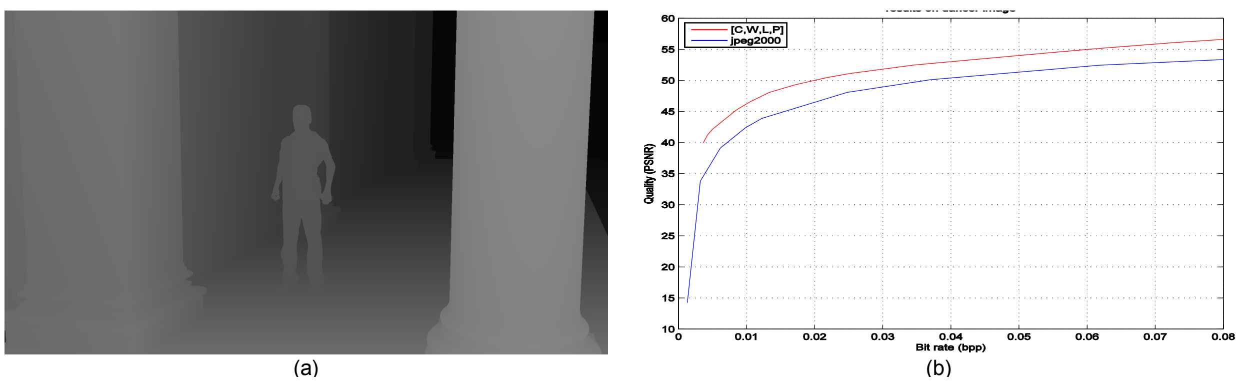

Figure 4. Comparison of geometrical wavelets with JPEG2000 on Dancer image (a) depth image (b) quality of compression

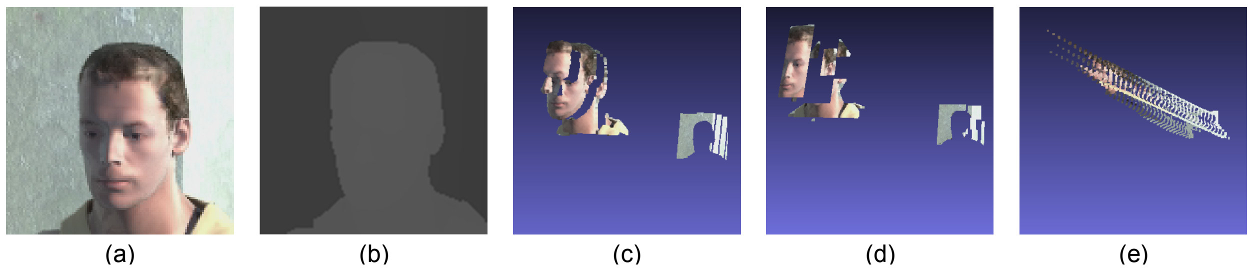

Figure 5. Performance of different compression schemes in preserving edges of depth image in 0.01 bpp (a) original color image (b) original depth image (c) original point-cloud (d) reconstructed point-cloud using geometrical wavelets (e) reconstructed point-cloud using JPEG2000

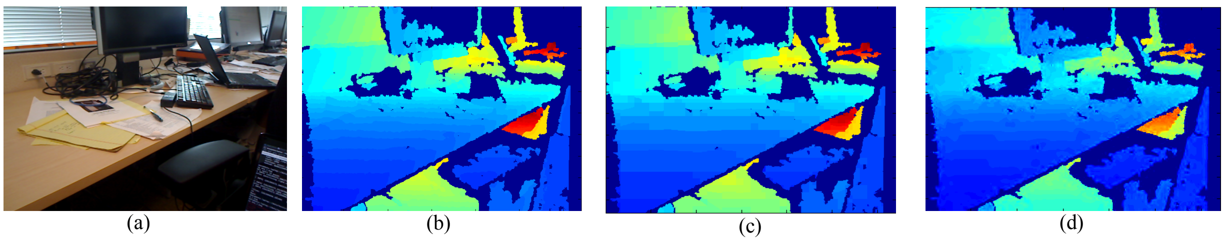

Figure 6. Compression of Desk image with bpp≈0.22 (a) original color image (b) original depth image (c) reconstructed depth image using geometrical wavelets with PSNR=37.45 dB, and (d) reconstructed depth image using JPEG2000 with PSNR=32.15 dB.

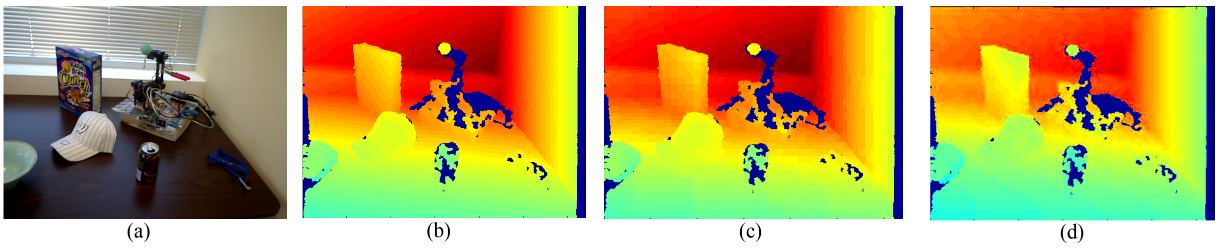

Figure 7. Compression of Table image with bpp≈0.2 (a) original color image (b) original depth image (c) reconstructed depth image using geometrical wavelets with PSNR=44.92 dB and (d) reconstructed depth image using JPEG2000 with PSNR=36.02 dB.

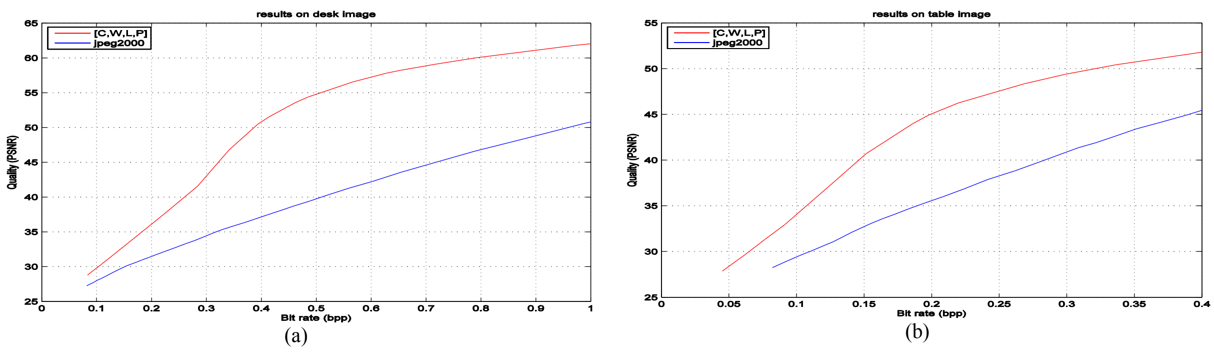

Figure 8. Quality of compression with geometrical wavelets and JPEG2000 on (a) Desk image, and (b) Table image.

III. EXPERIMENTAL RESULTS

In order to compare different methods of compression, we compressed depth images of Teddy and Cones, at first with constant and wedge functions, then with constant, wedge and linear functions and finally with all types of basis functions (i.e. constant, wedge, linear and plate). When we use constant and wedge functions to compress an image, P equals to log2(2) = 1. If three or four types of basis functions are used, P will be 2.

In addition to these geometrical wavelets, we compressed Teddy and Cones with intra-coded mode of JPEG2000 and H264 coding standards. We used Jasper and JM 14.2 libraries for JPEG2000 and H264 compression standards, respectively. Depth images of Teddy and Cones from Middlebury dataset [13] are shown in Fig. 1. Depth compression results in terms of rate-distortion performance are shown in Fig. 2. Compared to JPEG2000, geometrical wavelets achieved notably higher quality in the same bit-rates. In addition, they outperform H264 coding in most bit-rates. However, depending on the geometry of captured scene, performance of different geometrical basis functions is comparable to each other; e.g. on the cones image performance of wedgelets is higher, while on teddy image composition of all basis functions performs better. A similar image compression framework proposed by Morvan in [11]. Compared to results of Morvan, we achieved up to 5 dB higher quality on cones image, and up to 3 dB higher performance on teddy image.

For the sake of visual comparison, a part of original and compressed depth maps is shown in Fig. 3. Geometrical wavelets successfully preserved sharp edges around teddy bear and house, while H264 blurred them and JPEG2000 strongly corrupted them.

In another experiment, we compared platelet based coding scheme with JPEG2000 on the first frame of

Dancer sequence with size of 1920x1088 pixels [14]. This scene has simple structures like wall, floor, and column. As shown in Fig. 4 geometrical wavelets have higher performance than JPEG2000 in all bit-rates. We reconstructed point cloud of the scene using compressed and original depth maps. Fig. 5 shows parts of sample reconstructed point clouds in bit-rate of 0.01 bpp. As shown in Fig. 5 geometrical wavelets preserve sharp edges around the dancer's head; while JPEG2000 corrupts these edges and smooth them.

We also used geometrical wavelets for compressing Kinect depth images which have more complicated scenes with considerable noise. Depth images of Desk and Table from Washington dataset [15] are shown in Fig. 6 (a) and Fig. 7 (a). As shown in Fig. 6 (c) and (d), geometrical wavelets preserved sharp edges around monitor and window while JPEG2000 blurred and corrupted them. Similarly, the same happened in Fig. 7 (c) and (d) around box, cap and the edge between wall and window. Depth compression results for Desk and Table images in terms of rate-distortion performance are shown in Fig. 8. In the presence of noise, geometrical wavelets achieved notably higher quality than JPEG2000 in the same bit-rates.

IV. Conclusion

In this paper we evaluated performance of wedgelets, platelets and mix of them in depth image compression. Results of the proposed methods on Teddy, Cones, Dancer, Desk and Table depth images were compared with JPEG2000 and H264 coding standards.

It was shown that narrow field of view indoor depth images can be sparsely represented using geometrical wavelets due to their simple geometry and piecewise smoothness. Our results suggested that comparing with JPEG2000 and H264, the proposed methods better preserve edges and geometry of the scene which generally leads to 3dB gain in PSNR.

However, a big disadvantage of the proposed methods is long run time. While JPEG2000 and H264 use only a few seconds for compressing a 512x512 pixels image, wedgelets and platelets need several minutes. Our future work focuses on reducing run time.

References

[1] E.J. Candes, Ridgelets: Theory and Applications, Ph.D Thesis, Dept. Statistics, Stanford Univ., CA, 1998.

[2] D.L. Donoho, Wedgelets: Nearly-Minimax Estimation of Edges, Ann. Statist., vol. 27, pp. 859-897, 1999.

[3] E.J. Candes, and D.L. Donoho, Curvelets – A Surprisingly Effective Nonadaptive Representation For Objects with Edges, in Curves and Surface Fitting. Nashville, TN: Vanderbilt Univ. Press, 1999, pp. 105-120.

[4] C. Dossal, E.L. Pennec, and S. Mallat, Bandlet image estimation with model selection, Signal Processing, Vol. 91(12): p. 2743-2753, 2011.

[5] M.N. Do, and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, IEEE Transactions on Image Processing, Vol. 14(12): p. 2091-2106, 2005.

[6] D.L Donoho, and X. Huo, Beamlet Pyramids: A New Form of Multiresolution Analysis, suited for Extracting Lines, Curves, and Objects from Very Noisy Image Data, Proc. SPIE, vol. 4119, pp. 434-444, 2000.

[7] R.M. Willet, and R.D. Nowak, Platelets: A Multiscale Approach for Recovering Edges and Surfaces in Photon-Limited Medical Imaging, IEEE Transactions on Medical Imaging, vol. 22, no. 3, pp. 332-350, Mar. 2003.

[8] A. Lisowska, Geometrical Wavelets and their Generalizations in Digital Image Coding and Processing, Ph.D. Thesis, Faculty of Computer Science and Materials Science, Univ. Silesia, Poland, 2005.

[9] A. Lisowska, Smoothlets—Multiscale Functions for Adaptive Representation of Images, IEEE Transactions on Image Processing, vol. 20, no. 7, 2011.

[10] P. Merkle, et al., The effects of multiview depth video compression on multiview rendering, Signal Processing: Image Communication, vol. 24(1–2), pp. 73-88, 2009.

[11] Y. Morvan, Acquisition, compression and rendering of depth and texture for multi-view video, Eindhoven University of Technology, Eindhoven, 2009.

[12] P.A. Chou, T.D. Lookabaugh, and R.M. Gray, Optimal pruning with applications to tree-structured source coding and modeling. IEEE Transaction on Information Theory, vol. 35, no. 2, pp. 299 – 315, 1989.

[13] http://vision.middlebury.edu/stereo/data/scenes2003/.

[14] http://mpeg3dv.research.nokia.com/sequence.html.

[15] L. Kevin, B. Liefeng, R. Xiaofeng, and F. Dieter, A Large-Scale Hierarchical Multi-View RGB-D Object Dataset, In IEEE International Conference on Robotics and Automation (ICRA), May 2011, available at: http://rgbd-dataset.cs.washington.edu/dataset/rgbd-scenes/.