Temporal RVL: A Depth Stream Compression Method

Hanseul Jun*
Stanford University

Jeremy Bailenson†
Stanford University.

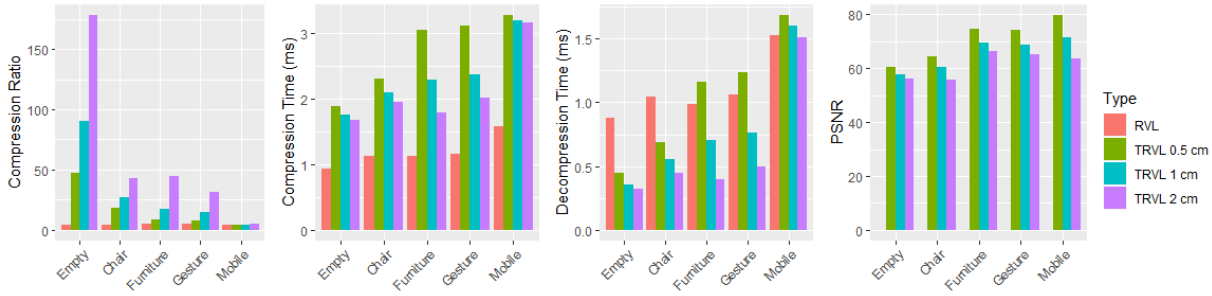

Figure 1: The compression ratio, compression time, decompression time, and peak signal to noise ratio (PSNR) per depth stream of RVL and TRVL with change tolerance thresholds of 0.5 cm, 1 cm, and 2 cm. As RVL is lossless, its PSNR is not provided.

Abstract

The advent of depth cameras has led to new opportunities, but at the same time has led to new challenges in the form of larger network bandwidth. To address this problem, we propose a lossy compression method Temporal RVL, which results in better compression with little loss of depth information. Temporal RVL adds a preprocessing step to RVL and effectively utilizes the similarities across frames, while maintaining important depth information such as edges. For the default settings, Temporal RVL achieves a compression ratio of 20.1 (4.2 times higher than RVL) while at the same time facilitating faster decompression.

Index Terms: Depth stream compression; depth camera

1 Introduction

With the advent of depth cameras as devices, depth streams—2D depth information—made their debut. For depth stream compression, as any other form of data, higher compression ratio and lower computational complexity are major goals, especially given obvious AR/VR use cases of depth cameras: using multiple cameras to build holograms [3] or decompressing and rendering depth streams in mobile devices [2]. Usage of multiple cameras multiplies the burden to compress and decompress depth streams and mobile devices have limited computational resources that demand computational efficiency. Unfortunately, standard video compression techniques (e.g., VP8 [1], H264 [4]) are unsuitable for depth streams as they do not properly preserve the edges (i.e., connectivity between objects) and leaves unsuitable artifacts (see Figure 2 for the artifacts).

Wilson [5] introduced RVL, a depth map compression method that is well-aligned to these conditions, which is lossless and supports fast compression and decompression. In this paper, we introduce a depth stream compression technique that extends RVL to utilize temporal redundancy and achieves better compression ratio without a significant increase in computational complexity.

*e-mail: hanseul@stanford.edu
†e-mail: bailenso@stanford.edu

Figure 2: Visual comparison of RVL (top-right), TRVL (bottom-left), and VP8 (bottom-right). These are point clouds rendered by the same color and depth stream but with different compression methods. The top-left is from the original color stream.

2 Temporal RVL

Temporal RVL (TRVL) first preprocesses the frames of depth streams, then uses RVL to compress the difference between the frame and the previous frame (i.e., delta encoding). In its preprocessing step, TRVL adds change and invalidity tolerance to depth streams based on observations from depth cameras with fixed positions.

2.1 Change Tolerance

Consider the depth value of a pixel which is corresponding to a person's hand changes by 5 cm from one frame to the next. There are two possibilities here. First, the hand could have moved very quickly backwards (i.e., positive on the direction of depth). Second, the hand could have moved to the side, and the ray corresponding to the pixel is now hitting the chest of the person which was 5 cm behind the hand (i.e., the ray detected the edge of the hand). We will call the change of depth corresponding to the moving hand *intra-object*, and the change of depth connoting a move to a new object *inter-object*.

Intra-object changes do not contain crucial geometric changes such as edges. From this perspective, if information loss should

Figure 3: The environments where the depth streams for the evaluation were taken: a room (left) and an indoor space (right).

happen, it should be intra-object changes, not inter-object changes. One characteristic of these inter-object changes that we can utilize for preservative purposes is their relatively large scale. Leveraging this, a small threshold (e.g., 1 cm) can be chosen as the upper bound for intra-object changes, then changes can be ignored until they diverge more than the threshold.

2.2 Invalidity Tolerance

Depth cameras cannot guarantee the measurement of every pixel, therefore leaving invalid pixels. While the invalidity of these pixels reflects the uncertainty of the corresponding measurements, this does not necessarily mean there was no object to measure. Especially, in case the pixel value was measured in the previous frame, the pixel value of the previous frame may still contain a decent amount of certainty as a measurement. Leveraging this, we suggest ignoring nonconsecutive invalid pixels in the preprocessing step.

Adding change and invalidity tolerance, the preprocessing step temporally stabilizes pixel values by changing a pixel value only when 1) the difference between the fixed value and a measurement is larger than a threshold or 2) when two consecutive measurements failed. The use of the difference between the fixed value and the measurement, instead of the difference between the previous measurement and the current measurement, limits the error from the preprocessing step to the given threshold. Since RVL is designed to efficiently compress running zeros, the temporal redundancy added by the preprocessing step boosts the efficiency of this delta encoding. For decompression, the compressed difference between frames gets RVL decompressed then added to the previous frame. We provide C++ code for this technique at https://github.com/hanseuljun/temporal-rvl.

3 Result and Discussion

We compared TRVL to RVL with five depth streams. The depth streams were taken using a depth camera (i.e., Kinect for Azure) with 640x576 resolution and 30 Hz update rate. The depth streams were taken from two different environments: a room with all objects in range of the depth camera and an indoor space with objects outside the range (see Figure 3). We used a Windows 10 PC with an Intel Core i7-8650U CPU @ 1.9GHz for the comparison. The names of the depth streams are:

Empty Depth stream from a room without anyone.

Chair Depth stream from a room with a person sitting on a chair and talking towards the depth camera.

Furniture Depth stream from an indoor space without anyone.

Gesture Depth stream from an indoor space with a standing person talking with gestures.

Mobile Depth stream from an indoor space while the depth camera itself is moving.

Figure 1 describes compression ratio, compression time, decompression time, and peak signal to noise ratio (PSNR) of RVL and TRVL with different change tolerance thresholds: 0.5 cm, 1 cm, and 2 cm. For compression and decompression time, we used the average time for compression and decompression per frame. Compared to RVL, TRVL resulted in higher compression ratio, longer compression time, and shorter decompression time. Between different tolerance thresholds, larger thresholds that added more temporal redundancy to depth streams resulted in higher compression ratio, faster compression and decompression, and lower PSNR.

In terms of the individual depth streams, for **Empty**, the compression ratio of TRVL was 90.6 being 19.3 times better than RVL. On the other hand, for **Mobile**, there was no significant gain of compression ratio from TRVL compared to RVL, though the compression ratio was still not worse than RVL.

In short, excluding extreme cases (i.e., **Empty** and **Mobile**) for a fair comparison, given 1 cm as the change tolerance threshold, TRVL showed 4.2 times higher compression ratio, 97% longer compression time, and 35% shorter decompression time compared to RVL.

To further examine the quality of compression methods, we rendered points clouds [2] with the compression methods using the same color and depth stream for each method. Figure 2 includes screenshots of RVL, TRVL with 1 cm change tolerance threshold, and VP8. RVL and TRVL resulted in virtually the same scene except for TRVL having less jittering due to its change and invalidity tolerance. In the meanwhile, the scene rendered with VP8 was not able to appropriately display the recorded stream. The video containing the full version of this comparison can be found as supplementary material of this paper.

4 Conclusion

We introduced a depth stream compression method that has a higher compression ratio and shorter decompression time than RVL, which is arguably the state-of-art depth stream compression method. With the compression ratio of 20.1 that was obtained with TRVL with 1 cm as the change tolerance threshold, a Kinect for Azure depth stream which originally requires 168.75 Mbps would require 8.4 Mbps that is in a much more satisfying range.

Acknowledgments

This research was supported by two National Science Foundation grants (IIS-1800922 and CMMI-1840131). The authors would also like to thank Kedar Tatwawadi for feedback on the paper.

References

[1] J. Bankoski, P. Wilkins, and Y. Xu. Technical overview of vp8, an open source video codec for the web. In *2011 IEEE International Conference on Multimedia and Expo*, pp. 1–6. IEEE, 2011.

[2] H. Jun, J. N. Bailenson, H. Fuchs, and G. Wetzstein. An easy-to-use pipeline for an rgbd camera and an ar headset. *PRESENCE: Teleoperators and Virtual Environments*, in press.

[3] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 741–754. ACM, 2016.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[5] A. D. Wilson. Fast lossless depth image compression. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pp. 100–105. ACM, 2017.