2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Effective Compression of Range Data Streams for Remote Robot Operations using H.264

Fabrizio Nenci Luciano Spinello Cyrill Stachniss

Abstract— **Most robots need the ability to communicate with a base station or with an operator during their mission. Teleoperated and semi-autonomous robots typically communicate continuously through a network connection with an operator. Transmitting raw sensor data over a low bandwidth network such as wireless or HSDPA, however, is problematic as the stream of sensor data is often large. In this paper, we present a method that exploits H.264 compression to reduce the size of range data streams from sensors such as the Kinect camera or the Velodyne 3D laser scanner. We developed a practical and effective solution that exploits the state of the art in video compression to produce high-quality results. Our method is easy to implement and can have practical impact for researchers building robots for the real world. We implemented and thoroughly tested our approach using a large number of range data streams. Furthermore, we analyzed the impact of data compression on the accuracy and size of the transmitted data. We show that even a highly compressed stream of depth images can be used with dense mapping techniques such as KinFu for building environment models.**

I. Introduction

A key application for mobile robots is their use in hazardous or difficult to access environments This includes search and rescue missions, exploration of unknown environments, digitizing archeological sites, or inspections of contaminated areas. Depending on the application scenario and the availability of humans in the loop, such robots are either remote controlled, semi-autonomous, or fully autonomous. For tele-operated and semi-autonomous operation, it is essential that the robot can communicate with a human operator and can transmit at least parts of the acquired sensor data. Users of such systems typically need to see what the platform is currently doing and what it is sensing, i.e., a local environment model or a subset of the acquired camera and depth images. To achieve that, the data must be transferred over a relatively slow network link such as WiFi or HSDPA. Besides the aspect of user intervention, transmitting sensor data to a remote base station for performing computations off-board can also be advantageous for fully autonomous robots and resource-constrained systems such as UAVs or humanoids. A central problem in real-world environments is that the network connection is not fast enough to transfer the full stream of sensor data to a remote computer. For example, the popular Kinect camera generates a data stream of RBG and depth images of around 45 MB/s, which exceeds the bandwidth limits of most networks.

Fig. 1. Remotely operated robot in an underground catacomb. The robot streams Kinect depth data for building a map to a remote operator.

In this paper, we address the problem of effectively transmitting range data from a typical depth sensor such as a Kinect camera or a Velodyne 3D laser scanner through a slow network connection. Our solution builds on top of modern video codecs such as H.264 to transmit a compressed version of the original data stream that requires only a fraction of the bandwidth. Exploiting existing video compressors has several advantages: it exploits years of research on video compression, it is highly effective as it leverages spatiotemporal relations in the data, it is easy to implement given existing libraries, and H.264 is supported by a large variety of devices. Our solution is able to deal with data streams in which the individual sample is described by more than 8 bit and proposes a range-based signal demultiplexing to reduce the effects of compression artifacts.

We present an extensive experimental evaluation in which we show that our solution is effective for transmitting depth streams. Our evaluation suggests that the approximation error, which is introduced by the lossy compression, is acceptable in the sense that dense mapping systems [13], [11] can build environment models from this data. This is useful for relaying depth data from semi- or fully autonomous exploration robots as the one shown in Figure 1. Here, our robot explores an inaccessible underground catacomb while relying on Kinect range data that is streamed to a remote machine. The software is released as open source and it is available at `http://www.ipb.uni-bonn.de/software`.

This work has been supported by the EC under contract number FP7-ICT-600890-ROVINA.

All authors are with the University of Freiburg, Dept. for Computer Science, 79110 Freiburg, Germany. Fabrizio Nenci and Cyrill Stachniss are also with the University of Bonn, Inst. of Geodesy and Geoinformation, 53115 Bonn, Germany.

978-1-4799-6934-0/14/$31.00 ©2014 IEEE

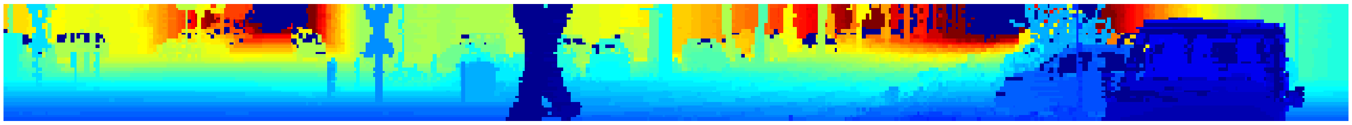

Fig. 2. A depth image built from a single rotation of a Velodyne scan

II. Related work

The problem of compressing data for network transmission has a long history. In the context of robotics, we mainly find two different classes of approaches: first, techniques that transmit models such as a 3D environment map over the network and second, methods that transmit the sensor data itself. In this paper, we address the second class of problems, i.e., transmitting range data from a range sensor over a network connection. A straightforward solution is to transform the range data into a depth image and transmit it using video codecs. Already in 2001, Krishnamurthy et al. [6] pointed out that a straightforward application of image compression algorithms to depth images does not lead to good results.

Chai et al. [1] address the topic of depth image-based rendering for 3DTV and propose a mesh-based 3D representation of the scene. They extend an algorithm previously used by Lindstrom et al. [8] for real-time height fields rendering. Schnabel and Klein [14] as well as Hornung et al. [3] explore the compression of point clouds based on octrees. These approaches build local models and are especially effective for large free-space areas. Related to that, Morvan et al. [10] use a quadtree-based decomposition of single depth images to compress depth data. Merry et al. [9] propose a solution for dense and regular point clouds that is based on a compression using a spanning tree representation. Kammerl et al. [4] extend the method of Schnabel and Klein [14] to compress the octrees using an algorithm that expresses the differences of the current point could with respect to the previous one.

In 2006, Oh and Ho [12] investigated means to achieve faster encoding with respect to H.264 when encoding depth images by estimating the motion vectors from the RGB images associated to the depth frame. This approach requires RGB and depth data and cannot be used for range data from 3D laser scanners. To improve the compression results of H.264 on depth data, Lai et al. [7] propose to use de-artifacting filters in combination with H.264 encoding. They show that this approach avoids artifacts and leads to an improved peak-signal-to-noise ratio. In 2013, Karpinsky and Zhang [5] propose a close-to-realtime approach for 3D geometry video compression exploiting the holoimage technique [2] for 3D models and streaming the reduced 2D image through H.264.

In this paper, we address the problem of effectively streaming range data over low bandwidth networks. In contrast to several other approaches, our method does not rely on local models, maps, or voxels and operates on the sensor data of typical range sensors used in robotics, i.e., depth cameras as well as 3D laser scanners.

III. Compressing range data streams using H.264

In this paper, we operate on the range data as it is generated by a depth camera, a 3D laser scanner, or other similar range finder devices. We encode each scan as a depth image, see Figure 2 for an example. This allows us to leverage existing video codecs to obtain a high quality compressed depth image stream. Video codecs such as H.264 are only effective if the individual images are part of a continuous image stream, i.e., there is a spatiotemporal relation between the images and the stream is not a random collection of images. This, however, is the case for most data streams that are obtained in the context of robotics.

A. Using H.264 on range data

Our approach makes use of a H.264 video encoding technique. This choice has the following advantages:

- The H.264 compression level is adjustable and allows for finding a good trade-off between bandwidth limitations and desired accuracy.
- The codec exploits the spatiotemporal relations in the data, which are often found in robotic applications including mapping, exploration, as well as search and rescue missions.
- H.264 is a wide-spread technology. Nowadays even mobile phones and inexpensive hardware solutions can natively compress high-resolution streams using this codec.

For each image, the standard H.264 video compression algorithm consists of three main steps. First, it tessellates the image in blocks. Second, it computes for each block a Discrete Cosine Transformation. Third, it applies a low-pass filter to the transformed block. The higher the compression level of the codec, the lower the cut-off frequency. An effect of the low-pass filter is that sharp edges are often strongly corrupted. After performing these three steps, the continuous nature of the data stream is exploited by storing differences with respect to other (key)frames.

The H.264 codec is tailored to compress RGB videos so that the resulting video artifacts are not perceivable by the human eye. As a conseguence, this does not necessarily minimize the residual error between the compressed image and the original one. For example, this codec generates small compression artifacts in smooth and similarly colored areas and strong artifacts in areas with moving edges. These artifacts result in errors in the compressed data stream. For depth data, this means that smooth surfaces show comparably few errors in the reconstructed depth images, whereas edges are often strongly affected by the compression artifacts. As a result of that, the value of a pixel in the compressed depth image can be substantially different from the original one.

A second problem of RGB-native video codes for range data compression is the number of bits per pixel that the codec supports. There exist high-definition profiles in the H.264 standard that support more than 8 bit per pixel. Most implementations, however, support only 8 bit per pixel and this value cannot be changed easily. Depth data from a typical range sensor, however, requires more than 8 bit to encode the distance information, e.g., 11 bit for the Kinect depth image and 16 bit for a Velodyne laser range scanner.

As a result of these two issues, the straightforward application of the H.264 codec for depth image compression leads to data streams with substantial errors in the range data. Therefore, we propose a solution that makes use of range-demultiplexing of the original data stream before compressing it with H.264. We subdivide the stream in multiple smaller ones, each describing a selected range of depth data. This leads to substantially reduced artifacts and, at the same time, keeps the size of the stream small.

B. Demultiplexing and compressing range data streams

Our approach starts with the range data from a range sensor. Each range scan is transformed to a depth image (some sensors such as the Kinect provide the depth image directly). We demux the depth image in multiple channels, compress each channel individually using H.264, and transmit the created H.264 streams. At the receiver's end the streams are decompressed and recomposed into a single depth image stream.

Our range-demuxing strategy works by assigning a selected range of depth data to each demuxed channel. As a result, each channel contains only range data within a certain depth interval. Let R be the maximum range of the sensor, J the number of range intervals, and B the number of bits that H.264 supports per channel, i.e., typically $B = 8$.

Without loss of generality, we consider a uniform depth subdivision. The metric resolution per unit r_c of each channel c is computed as

$$r_c \quad = \quad \frac{R}{J \cdot 2^B}. \tag{1}$$

Thus, r_c represents the discretization of the range data. Let $\mathbf{D}$ be the original depth image. For the j-th channel c_j, we select all range measurements in the interval

$$\begin{aligned} \mathcal{H}_j &= [D \cdot j, D \cdot (j+1)) \\ &= \{x \in \mathbf{D} \,|\, D \cdot j \leq x < D \cdot (j+1)\}. \end{aligned} \tag{2}$$

This leads to the channel image $\mathbf{D}_j$, which is defined as

$$\mathbf{D}_j \quad = \quad \frac{f(\mathbf{D}, \mathcal{H}_j) - D \cdot j}{D} \cdot 2^B, \tag{3}$$

where $f(\cdot)$ returns an image with pixels having depth in the interval $\mathcal{H}_j$ and with all other pixels equal to zero. Note that all elements of $\mathbf{D}_j$ take values between 0 and 2^B. In the remainder of this paper, we call the individual depth images $\mathbf{D}_j$ of a demuxed channel a *"slice"*. For further processing, we assume the $\mathbf{D}_j$ to be 8 bit images that are compressed with the H.264 codec.

Depending on the properties of the sensor and on the requirements of the application, the range intervals of the individual channels can be designed to have non-uniform ranges. This allows for using a high resolution to capture specific depth intervals and coarser ones for areas where high accuracy is not required or not available.

In addition to the set of slices $\mathbf{D}_j$ with $j \in \{1, \ldots, J\}$, we compute an index matrix $\mathbf{K}$ with elements in $\{0, \ldots, J\}$ indicating for each pixel, which slice is the one that contains the valid depth information. An index of zero is used to indicate a missing measurement. The use of $\mathbf{K}$ is important as the decoder needs to know which slices $\mathbf{D}_j$ to use for reconstructing the range measurements on a per pixel basis. Thus, the matrix $\mathbf{K}$ must be transmitted with lossless compression within H.264. Although this may sound like a large overhead, the exploitation of the temporal dependency between consecutive index matrices $\mathbf{K}$ leads to high compression rates.

Note that the overall number of bits used to model a range measurement in the input is typically smaller than the one in the raw demuxed depth image. Nevertheless, the H.264 codec typically compresses the demuxed stream by exploiting the spatiotemporal correlation of the data. As a result, the encoded demuxed stream is substantially smaller than the input.

In sum, the advantage of the demuxing approach is twofold. First, it allows us to encode data streams using an arbitrary number of bits per measurement. Second, it limits the impact of compression artifacts in the compressed data stream thanks to the normalization in Eq. (3).

IV. Experiments

Our experiments are designed to show that the proposed approach is an effective mean for compressing range data streams. First, we evaluate the accuracy under lossy compression. Second, we evaluate the bandwidth requirements at different compression levels. Finally, we evaluate the impact that the compression has on mapping algorithms such as KinFu [13].

A. Setup

For our evaluation, we use 15 real world robotic datasets. Except for one dataset, all are publicly available through [16] and [15]. The non-publicly available dataset consists of Kinect depth data that we recorded in a catacomb in Rome during an archeological digitization experiment with our robot shown in Figure 1. Overall, the datasets include far-range data as well as close range recordings, moving sensors, static sensors and moving people.

For the H.264 implementation, we rely on the x264 codec by VideoLAN [17]. For varying the compression level of the video codec, we change the quality profile parameter *qp*, which takes integer values between 0 (lossless compression) and 69 (maximum compression).

The key parameter of our range-demuxing approach is J, the number of slices. In our experiments, we mainly use two configurations with $J = 6$ and $J = 24$ slices to which we

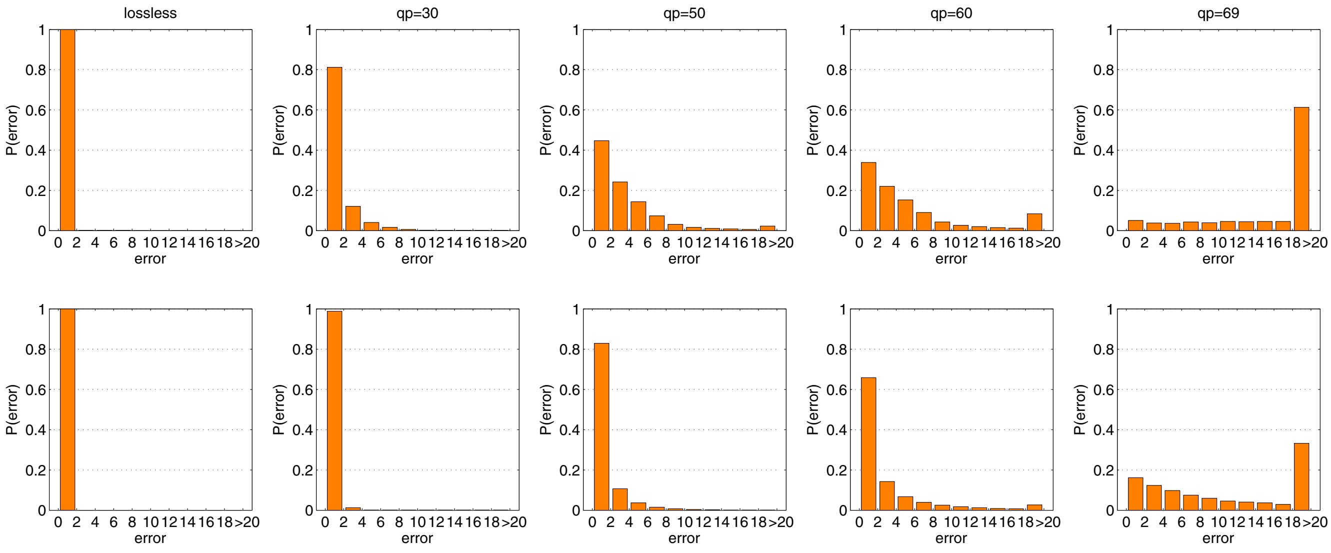

Fig. 3. Error evaluation for our method on Kinect range data on the 15 real world datasets. Histograms resulting from the “coarse”(first row) and “fine”(second row) configurations. From left to right, histograms correspond to the quality profiles 0, 30, 50, 60, 69. The error is measured in cm and each bin corresponds to 2 cm. We grouped all errors larger than 18 cm into the last bin.

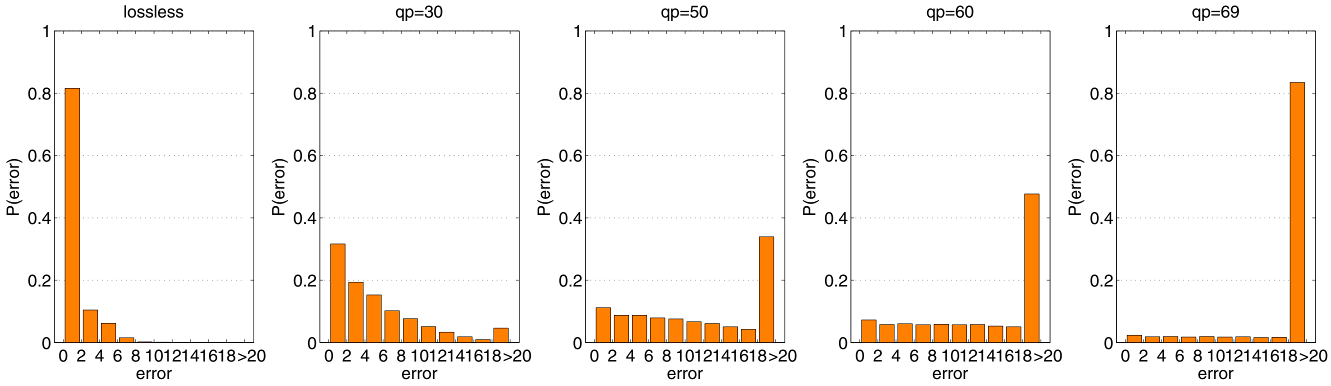

Fig. 4. Error evaluation of the baseline approach using H.264 on a 8 bit channel. Compared to our approach, the distributions show large errors. In the plots, the error is shown in cm and each bin corresponds to 2 cm. We grouped all errors larger than 18 cm into the last bin.

refer to as “coarse” and “fine”. This corresponds to a depth resolutions between $0.008\,m$ and $0.002\,m$ at a maximum sensor range of $12\,m$.

B. Quality of the compressed data stream

The first set of experiments is designed to evaluate the error between a compressed data stream and the original input. We define the error as the absolute difference of the individual range measurements.

We analyze the error for different parameters using over 14,000 depth images from all 15 datasets and present the error distributions by using histograms. Each bin of those histograms depicted in Figure 3 spans a range of 2 cm. A plot with a high peak in the first bin indicates that most measurements are affected by a small or zero error. The five columns of Figure 3 summarize the results for the quality profiles 0, 30, 50, 60, and 69. The rows show the result for the settings “coarse” (top) and “fine” (bottom). The quality profile 69 (maximum compression) typically leads to strong compression artifacts and the data is, at least from our point of view, hardly usable in robotic applications. For quality profiles below 60, the accuracy increases substantially and the corresponding histograms are peaked at the first bin. The quality profile 0 corresponds to a lossless compression of the data stream. As expected, smaller slices and thus finer discretization of the depth measurements yield lower errors. Note that the errors of the images reconstructed using the “fine” resolution with the quality profile 50 appear to be similar to the ones reconstructed using the “coarse” resolution and the quality profile 30.

Additionally, we compared our method to a baseline that consists of rescaling the 11 bit input stream of the Kinect to 8 bit for applying standard H.264 compression. This is equivalent to setting $J = 1$ in our approach. The results are shown in Figure 4. Compared to our approach, this strategy leads to substantially increased errors and the error histograms are not peaked at the smallest bin, even for low quality profile values.

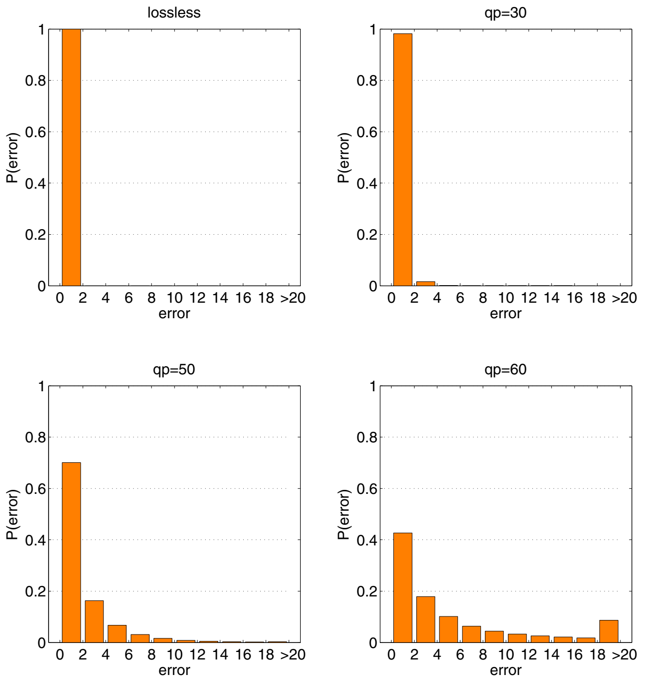

Fig. 5. Pixel error evaluation for our method on Velodyne range data. Every bin spans over 0.20m. The quality profiles used are: 0, 30, 50 and 60. Note the resemblance with the histograms of the second row of Figure 3.

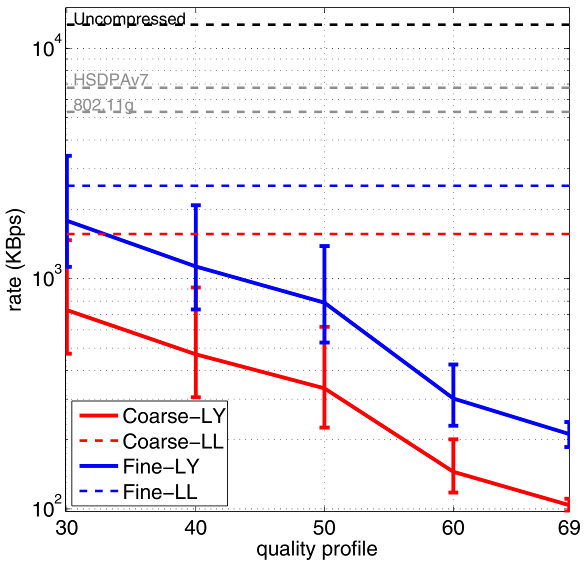

Fig. 6. Average bitrate of Kinect depth data streams over 15 datasets shown in log-scale. We compared "fine" and "coarse" slice setups (blue and red) at different compression profiles. As expected, the fine setup results in a higher bandwidth requirement. Lossless compression results are shown with the dashed curves. Note that there is a large advantage of our approach with respect to the uncompressed stream (black dashed line). In every compression profile, we are able to transmit a full framerate depth stream on standard wireless or cellular network (gray dashed lines).

To illustrate the wide applicability of our approach, we compressed the range data stream of a Velodyne 3D lidar. The Velodyne consists of an array of lasers arranged on a rotating column. We generate a depth image by collecting all the laser measurements of a single rotation. A example depth image is shown in Figure 2. The maximum range is $80\,m$ and we use $J = 16$ slices, which leads to a depth resolution of approx. $0.02\,m$. No changes are needed in our algorithm. The results are shown in Figure 5 and show a similar trend to those obtained with the Kinect sensor.

C. Bandwidth requirements

In this experiment, we evaluate the bandwidth requirements of our streaming method on the Kinect datasets. Figure 6 shows the average bitrate at different compression levels. We evaluate the performance of the "coarse" and "fine" slice setup. With our approach, all quality profiles, even the lossless one, allow for streaming on a standard 802.11g WiFi connection or a HSDPA cellular network at full frame rate. Note that in all settings, we find large bandwidth savings with respect to the original, uncompressed Kinect depth data stream.

Using the "fine" slice setup (blue curve in Figure 6) results in a bandwidth requirement that is approximately twice as big compared to the "coarse" setup (red curve). The same holds when comparing the lossless quality profiles with each other (dashed curves).

Considering all the evaluated datasets, the average size using a *lossless compression* is approx. 0.31 byte per pixel. This is a large compression compared to the uncompressed Kinect depth data of approx. 1.3 byte per pixel given that there is no loss of information. Thus, our approach is twice as effective as the method proposed by Kammerl et al. [4], which achieves a compression of approx. 0.65 byte per pixel at a similar resolution.

qp	size	size+4 slices
0	140 MB	141 MB
30	118 MB	118 MB
40	72 MB	72 MB
50	46 MB	46 MB
60	5.9 MB	6.1 MB
69	0.99 MB	1.2 MB

Fig. 7. Effect on the stream size when increasing the number of unused slices for one dataset consisting of 1133 images.

An interesting feature of using the H.264 compression in our approach is that a slice containing no range information is greatly compressed. In a small test, we added four empty slices for each depth image and evaluated the effect on the resulting size of the stream at different quality profiles. The table in Figure 7 summarizes the results. As can be seen, independent of the quality profile, the increase in bandwidth is limited and always smaller than 1%.

D. Running Time

We evaluated the running time of our approach by using a Intel i7-4770 cpu, constraining the process to use only 1 of the 8 virtual cores. Figure 8 shows how the quality profiles and slices affect average compression time. At the increase of slice quantity (finer multiplexing) there is an increase of computation time that can be traded-off with respect to lower

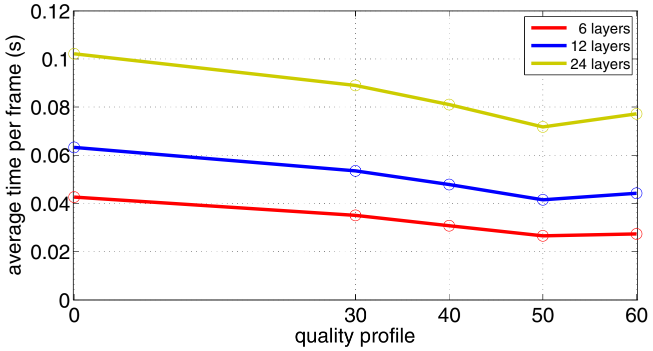

Fig. 8. Evaluation of average compression time per frame.

quality profiles. This results in a running time up to 25Hz. This can be greatly improved by exploiting multi-threading.

E. Using compressed data streams for dense mapping

In many robotic applications, the error in a single depth measurement may not be critical. Many techniques in robotics are founded on probabilistic methods and have been designed to cope with sensor noise. One of those applications is mapping and exploration. Therefore, we qualitatively evaluated the impact of the compression on a state-of-the-art, dense mapping technique. For this, we employ KinFu [13] on a tabletop dataset. Our aim is to illustrate that the resulting maps can be interpreted by human operators even at high compression rates. Figure 9, which depicts map examples build with KinFu, suggests that even with a quality profile of 50, most details of the scene are preserved.

V. Conclusions

Transmitting raw sensor data over a low bandwidth network is problematic for tele-operated or semi-autonomous robots as the sensor data is often large. In this paper, we investigated how to exploit H.264 compression to reduce the size of a range data stream from sensors such as a Kinect camera or a Velodyne 3D lidar to transmit it through a low bandwidth network. We presented a practical and effective solution that exploits the state of the art in video compression, can handle streams with more than 8 bit, reduces the impact of the compression artifacts, and is easy to implement given existing H.264 libraries. We conducted an extensive and thorough experimental evaluation using large amounts of range data streams by analyzing bandwidth, time and accuracy. Additionally, we showed that even a highly compressed stream can be used with a dense mapping technique such as KinFu for building environment models.

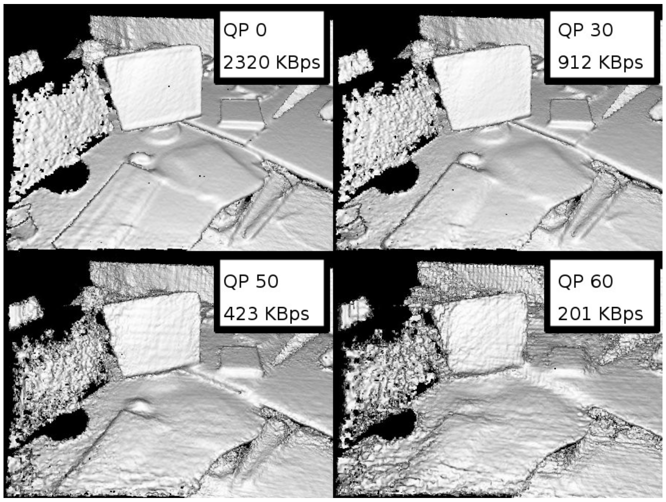

Fig. 9. Dense reconstruction by KinFu of a tabletop scene at different quality profiles. Despite compression artefacts, most objects are visible and well understandable by a human operator.

References

[1] B. B. Chai, S. Sethuraman, and H.S. Sawhney. A depth map representation for real-time transmission and view-based rendering of a dynamic 3d scene. In *First Int. Symp. on 3D Data Proc. Vis. and Transmission*, 2002.

[2] X. Gu, S. Zhang, P. Huang, L. Zhang, S. T. Yau, and R. Martin. Holoimages. In *ACM Symp. on Solid and Physical Modeling*, 2006.

[3] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34, 2013.

[4] J. Kammerl, N. Blodow, R.B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time compression of point cloud streams. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[5] N. Karpinsky and S. Zhang. 3D range geometry video compression with the h. 264 codec. *Optics and Lasers in Engineering*, 2013.

[6] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman. Compression and transmission of depth maps for image-based rendering. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP)*, 2001.

[7] P. Lai, A. Ortega, C.C. Dorea, P. Yin, and C. Gomila. Improving view rendering quality and coding efficiency by suppressing compression artifacts in depth-image coding. In *IS&T/SPIE Electr. Imaging*, 2009.

[8] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner. Real-time, continuous level of detail rendering of height fields. In *Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1996.

[9] B. Merry, P. Marais, and J. Gain. Compression of dense and regular point clouds. In *Computer Graphics Forum*, volume 25, 2006.

[10] Y. Morvan, D. Farin, and P.H.N. de With. Depth-image compression based on an rd optimized quadtree decomposition for the transmission of multiview images. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP)*, 2007.

[11] R.A. Newcombe, A.J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011.

[12] H. Oh and Y.-S. Ho. H.264-based depth map sequence coding using motion information of corresponding texture video. In *Advances in Image and Video Technology*. Springer, 2006.

[13] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point cloud library (pcl). In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[14] R. Schnabel and R. Klein. Octree-based point-cloud compression. In *Symposium on Point-Based Graphics*, 2006.

[15] L. Spinello and K. O. Arras. People detection in rgb-d data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[17] VideoLAN. x264 library. http://www.videolan.org/developers/x264.html.