Compact Encoding of Robot-Generated 3D Maps for Efficient Wireless Transmission

Michael Kaess   Ronald C. Arkin   Jarek Rossignac

College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
{kaess,arkin,jarek}@cc.gatech.edu

Abstract

This work focuses on real-time compression of laser data on board a mobile robot platform. Data is transmitted from the robot over low-bandwidth channels or incrementally in short bursts to a host, where it can be further processed for visualization. For compression purposes, the data is represented as a gray scale depth image. Considered are existing lossless image and file compression schemes (Unix compress, gzip, bzip2, PNG, Jpeg-LS), as well as wavelet transformations tailored to the specific nature of the data. Testing is done on several sets of indoor data acquired by a robot moving through rooms and hallways. The results show that Jpeg-LS compression performs best in this setting.

1 Introduction

Mobile robots can be used to acquire 3D models of the environment for a variety of military and commercial applications. Stereo vision or laser range finders can be used for acquiring the data. Laser sensors provide more accurate 3D information, but vision sensors are smaller and cheaper, and additionally provide texture maps for more realistic 3D models. Processing vision data, however, is usually very resource consuming and can often not be done in real-time, while laser data directly yields a 3D model.

The accuracy of models acquired with range finders [12] is typically higher than of those acquired with stereo vision [6]. Although vision systems not only provide geometry, but also color, which may be used as texture to enhance the acquired model, the onboard processing and transmission of color information may be too expensive in mobile situations. Also, in the absence of vision data, for example in dark environments, the accuracy of the model itself is important. Therefore the sensor chosen for this work is a laser range finder.

Generation of the 3D model can either take place on board the robot, or on a remote host. In either case the data has to be transfered from the robot to the host, typically over a wireless network connection. Wireless connections are restricted in bandwidth. Furthermore, the maximum bandwidth usually cannot be achieved, due to distance between sender and receiver, walls, and interference with other radio signals. This is made even worse in military scenarios, where countermeasures may be present or low bandwidth transmitters are used [1]. This gives rise to the question of how to minimize the amount of transmitted data.

The raw data from the laser range finder can be represented as a sequence of gray scale images. Therefore, in addition to compression methods specific to the nature of the laser data, existing image compression schemes may also be used.

An application of this work is a mobile robot which can move through a building while acquiring and transmitting a 3D model of the environment. Data is transmitted over a low bandwidth channel, allowing use of low cost hardware and coping with interferences, reducing the bandwidth. We show results from our experimental setup with a laser range finder mounted on top of a mobile robot platform.

A different application is data transmission in short bursts, to make detection of the robot more difficult. The transmitted model can be used to familiarize people with a building before entering it, for example in a hostage situation. The lifetime of the robot might be reduced, since it could be destroyed by an enemy when detected. Therefore all the data packets sent up to a specific time must result in a complete model of the environment scanned until then.

Furthermore, some compression schemes, like wavelets and octrees, allow for incremental transmission. A progressive transmission of the model guarantees that, even if the transmission were to be interrupted, the recipient will have an approximating model suitable for use.

The next Section overviews related work. In Section 3 the approach is discussed in detail. Section 4 describes the implementation, followed by a comparison of the results from different compression schemes in Section 5. Section 6 concludes this work.

2 Related work

2.1 3D models

Acquiring compact 3D environment models was approached by different groups in very different ways.

In [6] stereo vision information is used to generate a 3D model while at the same time robot motion is estimated. The resulting models include texture mapping for realistic visualization. This algorithm works under a "suitable planarity assumption", which is fulfilled for non-cluttered indoor environments like corridors. The data is processed offline, so the approach is not suited for real-time applications.

Other groups used laser range finders to acquire similar models. In [13] concurrent mapping and localization for indoor environments is performed using a horizontally oriented laser scanner, while a second upward-pointed laser acquires 3D information of the environment. The resulting 3D mesh is simplified using algorithms from computer graphics making no assumptions about the environment. With a similar setup for an urban environment, [3] describes mapping with scan matching using additional sensors to improve dead-reckoning. An additional camera provides images for texture mapping.

[7] in contrast describes an algorithm that identifies planes in the acquired data using the probabilistic method of expectation maximization to reduce the amount of data. Points which are not part of such a plane remain in the model, allowing for non-flat surfaces with the drawback of increased complexity. An omni-cam provides texture maps for realistic visualization. [12] gives a comprehensive overview of related work.

2.2 Compression

There is a vast amount of literature on compression available. Our search focused on standard text based compression, lossless compression of gray scale images, and wavelets.

Text-based compression schemes are typically based on building a dictionary of symbols and assigning codes to encode redundancy efficiently. Huffman encoding [5] for example chooses the codes based on the frequency of each symbol. The text, a sequence of symbols, is then substituted by the corresponding sequence of codes, and the dictionary itself compressed with a different algorithm. Example implementations of text dictionary based encoders are UNIX compress and GZIP, which implement Lempel-Ziv-Welch (LZW) and Ziv-Lempel (LZ77) respectively.

Image based compression can also be based on dictionaries, the Portable Network Graphics (PNG) image format is an example. Another simple method is Run Length Encoding (RLE), as used in the Tagged Image File Format (TIFF).

More successful, however, are statistical algorithms based on non-linear predictors. Examples are JPEG-LS and CALIC. JPEG-LS is based on LOCO-I (Low Complexity, Context-Based Lossless Image Compression Algorithm), which is described in [14]. Efficient implementations of JPEG-LS exist, and also provide a "near lossless mode", for which the maximal allowed error can be specified. CALIC, in contrast, is a complex predictive scheme, commonly only used as a benchmark. Thus it is not suited for real-time application on robots with restricted processing power, and will not be considered here.

Wavelets, another family of algorithms used for image compression, predict new values from the values of previously decoded neighbors. The image is substituted by a smaller average image and a set of values representing the difference of the original from the predicted values. The algorithm can be applied recursively, resulting in a single average value and a set of mostly small difference values. The overall number of values needed to represent the image does not change, but depending on the quality of the predictor, most values are near zero, and can therefore be encoded efficiently using Huffman [5] or Arithmetic [8] encoding. An accessible introduction to wavelets can be found in [11], a comprehensive tutorial is given in [10].

[9] compares a set of algorithms (UNIX compress, GZIP, LZW, old lossless JPEG, JPEG-LS based on LOCO, CALIC, FELICS, S+P Transform, PNG and other non-gray-scale methods) based on the performance when applied to scanned prints of standard CCITT images as well as gray scale pictorial images. For gray scale images, CALIC and JPEG-LS obtained the highest compression rates, with text based compression (UNIX compress) and old JPEG performing worst.

[2] arrives at a similar conclusion for medical gray scale images, favoring CALIC in almost all cases, directly followed by JPEG-LS. This comparison additionally includes the lossless mode of the JPEG 2000 scheme, performing just below CALIC and JPEG-LS in most cases.

3 Experimental Design

3.1 Acquiring Data

A SICK laser scanner is mounted on top of a Nomad mobile robot platform, pointing upwards with the scan plane perpendicular to the motion of the robot, as shown in Figure 1. [13, 3, 7, 12] concentrate on acquiring accurate models, and simplifying the complete model for storage, transmission and texture mapping. We, in contrast, emphasize the compression of incremental portions of the data, allowing

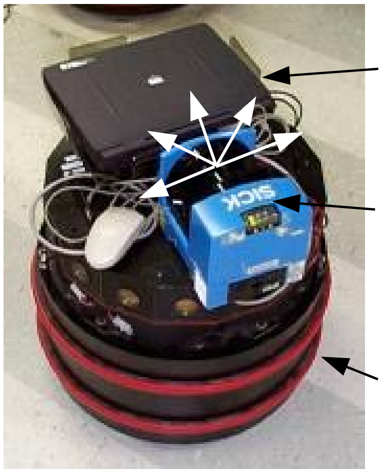


Figure 1: *Nomad 150 mobile robot platform with SICK LMS 200 laser range finder mounted on top.*

for real-time transmission from the robot to a host. If necessary, simplification algorithms can then be applied off-line before visualizing the model. Thus complex processing is postponed until arrival of the data on the more powerful host machine, requiring only low performance on-board processing for the robot. As a result, the mobile part of the system can be smaller, have lower power consumption, need smaller batteries, and become cheaper.

Although we cannot perfectly locate the model in space, due to the error inherent to dead-reckoning, we focus instead on transmitting the acquired 3D information in robot coordinates to the host, which can then either process them or correlate them with other calibration data, such as measurements from a second, horizontally oriented laser scanner. Under the assumption of flat surfaces, it might also be possible to use the 3D information only for localization. It needs to be noted that successive 3D scans generally do not overlap, and therefore only the orientation of the robot can be corrected, not the distance traveled along a hallway for example. Since most of the odometry errors originate from rotations of the robot, this approach seems to be interesting for future work.

3.2 Compression

Although each measurement corresponds to a 3D location and could be represented using 3 coordinates, it is advantageous to encode it as a distance map, with one coordinate per sample, due to the resulting data reduction. Moreover, the raw data can be seen as a cylindrical depth map of the environment, having one dimension in the radial scan direction, and a second dimension along the path of the robot. The raw data of the laser sensor consists of 13 bit distance measurements. Visualizing the data using a grey-level coding of depth produces images similar to the one in Figure 2(a). Thus, we can treat the raw data as an image. For incremental transmission, successive portions of the data need to be compressed, corresponding to strips of such depth images. Existing image compression algorithms can be applied.

The noise in the picture will reduce the performance of the compression algorithm, so smoothing of the image before compression might be appropriate. This also reduces the noise in the resulting 3D model, but it also causes loss of information. Since the data is processed on the host side, lossless transmission is preferable. Progressive transmission is desirable, providing lossless transmission whenever the connection allows. We concentrate on lossless compression schemes, but also mention some lossy compression methods for comparison.

Besides existing text and image compression software, different wavelet transforms were evaluated. Wavelets split the data into low- and high-frequency information. They use a prediction function to predict a sample from its neighbors. The goal is to find a suitable predictor, so that the differences are minimal. The transform is applied recursively on the remaining averages, finally leaving one average value from all the data, plus a set of small difference values. The overall number of values does not change. However, the set of near-zero values can be represented much more compactly than the original data, using Huffman or Arithmetic encoders.

Wavelet transforms are related to Fourier transforms from signal processing. The main difference is that they are easier to handle, mainly in terms of an easily derivable inverse transform. They can also handle boundaries, typically encountered in the spatial domain of images.

The simplest wavelet transform, the Haar Wavelet, predicts a constant function, and replaces two neighboring samples a and b by their average $s = \frac{a+b}{2}$ and the difference $d = b - a$. As described in [10], this can be rewritten as

$$\begin{aligned} d_{n-1,l} &= s_{n,2l+1} - s_{n,2l} \\ s_{n-1,l} &= s_{n,2l} + \frac{d_{n-1,l}}{2}, \end{aligned}$$

allowing for efficient in-place calculation and simple derivation of the inverse transformation.

The Linear Wavelet Transform [10] extends the Haar Wavelet from constant to linear functions. It needs to examine both, the left and the right neighbors of a sample to predict its value, and is given by

$$\begin{aligned} d_{n-1,l} &= s_{n,2l+1} - \frac{s_{n,2l} + s_{n,2l+2}}{2} \\ s_{n-1,l} &= s_{n,2l} + \frac{d_{n-1,l-1} + d_{n-1,l}}{4}. \end{aligned}$$

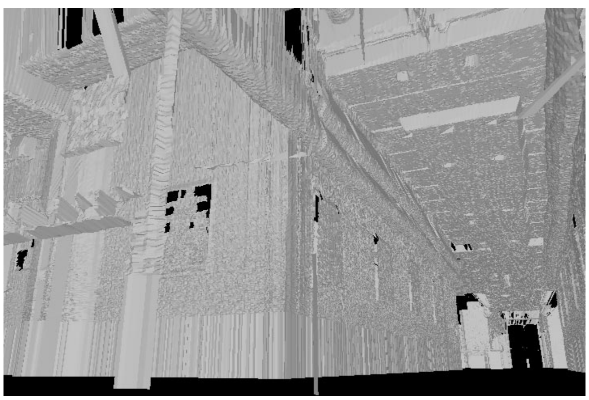
Assuming that flat surfaces dominate, typically the case for indoor environments, the wavelet needs to be able to predict lines. Therefore the Linear Wavelet should be well suited along the direction of movement of the robot, that is the horizontal direction in Figure 2(a). For the perpendicular laser scan direction, lines imply a trigonometric function on the radial distance measure. The Linear Wavelet translated into Cartesian space given the radial distances is defined by

$$\begin{aligned} d_{n-1,l} &= \frac{2\cos 0.5^\circ}{\frac{1}{s_{j,2l}} + \frac{1}{s_{j,2l+1}}} \\ s_{n-1,l} &= s_{n,2l} + \frac{d_{n-1,l-1} + d_{n-1,l}}{4}. \end{aligned}$$

Our Radial Line Predictor uses this wavelet for compression along the scan direction, and the Linear Wavelet along the movement direction of the robot.

Care has to be taken to correctly implement boundaries, allowing the compression of arbitrarily sized images, not restricted to powers of 2. The resulting data is compressed using an adaptive Huffman encoder.

(a) The raw data of the laser scanner as depth image.



(b) 3D model view

Figure 2: *Raw data and 3D model view: Darker values in the depth image (a) represent shorter distances from the robot. Each* 180° *scan corresponds to one pixel column in the image. The ceiling is always visible along a horizontal line through the center of the image, since the laser pointed upwards. The robot drove along a corridor (left to right in the image), the right and left walls are visible in the upper and lower part of the depth image, respectively. Near the center of the image, the robot turned* 90° *near a glass door entrance, and continued into another corridor. The white lines on the corridor walls are errors caused by picture frames deflecting the laser ray. The 3D model in (b) shows a view from outside the glass door entrance. The robot turned from the left corridor into the center corridor of the building. Note that the 3D model contains more data then shown in (a), and that it was created after lossless transmission, that is from unmodified laser data.*

4 Implementation

The overall system architecture is shown in Figure 3.

4.1 Hardware

The SICK LMS 200 laser range finder, mounted on top of a Nomad 150 mobile robot (see Figure 1), scans 180° in 0.5° steps from one side over the top to the other side, returning a 13 bit integer value representing the distance to the next object for each of the 361 steps. About 38 complete 180° scans are performed per second, but in the current implementation only 5 are available because of restrictions of the standard serial RS232 interface. The robot moved on average with a speed of about $0.25\frac{m}{s}$, that is one scan is performed every $5cm$. In the perpendicular scan direction, the same resolution is achieved for objects at a distance of $6m$. Since the calculations performed on the robot are not complex, it is possible with a different serial interface to use all available scans with the robot moving at $2.4\frac{m}{s}$, resulting in the same model quality. On the other hand, the model quality could be increased by acquiring more data in the same time period. It makes little sense to increase the horizontal resolution above the vertical resolution for these kinds of indoor environments. If objects on average are about $3m$ distant, an optimal value for the horizontal resolution is $3m \cdot \tan(0.5) = 2.6cm$, yielding a robot speed of about $1\frac{m}{s}$.

The laser scanner operates in the non-default millimeter mode. In this mode, the 13 bits returned for every single distance measurement represent distances from 0 to 8.191 meters, where the highest value

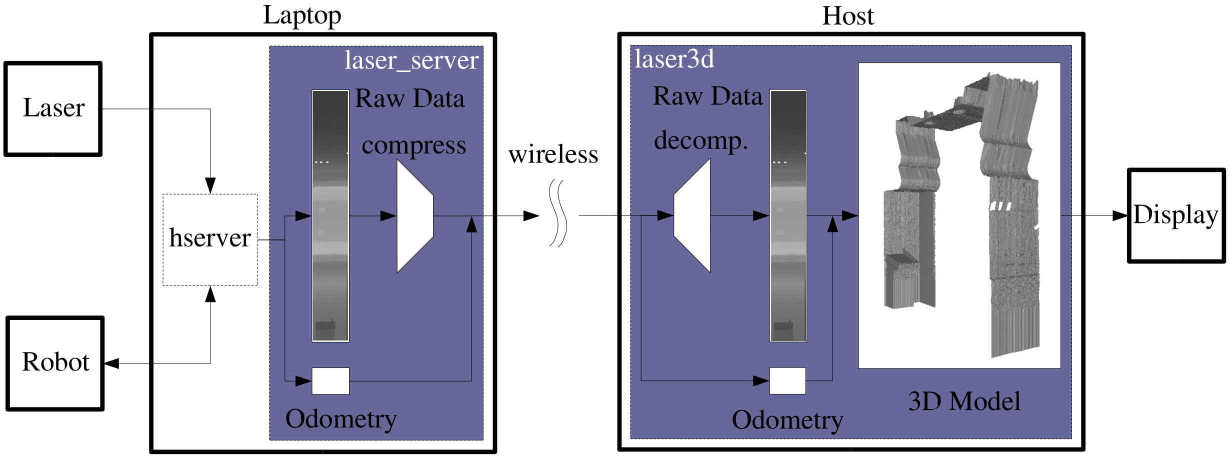


Figure 3: *System architecture: The mobile part of the system consists of a laptop and a laser scanner, both on board a robot.* HSERVER *is running on the laptop to control the robot and acquire data from the laser.* LASER_SERVER *requests laser data and odometry from* HSERVER, *compresses the data, and sends it together with the position information over a wireless network connection to the remote host. On the host side,* LASER3D *recovers the original data, converts it to a 3D model, and displays it.*

8191 represents all distances of $8.191m$ and more. According to the technical information of the scanner, the statistical error in the range from $1m$ to $8m$ is at most $\pm 15mm$. Nevertheless, the resulting models show much more detail compared to the ones generated in centimeter mode. Some experiments with the laser pointed at a planar surface reveals this statistical error as shown in an example scan in Figure 4. The distance to the wall is $270mm$, the statistical error from the first 30 points averaged over several frames is in the order of $\pm 8mm$, where this value increases slightly with the distance.

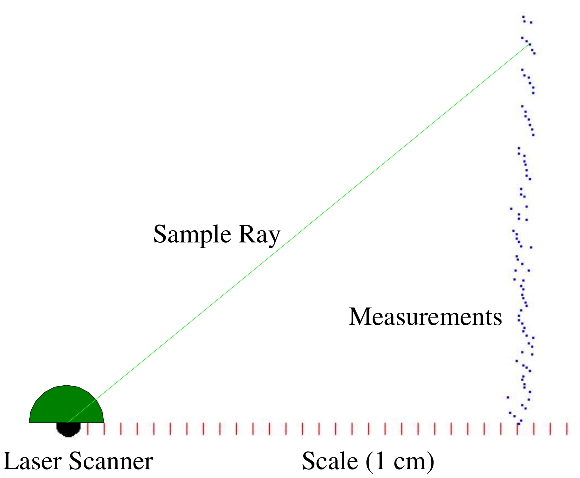


Figure 4: *The dots on the right represent a straight wall as seen by the laser scanner. A scale is shown with a distance of* $1cm$ *between adjacent lines.*

4.2 Software

To control the movement of the robot and record the data from the laser scanner, HSERVER and a modified version of LASERFIT, called LASER_SERVER, are used, both part of the freely available MISSIONLAB[1] software from Georgia Tech. HSERVER can either be used directly to teleoperate the robot, or be interfaced to MISSIONLAB to allow autonomous execution of missions.

The data from the laser together with odometry and time stamps can be recorded to a file to simplify the testing of different compression algorithms. Recorded data can then be replayed with LASER_SERVER in real-time, making it possible to test the software for time constraints. Additionally, LASER_SERVER can be used to compress and transmit the data incrementally over network in real-time to the visualization software LASER3D, which is described next.

The modified version of LASER_SERVER, and the visualization software LASER3D are also freely available[2] on the web.

Compression and decompression are built into LASER_SERVER and LASER3D, respectively. This allows for real-time compression and transmission of the data over a (wireless) network connection from the robot to a host. A special mode allows for fast compression of the data, without actually displaying a 3D model, printing statistical results for comparison of the different algorithms.

[1] `http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/`

[2] `http://www.cc.gatech.edu/ai/robot-lab/research/3d/`

Implemented and tested lossless compression modes include uncompressed, bit stuffing, UNIX compress, GZIP, BZIP2, JPEG-LS, and different kinds of wavelets. Additionally, nearly lossless JPEG-LS with specifiable maximal error is provided as a lossy compression mode.

Each packet starts with one byte specifying the compression algorithm used. The next three bytes contain the byte length of the complete packet. The next byte specifies the number of scans n in the packet, followed by $3n$ floats representing the poses of all scans. The pose information should also be compressed, but since it represents only about 2% of the uncompressed data, this is omitted for now. The second part of the packet contains the compressed laser range data.

Uncompressed mode sends 361 measurements (2 bytes each) per scan. This sums to 722 bytes for each scan. Bit stuffed mode uses only 13 bits per measurement, resulting in $361 \cdot \frac{13}{8} = 587$ bytes. UNIX compress[3] , GZIP[4] and BZIP2[5] call the corresponding shell command on the 16 bit raw data (PGM image format without header). PNG calls ImageMagick[6] on 16 bit raw PGM format. JPEG-LS is implemented by calling the 16 bit encoder and decoder loco16e/loco16d[7] and nloco16e/nloco16d for lossless and near lossless mode, respectively.

The wavelet transforms are implemented in LASER_SERVER/LASER3D, while compression is done by calling a command line Huffman encoder[8] .

4.3 Visualization

After decompression on the host side, the model is displayed using OpenGL. The raw data of each scan is converted from cylindrical to Cartesian coordinates. The raw integer data d representing the distance to the next object in millimeters is scaled to meters and transformed to Cartesian floating point coordinates using the pose of the robot x, y, θ, the angle α describing the direction of the scan within the scan plane from $-\frac{\pi}{2}$ to $+\frac{\pi}{2}$, the height of the center of the laser scanner $H_R = 0.52$, and the offsets x_O, y_O, z_O:

$$
\begin{aligned}
x &= x_R + \frac{1}{100} d \cdot \cos(\alpha)\cos(\theta) + x_O \\
y &= y_R + \frac{1}{100} d \cdot \cos(\alpha)\sin(\theta) + y_O \\
z &= H_R + \frac{1}{100} d \cdot \sin(\alpha) + z_O
\end{aligned}
$$

To get the actual mesh, two adjacent scan points P_1 and P_2 taken at angles α and $\alpha + 0.5°$ of the same complete scan and the corresponding two points Q_1 and Q_2 of the next scan are connected to form two triangles $P_1P_2Q_1$ and $Q_1P_2Q_2$. For each triangle ABC the corresponding normal vector

$$n = \frac{AB \times AC}{||AB \times AC||}$$

is calculated. To safe time the triangles are not passed separately to OpenGL but in strips, where a strip consists of all triangles between two scan planes. $n+2$ points are necessary to describe n triangles. Besides the first 2 points, each following point describes another triangle and is passed together with the corresponding normal vector. To avoid transmission of that data for every updated view of the model, OpenGL lists are used to store the data and the necessary transformations are then performed on these lists.

Different display modes allow for shaded models as well as visualization of underlying data by assigning random colors to triangles, quads or strips. The program also allows for different floor modes including a checker board to allow better estimation of distances.

Since the laser scanner is mounted on top of the robot with its center at a height of $52cm$, the resulting mesh starts at that height. Connection to the floor is done by projecting the first and the last scan point of every 180° scan down to height 0. This makes the model look more realistic, but is also dangerous since it may pretend a free space were obstacles actually are.

At the beginning the observer is located in the model at the same point the robot started, with the same orientation. The height of the eyes are assumed to be at $1.7m$, more than a meter above the actual location of the laser scanner.

The light source used to make the model more realistic is positioned in a height of $0.8m$ over the eyes of the observer. Usage of the OpenGL two-sided light model and the front-and-back polygon mode makes it possible to see scanned surfaces also from the other side.

The user can navigate through the model in arbitrary directions and orientations.

5 Results

To compare the different algorithms, five data sets were recorded. Data Set 1 and 2 were acquired mainly along corridors, Set 3 in a mixture of rooms and corridors, and 4 and 5 inside rooms. For a given bandwidth of 10 kBytes per second, the resulting transmission times for a packet of 20 scans are compared in Table 1.

[3]Linux compress 4.2.4

[4]Linux gzip 1.3.3, 2002-03-08

[5]Linux bzip2, Version 1.0.2, 30-Dec-2001

[6]Linux ImageMagick 5.4.7, "convert -quality 100 x.pgm x.png"

[7]Linux loco16e/d, JPEG-LS Reference En/Decoder - V.1.00X, freely available for testing and evaluation, see `http://www.hpl.hp.com/loco`

[8]Compressors, Version 1.5, David Bourgin, dbourgin@ufrima.imag.fr

Table 1: *Transmission times for packets containing 20 scans, including header and uncompressed odometry information. Assumed is a transmission rate of 10 kBytes per second. Evaluation is done on five different data sets. Data Sets 1 and 2 are mainly corridors, 4 and 5 are inside a room, and 3 is a mixture of rooms and corridors.*

Data Set		1	2	3	4	5	Average
	Number of scans	5600	4500	800	800	1400	
	Size of raw data in bytes	4043200	3249000	577600	577600	1010800	
	Compression method	Transmission time in seconds (rounded)					
Raw formats	Raw data	1.469	1.469	1.469	1.469	1.469	1.469
	Bit stuffing	1.197	1.197	1.197	1.197	1.197	1.197
Text based compr.	Unix COMPRESS	1.112	1.149	1.222	1.382	1.326	1.238
	GZIP	0.848	0.874	0.914	1.028	0.995	0.932
	BZIP2	0.686	0.724	0.768	0.910	0.864	0.790
Image based compr.	PNG	0.870	0.907	0.953	1.102	1.058	0.978
	Haar Wavelet	0.854	0.891	0.904	0.985	0.948	0.917
	Linear Wavelet	0.814	0.865	0.861	0.970	0.917	0.886
	Radial Line Predictor	0.812	0.864	0.859	0.969	0.915	0.884
	JPEG-LS	0.501	0.545	0.529	0.618	0.572	0.553
Lossy compression	JPEG-LS, max err 1	0.350	0.395	0.385	0.475	0.428	0.407
	JPEG-LS, max err 2	0.291	0.332	0.325	0.413	0.367	0.346
	JPEG-LS, max err 4	0.237	0.277	0.264	0.346	0.303	0.285

The results confirm [9] and [2] in JPEG-LS as being the best compression scheme, and text-based compression algorithms Unix COMPRESS and GZIP performing the worst. BZIP2 does significantly better than the latter two, but is still far behind JPEG-LS. It also needs to be mentioned that BZIP2 compression takes a multiple in time of the other algorithms. As one would expect, lossy compression schemes perform significantly better, even for a maximum error of only $\pm 1mm$.

As one would expect, the Linear Wavelet Transform performs better than the Haar Wavelet Transform. However, the difference is relatively small. Haar should not be a good predictor for the data, because at least in scan direction the data will in general not be constant. The small difference in performance suggests that the Linear predictor is also not suited. Radial Line Prediction performs only marginally better than the Linear Wavelet, since a linear function in the measured distance is a good approximation to the lines in radial space. All three algorithms perform better than GZIP, but are significantly below the performance of JPEG-LS.

Table 2 shows results for Data Set 3, including theoretical transmission times calculated from the entropy values for Linear Wavelets before Huffman encoding. Even the theoretically optimal encoding does not reach the JPEG-LS result. This suggests that there are better predictors for the data. One possible explanation for the bad performance provides the inherent noise of the data, visible in Figure 4, which is not addressed by our approach.

Table 2: *Transmission time in seconds for packets of size 20 for Data Set 3. For the output of Linear Wavelet application, specified are results from Huffman and Arithmetic encoders, as well as theoretical limits of compression given by the entropy of the data. JPEG-LS performance is listed for comparison.*

Compression Method	Transmission time (in s)
Linear Wavelet + 8 bit Huffman	0.861
Linear Wavelet + Arithmetic	0.804
theoretical limit arithmetic, 8 bit	0.734
theoretical limit arithmetic, 16 bit	0.537
JPEG-LS	0.529

Beyond the comparison of different algorithms, Table 1 also shows a difference between corridor and room data. Corridors contain less complex surfaces compared to rooms, the data can therefore be easier compressed, speeding up transmission.

Another interesting view is provided in Table 3, showing the dependence of compression ratio on the packet size. Note that transmission time is given for sets of 20 scans, even though the packets have different sizes. For big packets, the packet length has not much influence on the time. For small packets, increasing by a few scans reduces transmission time significantly. The values suggest that packet sizes between 20 and 50 should be chosen as a compromise between update frequency and compression ratio.

Table 3: *Influence of packet size (in number of scans) on transmission time (in seconds for* 10 *kBytes per second transmission rate) for selected algorithms. For evaluation all five data sets were used and the average is reported.*

	Packet size			
	10	20	50	100
Compr. method	Transmission time (in s)			
BZIP2	0.888	0.790	0.702	0.659
JPEG-LS	0.578	0.553	0.537	0.530
Linear Wavelet	0.927	0.886	0.857	0.847

6 Conclusion

This paper compared compression methods in terms of transmission time of laser generated 3D maps. The goal was compact incremental transmission of the data, not simplification of a complete 3D model, as is commonly used for visualization purposes. Compared were transmission times using existing image and text based compression algorithms, and wavelet algorithms tuned to the specific nature of the raw laser data.

For most such applications, the usage of lossless JPEG-LS is recommended, resulting in lowest transmission times compared to all lossless compression schemes tested. Since it is a complex predictor scheme, some demand of on-board processing can be expected. If on-board processing is severely restricted, wavelets with Huffman encoding should be preferred.

If real-time is an essential part of the application, and certain minimum transmission rates cannot be guaranteed at all times, then progressive transmission methods should be used. Wavelets are suitable, but this application was not further explored here.

If very small errors can be accepted, lossy JPEG-LS should be used to reduce transmission times significantly. Of advantage is the guaranteed maximum error, specifiable by the user.

Acknowledgments

The authors would like to thank Jonathan F. Diaz and Alexander Stoychev for their implementation of LASERFIT. The Georgia Tech Mobile Robot Laboratory is supported from a variety of sources including DARPA, Honda R&D, C.S. Draper Laboratory, and SAIC.

References

[1] R. Arkin, R. Burridge, and T. Collins, “Control Systems for Semiautonomous Robot Groups”, in submission, 2003.

[2] D. A. Clunie, “Lossless Compression of Grayscale Medical Images - Effectiveness of Traditional and State of the Art Approaches”, *SPIE Medical Imaging*, 2000.

[3] C. Frueh, A. Zahkhor, “Fast 3D Model Generation in Urban Environments”, In *International Conference on Multisensor Fusion and Integration for Intelligent Systems,* Baden-Baden, Germany, pp. 165-170, August 2001.

[4] J.-S. Gutmann, K. Konolige, “Incremental Mapping of Large Cyclic Environments”, In *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 318-325, 1999.

[5] D. A. Huffman, “A Method for the Construction of Minimum Redundancy Codes”, *Proceedings of the IRE*, pp. 1098-1101, 1952.

[6] L. Iocchi, K. Konolige, M. Bajracharya, “Visually Realistic Mapping of a Planar Environment with Stereo”, *International Symposium on Experimental Robotics (ISER)*, pp. 521-532, 2000.

[7] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, S. Thrun, “Using EM to Learn 3D Models of Indoor Environments with Mobile Robots”, In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 329-336, 2001.

[8] W. B. Pennebaker and J. L. Mitchell, “An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder”, *IBM J. Res. Dev.*, pp. 717-726, 1988.

[9] A. E. Savakis, “Evaluation of Lossless Compression Methods for Gray Scale Document Images”, *International Conference on Image Processing*, Vancouver, Canada, September 2000.

[10] P. Schroeder, “Wavelets in Computer Graphics”, *SIGGRAPH Course Notes*, 1996.

[11] E. J. Stollnitz, T. D. Derose, D. H. Salesin, “Wavelets for computer graphics: theory and applications”, Morgan Kaufmann Publishers, 1996.

[12] S. Thrun, “Robotic Mapping: A Survey”, In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*, 2002.

[13] S. Thrun, W. Burgard, D. Fox, “A Real-Time Algorithm for Mobile Robot Mapping with Applications to Multi-Robot and 3D Mapping”, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[14] M. J. Weinberger, G. Seroussi, G. Sapiro, “LOCO-I: A Low Complexity, Context-Based Lossless Image Compression Algorithm”, *IEEE Data Compression Conference*, 1996.