SMOKEBOT

Mobile Robots with Novel Environmental Sensors for Inspection of Disaster Sites with Low Visibility

Project start: January 1, 2015

Duration: 3.5 years

Deliverable 6.3

Technical Specifications for Bandwidth Supervision and Data Transfer Adaptation

Due date: month 40 (April 2018)

Lead beneficiary: LUH

Dissemination Level: Public

Main Authors:

Paul Fritsche (LUH)

Version History:

0.1: Initial version, PF, March 2018

0.2: Corrected version, PF, April 2018

Contents

1 Abstract

This deliverable describes the methods used in Task 6.3 (“Robot-Interface Data Transfer”) to ensure a reliable data transfer. Task 6.3 is part of Work Package 6 (“Human-Robot Interface), which is concerned with methods for two-way human-robot communication to provide an intuitive and clear user interface to enable quick interaction. The objective of Task 6.3 is to reduce the amount of data to be send to the operator in dependency of the available transmission rates and the current focus of the mission.

2 Introduction

To effectively control a partly autonomous robot remotely, a user interface is needed to provide the operator with the necessary information about the state of the robot and the current environment. This task is especially challenging in disaster robotics, where the robot often operates in an unknown environment with low-bandwidth and erroneous data communications.

In this document various mechanisms are presented to adapt the quality of the current data transfer and to reduce the amount of data to transport. The communication between the robot and the user interface is realized using Wi-Fi.

In the SmokeBot project, the robot is equipped with several sensors for environmental inspection. These sensors include LIDAR, RGB and thermal cameras, radar and gas sensors. These sensors deliver information about the environment, which are useful for the operator in different situations with different visibility constraints.

In this document, the produced data by the sensors is analyzed in terms of bandwidth usage and different methods are introduced to reduce the amount of data to transfer, while retaining the main information for the operator. Also a comparison between the TCP and UDP transport protocol for the described use case is presented and the introduced methods are evaluated.

The document is organized as follows:

- First, a system overview over the robot and the sensor data is given
- Second, the data link between the user interface and the robot is described
- Third, methods to robustly transmit data for cameras, maps and LIDAR are described
- Afterwards, some experimental results are presented
- In the end, we conclude the deliverable with a short discussion

3 System Overview

The SmokeBot robot is equipped with several sensors, offering data for different applications like mapping, localization and navigation. These applications also produce data such as environmental maps, the robots position or information about obstacles in the environment. The data from the sensors, as well as the data produced by the applications can be of interest to the user and it should be possible to visualize them for remote operation.

This document is concerned with the robust data transfer between the robot and the operator interface, which are connected over Wi-Fi, using the Robot Operating System as a middleware. To serve this purpose, some sensors which are particularly important for an operator will be introduced in more detail to determine the requirements for an effective data transfer over Wi-Fi.

The system overview is visualized in Figure 1. The data for the user interface is transmitted using two nodes. The first node gets data from the sensors and other sources and processes the data to send it over Wi-Fi. The second node receives the data and reassembles the send messages. Depending on the current situation, the data transfer nodes can be configured, which data should be send over the Wi-Fi connection.

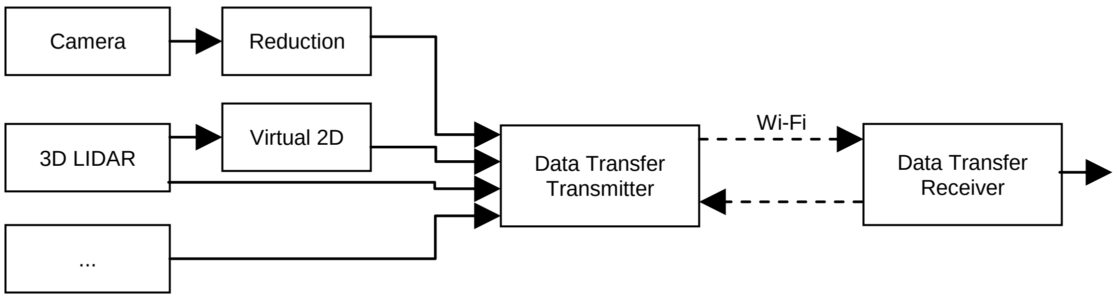


Figure 1: System overview

One important sensor for mapping and navigation is the 3D LIDAR, which can be useful for an operator to get a 3D view of the current environment. The used LIDAR in the SmokeBot project is a Velodyne VLP-16, which can output up to 300.000 points per second with a 360° field of view and 16 different layers to record a detailed representation of the robot environment. In the test scenario the LIDAR recorded a 360° scan with an angular resolution of 0.25° and a frequency of 10Hz, resulting to over 23.000 points per scan. With each scan point containing the x, y and z coordinates having the size of 3 Bytes, this totals in size of 0.7 MB per Scan or a needed bandwidth of 7 MB/s.

Another important sensor in the SmokeBot project is the thermal camera FLIR A655sc. Using a thermal camera instead of a RGB camera has the advantage, that it is possible to get a detailed picture of the environment even in smoky conditions. Otherwise, transmitting the thermal image is analog to transmit a grayscale image. The used thermal camera produces up to 50 images per second with a resolution of 640x480 pixel and 16 Bit per sample. This results in a size of 0.62 MB per Message or a needed bandwidth of 31MB/s.

Table 1 summarizes the data to transmit. Regarding the operating environment of the robot and the reduced and possibly lossy bandwidth, it already becomes clear that the data needs to be reduced before transmitting it to the operator interface. Before describing the methods to reduce the sensor data in section 4 and 5, the communication between the robot and the operator interface is examined in more detail.

Table 1

Data	Size per Message	Bandwidth	Hertz [1/s]
Thermal Camera	0.62 MB	31 MB/s	50
LIDAR	0.7 MB	7 MB/s	10

3.1 Data Link

The communication between the operator and the robot is realized using the Robot Operating System (ROS), which provides an easy to use and extensible framework to transmit all kinds of data. The operator and the robot are connected via Wi-Fi. Especially in disaster robotics, where the environment of the robot is often unknown, this provides a challenge due to limited transmission rates and possibly lossy connections because of bad Wi-Fi coverage.

Using ROS, data is generally send as messages, which can contain different types of information such as sensor data, teleoperation commands, the robots position or general status information of the robot. To send these messages over a network, ROS currently implements the two standard network protocols TCP and UDP. Both protocols split up each message into several packets at the transmitter side, send them over the network, and assemble the message again at the receiver side.

TCP is a reliable and ordered transport protocol, which guarantees that the packets are transmitted in the correct order and without any loss. Each received packet is acknowledged by the receiver and if a packet is lost, it is send again by the transmitter until it is acknowledged. This means, that the full message is always transmitted without any missing packets. However, a lossy connection can significantly extend the transportation of one message, because many packets need to be transmitted multiple times, which also increases the limited network traffic.

Contrary, UDP is a connectionless transport protocol. Each packet is only send once and the receiver does not acknowledge a received message. This makes it possible, that a message can always be send in the same amount of time and no additional traffic due to retransmissions is generated. However, there is no guarantee that the complete message is received, because packets can get lost again. Depending on the datatype, a single lost packet can mean, that the whole message is lost. This applies especially to large message types such as images or point clouds.

For the user interface it is important, that the transmitted data is as up-to-date as possible. Therefore, the message latency should be as low as possible. Also, depending on the Wi-Fi quality, the transmission bandwidth can vary significantly and the amount of transmitted data should be adapted accordingly.

The robot has the ability to drop repeaters when the signal strength gets too low. However, the use of repeaters increases the risk of lost packets and increases the latency for each additional repeater. Especially increasing the latency has a negative impact on TCP throughput, because the sender spends more time idle waiting for acknowledgments, while it has no effect on UDP.

For these reasons, the communication between the robot and the user interface should be realized using UDP to better control the amount of sent data and to guarantee minimal latency. However, additional requirements have to be taken to realize an effective data transfer, especially in the case of large messages. In the following two sections, methods to transmit two types of data, images and point clouds, are introduced, which are robust against lossy connections.

4 Image Transportation

Camera images are a very important source of information for the operator, as they contain very detailed information about the environment. However, the transmission of images can be very bandwidth intensive, depending on the resolution and the frequency of the images.

The goal for the user interface is to display an as much detailed as possible image at a sufficient rate. If the image has a high resolution and the bandwidth is too low, it could take too long to transmit the image and the operator cannot react to an event in time. If the image has a low resolution, the image is transmitted faster, but the operator could miss important details due to the reduced details in the image.

Using ROS, images are normally transmitted using the image_transport package. This package offers different approaches to reduce the amount of data per image to transmit with different compression methods. Currently it is possible to transmit images as jpeg or theora compressed files. These compression methods reduce the size per image, but they are also vulnerable to packet loss. A single lost packet of an image message might lead to the loss of the whole image and it cannot be reconstructed by the user interface. Also, they are not adaptable to varying bandwidth limits, as the compression level must be set manually.

As an example, transmitting a 320x240 gray scale image, where each pixel has a 1 Byte value, has a size of 76800 Byte. With the maximum packet size for Ethernet being 1500 Byte, and a header size of 36 Byte, this means that at least 53 packets have to be transmitted for one uncompressed image. When assuming a packet loss rate of 1%, this means that there is only a 59% chance to transmit an image successfully. Therefore, another transport mechanism for images is introduced in this section, which is robust to lost packets and varying network bandwidth.

4.1 Image Splitting

As described before, an image should be transmitted at a constant rate with low latency. If the data link is good, the image should be transmitted with full details, but if there is packet loss or a low bandwidth, it should be still possible to display a low quality image.

One solution would be to split up the image into several regions of interest and transmit each region in one packet. However, this has the disadvantage that parts of the image could not be recovered due to packet loss. Instead the splitting should be able to recover information from all regions of the image, with increasing details when the data link is good.

Table 2

(a)

0	2
3	1

(b)

0	2	0	2	0	2	0	2
3	1	3	1	3	1	3	1
0	2	0	2	0	2	0	2
3	1	3	1	3	1	3	1

One splitting strategy to achieve this is described in [1], where the image is split up pixelwise to transmit information from all image parts in each packet. The split pattern is shown in Table 2(a). This pattern is applied to the whole image. An example for an 8x4 pixel image is shown in Table 2(b), where each subimage consists of pixels marked with the same number.

This pattern uses the fact that neighboring pixels are often related. So each subimage contains information from the whole image and the more subimages are transmitted, the more detailed the whole images becomes. If not all subimages are transmitted, the missing parts must be interpolated as described in the next section.

Table 3

(a)

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

(b)

0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

If an image needs to be split into more than four subimages to be small enough for transmission, the pattern is used recursively. For example, subimage 1 is again split up using the pattern of Table 2(a). This results in the pattern shown in Table 3(a) with 16 subimages. Each splitting multiplies the step size between neighboring pixels p by a factor of 2. The number of the subimages I is given by

$$p = 2^d$$
$$I = p^2 = (2^d)^2$$

One subimage has to be transferred by one Ethernet frame. As discussed before, a standard Ethernet frame, using the ROS protocols, has a payload of 1464 Bytes. With a size of each pixel of 1 Byte, it is possible to transmit 1464 pixels per subimage and that at least 53 subimages are needed to transmit the full image. To achieve this, three recursion are needed, resulting in 64 subimages as shown in Table 4.

Table 4

no. of steps (d)	0	1	2	3	4
pixel distance (d)	1	2	4	8	16
no. of subimages (I)	1	4	16	64	256

4.2 Image Reconstruction

On the receiver side the image has to be reconstructed again. To do this the user interface listens for image segments and reconstructs an image for segments with the same time stamp. The more segments are received the more detailed the image can be displayed. To do this, different image interpolation methods can be used.

The currently implemented interpolation method uses a nearest neighbor interpolation. For each missing pixel, the nearest pixel of a received segment is searched. An example of this is shown in Figure 2, with only three received segments. For comparison, the full image is shown in Figure 3, containing of 64 subimages.

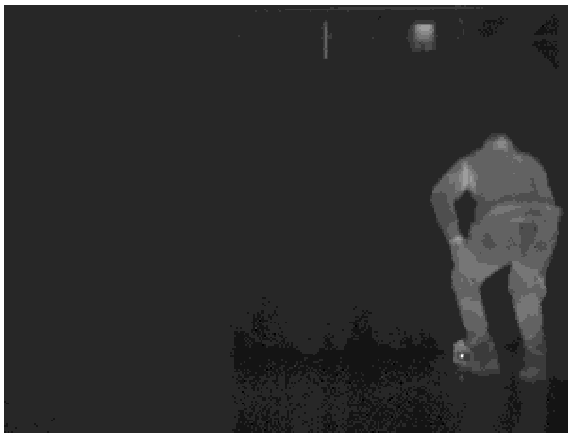


Figure 2: All packets are succesfully transmitted

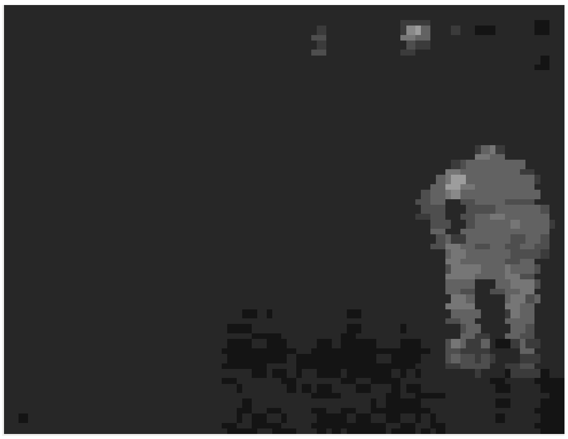


Figure 3: Only three packets are succesfully transmitted

The time a new image is constructed can also be varied. For example, when the robot is moving, new images should be constructed quickly with the same frequency as the incoming data. On the other side when robot is standing still, the image can be updated more slowly. This has the advantage, that the missing subimages can be integrated into the reconstructed image over time to increase the details in the image.

5 Map Transportation

Another useful information for the user are maps of the environment in which the robot is moving. One popular type of map are grid maps, which have some similar properties like images. In this kind of map, the environment of the robot is divided into a regular grid of cells, where each of these cells contains a probability if it is occupied or not. Having these information, the robot is able to localize itself and to perform path planning. For a user these maps are useful to see the position of the robot in context of the whole scene.

However, the resolution of these grid maps can usually change during operation of the robot and are generally higher than the resolution of a typical camera. However, grid maps can still be transported in the same way as described in the last section. To do this, they are first converted to a normal image with additional information like the origin of the map and the resolution. This additional information needs around 120 Byte, making it possible to transmit up to 1300 pixel per message, where each pixel is a value between 0 and 100, depending whether the pixel is free or occupied. In the next step the number of recursion steps for the splitting pattern is determined based on the height and width of the map. To reduce the number of recursion steps, the map can also be down sampled. This is useful, because ROS is not designed for sending large number of messages.

For example, as it can be seen in Table 4, with four recursion steps up to 332800 pixels can be transported, split into 256 subimages with 1300 points each. With the exponential growth of the number of subimages with every additional recursion step, more recursions steps should be avoided and the resolution of the grid map should be down sampled instead.

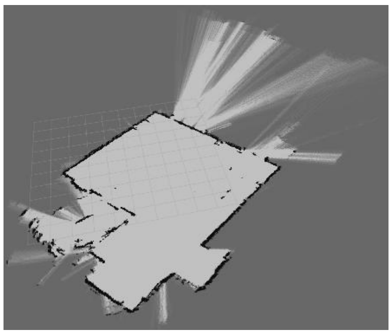


Figure 4: Map at original resolution

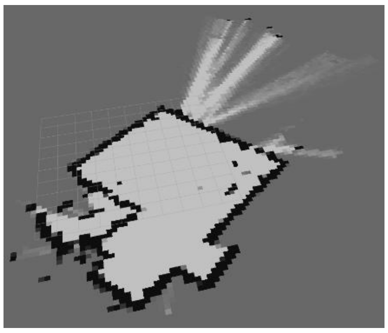


Figure 5: Map at reduced resolution

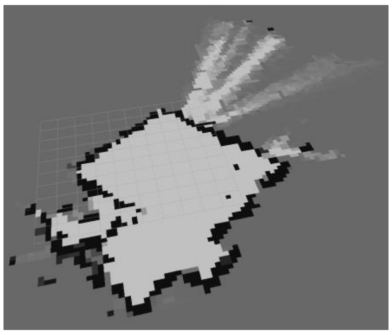
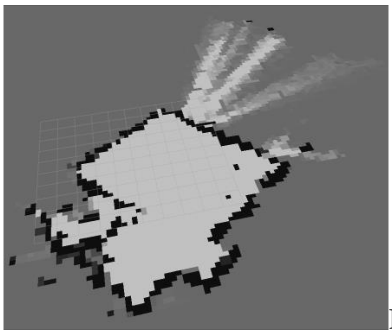


Figure 6: Map with 50% data loss

When down sampling the map, care should be taken to not loose information of occupied areas in the map, because these are of special importance for the navigation. Most interpolation methods, like bilinear interpolation, compute an average from multiple pixels from the original image when downsizing it and therefore blur the occupied areas. Contrary, for downsizing the map, the pixel with the highest occupancy value in an area is used to approximate it by a single pixel. This way, information about occupied areas is preserved. An example for a down sampled map with data loss can be seen in Figure 4 to  Figure 6.

6 Point Cloud Transportation

Point cloud messages are also very large messages that have to be split up into several packets to send them over Wi-Fi. In the case of the used laser scanner with 23.000 points or 0.7 MB per Scan, this means that the point cloud is split in more than 478 packets for transportation and if one packet is lost, the whole cloud message is lost again.

6.1 3D Scan

To reduce the amount of data in a first step, the point cloud can be down sampled to reduce the total number of points. This can be done using a voxel grid filter, which divides the point cloud into 3D boxes of a defined size. For each of these boxes the centroid of the contained points is computed and used to approximate them. By changing the size of the voxels, the number of points in the filtered point cloud can be controlled. For example, by using a box length of 0.1m, the number of points is reduced to approximately 5500 points.

Analog to the image transportation, the filtered point cloud also has to be split up in a way, that each packet is independent of the others and the cloud can still be visualized, even if some packets are lost. A simple way to do this is to split up the point cloud into several smaller clouds and reassemble the cloud at the receiver side. The data of the point cloud consists of the x, y and z coordinate, each saved as 4 Byte value. Additionally, each point cloud has information like the coordinate frame and the time step. To leave enough room for these data, a point cloud should not contain more than 80 points to fit into one packet. Therefore, a typical point cloud with up to 6000 points is split up into dozens of smaller point clouds.

On the receiver side the small point clouds can be reassembled and visualized due to their time stamps. Point clouds with the same time stamp are merged, which gives a detailed 3D view of the robot's environment. When a point cloud with a new time stamp is received, the visualized point cloud is reset.

6.2 Virtual 2D Scan

Another way to reduce the amount of data even further is the use of virtual 2D scans, which are introduced in [2]. The raw 3D point cloud contains much redundant and unnecessary information. Virtual 2D scans are a way to remove these redundant points and compute a 2D laser scan for special purposes such as localization or navigation.

Virtual 2D scans are calculated in two steps. First the amount of points in the point cloud S are reduced by a function f, that depends on the special purpose of the virtual 2D scan

$$f: S \to S_r, \qquad having\ S_r \subset S,$$

where the number of points in S_r is much less than S. In the second step the virtual 2D scan V_S is calculated from the reduced point set S_r:

$$g : S_r \to V_S$$

with

$$g : (x, y, z) \to (x, y), \ \ x, y, z \in \mathbb{R},$$

reducing the 3D coordinates x, y and z to a point on the 2D plane. This way a 3D point cloud can be reduced to a 2D scan while trying to keep the most relevant information of the scan.

There are two kinds of virtual 2D scans presented in [2] for different applications. The landmark scan is mainly used for robot localization and the obstacle scan is mainly used for navigation. Both scans reduce the point cloud to one scan point per column. To do this, at first the ground plane is removed from the point cloud.

Afterwards, for the landmark scan only points that are classified as an object point and that belong to a planar surface are selected. If there are more than two objects, the most distant one is selected, as it is more likely to detect this object in the next scan again.

For the obstacle scan, all points above the ground plane and below a defined height greater than the robot are selected. Here the closest point to the robot per column is selected to detect all objects the robot could collide with.

Both these scans reduce the amount of data, while trying to contain as many information for the operator as possible. With an angular resolution of 0.5° per scan point the resulting virtual 2D scan can contain up to 720 points per scan.

Because a laser scan assumes constant angle increments only the minimum and maximum angle, the angle increment and the distance values have to be transmitted. For this reason, the size of a whole laser scan message with 720 points is about 2908 Byte and can be send in two UDP packets.

To also make this robust against packet loss, the laser scan can be split up into two separate packets. The first packet contains the points with even indexes and the second packet contains the points with odd indexes, starting from 0. This way, information from every part of the scan be recovered, even if one packet is lost.

7 Experiment

An experiment is conducted to test the influence of network latency and packet loss on the network performance. Afterwards the methods introduced in the last two sections are tested under challenging network conditions.

7.1 TCP and UDP bandwidth evaluation

To test the performance of TCP und UDP a small network was setup, consisting of two computers and a router. One computer was connected to the router by cable, the other via Wi-Fi. To simulate a bad Wi-Fi connection, the kernel component netem is used, which can simulate additional network latency and packet loss. The tool iperf is then used to measure the bandwidth of the connection over a ten second interval.

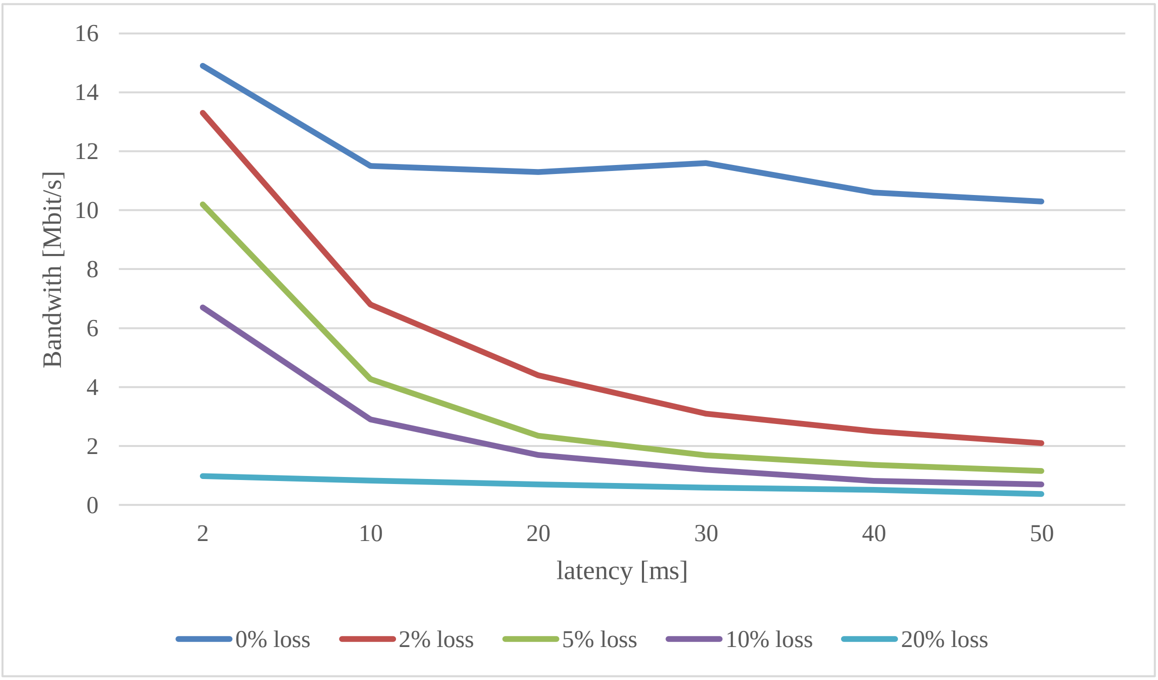


Figure 7: TCP bandwidth

The measured TCP bandwidth for different delays and amounts of packet loss can be seen in Figure 7. When there is no packet loss, a high bandwidth can be achieved even with large latencies up to 50ms. However, with an increased rate of packet loss, the bandwidth decreases rapidly, especially at higher latencies.

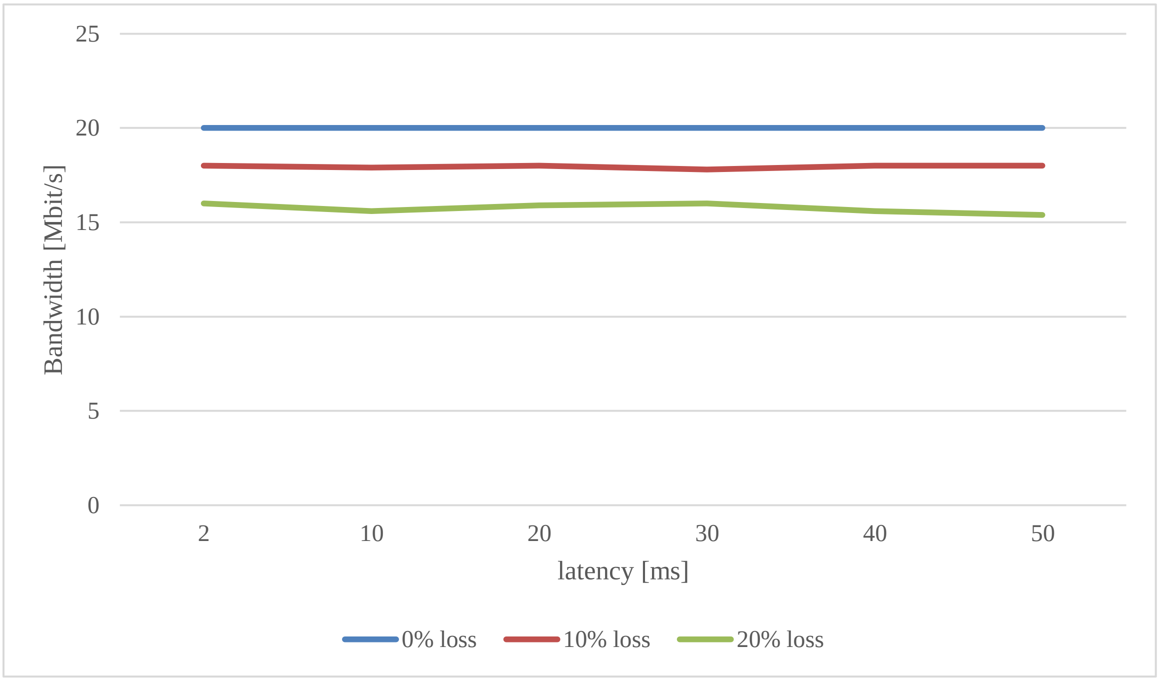


Figure 8: UDP bandwidth

The same test was done for a UDP connection. Here the rate at which the sender transmits the data was set to 20 Mbit/s. As it can be seen in Figure 8, the received bandwidth is only dependent on the set loss. For 0% loss the received bandwidth is 20 Mbit/s, for 10 % loss it is 18 Mbit/s and for 20% loss it is approximately 16 Mbit/s and therefore several orders higher than for a TCP connection affected by large latencies and packet loss.

7.2 Image transfer evaluation

To test the data transfer capabilities of the introduced methods with ROS over a wireless network an experiment using the same hardware as above is conducted. A recorded bag file is used to play back the sensor data of the thermal camera and the LIDAR. The data is again transmitted over a Wi-Fi network and the latency and packet loss was simulated using netem to get reproducible results.

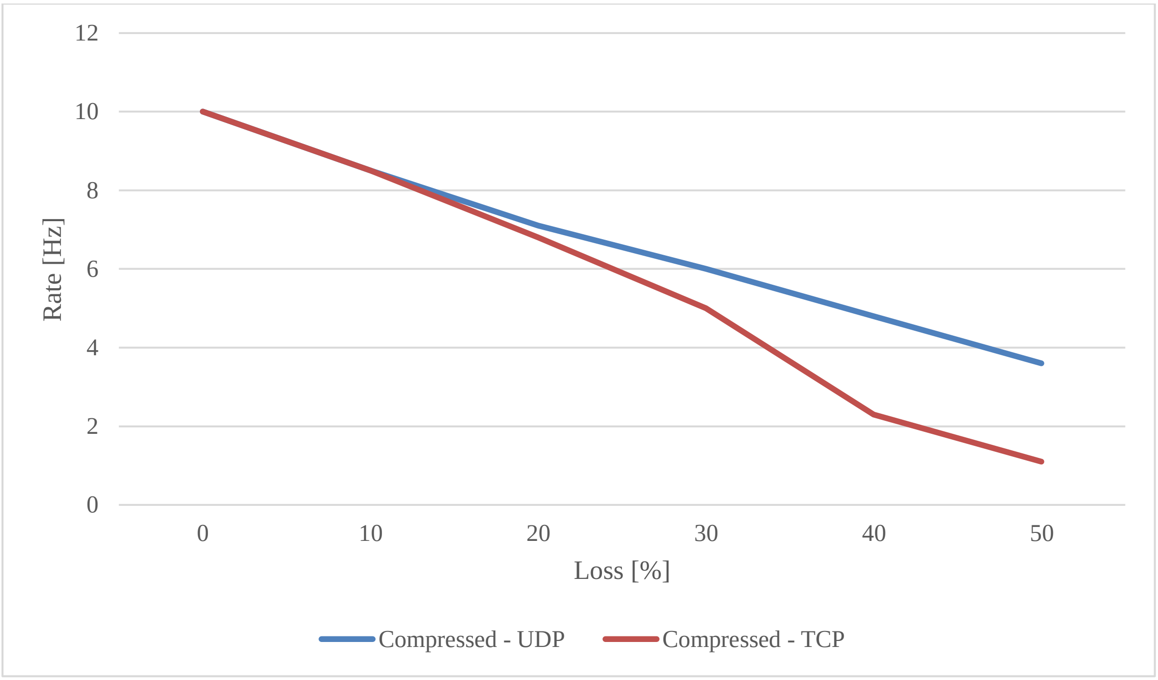


Figure 9: Ros Image Transport Rate

In Figure 9 the mean number of received frames per second is visualized using the standard compressed image transport method of ROS over a period of one minute. It can be seen for both UDP and TCP transport that the rate of received images drops, because some data packets are lost. For high losses, the TCP protocol performs worse than UDP. This is because of the retransmission mechanism in the TCP protocol. Also the time between two transferred messages becomes more unsteady. Sometimes the image is updated after a few milliseconds, sometimes it takes several seconds to update the image. Another disadvantage of the compressed image transfer is the reduced resolution. However, trying to transmit the uncompressed image over UDP, even with only 2% loss, fails, because of the large data size of one image. For TCP, with low latency it fails at about 20%.

Contrary, when using the method introduced in section 5, the rate of the received images stays constant at 10 Hz. Only the mean number of received packets decline linear with the packet loss. In the described scenario, an image is split into 64 segments. Even with a packet loss of 90%, on average six packets are transmitted successfully and can be visualized on the receiver side and when the connection is better the transmitted image automatically gets more detailed.

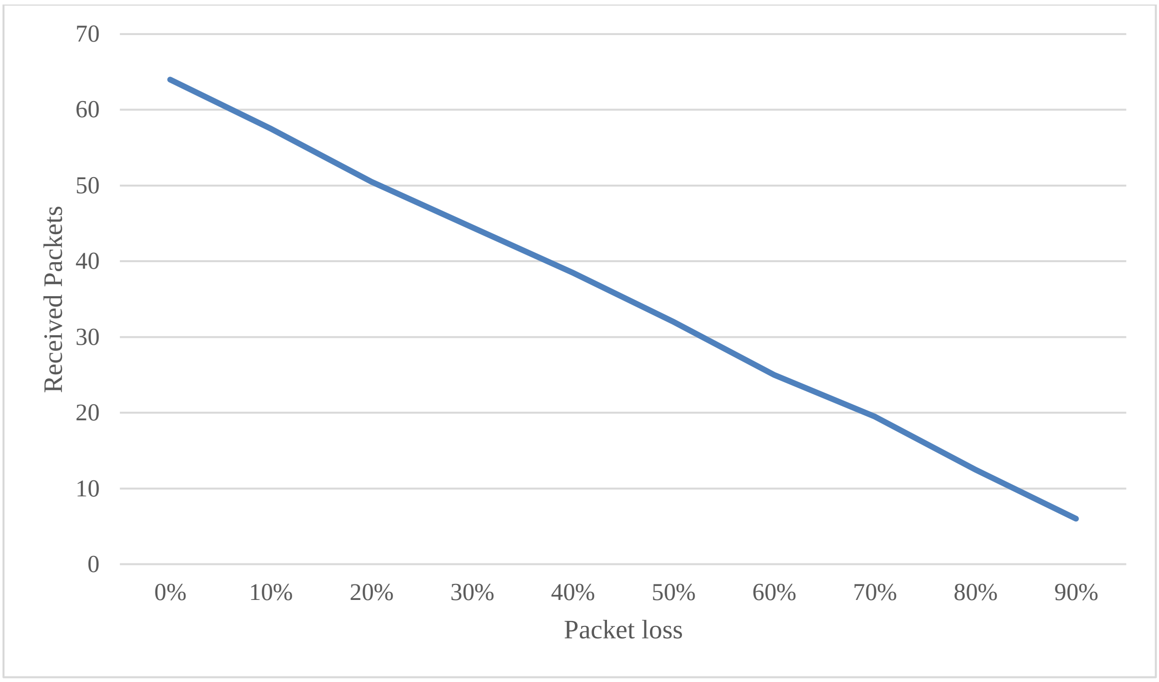


Figure 10: Received Packets

7.3 Point cloud transfer evaluation

Contrary to images, there are no point cloud compression methods implemented in ROS. Therefore, the point cloud is split up in many packets and even a very small percentage of packet loss make the transportation of the full point cloud over UDP unsuitable. In the test scenario, even with a packet loss under 3%, almost no point cloud can be successfully transmitted. Using TCP, the transmission of the point cloud is more stable, but the rate drops to 5 Hz because of the needed retransmissions at 10% loss. At a loss around 20% almost all point clouds are lost. In the tested scenario the latency of 2ms was very small. In a network with higher latencies, the performance would be much worse.

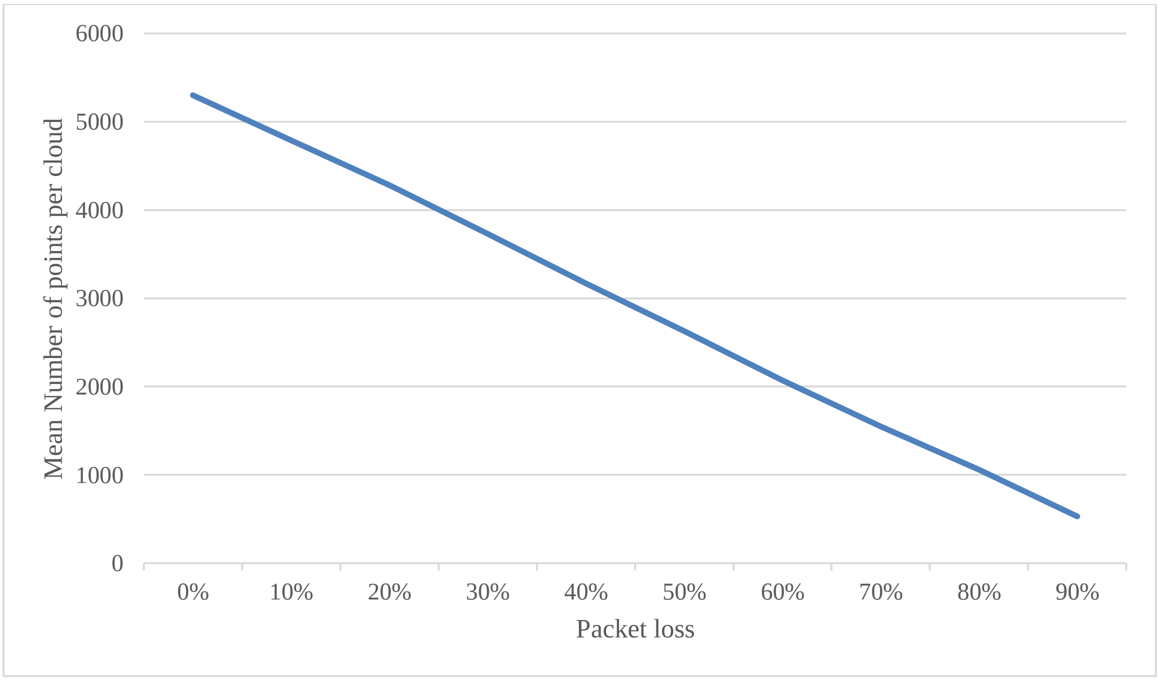


Figure 11: Number of points per received cloud.

In Figure 11 the number of points per received cloud is visualized using the method described in section 6.1. The full cloud consists of ca 5500 points and is visualized in Figure 12. As it can be seen in Figure 11, the number of points decreases linearly with the percentage of packet loss and still works for very high data loss rates. In Figure 13 and Figure 14 two examples of the same scene are shown, where only 50% and 10% of the points are transmitted. It can be seen, that when not all points are transmitted, the basic structure of the point cloud is still visible.

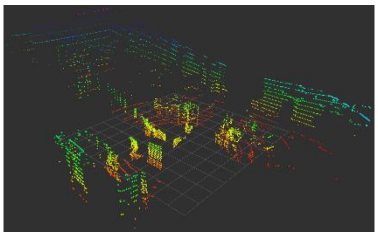


Figure 12: Fully transmitted point cloud

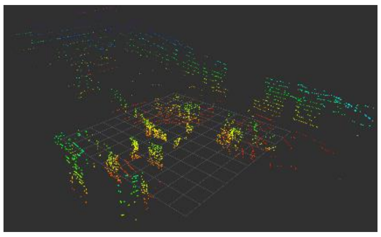


Figure 13: 50% transmitted point cloud

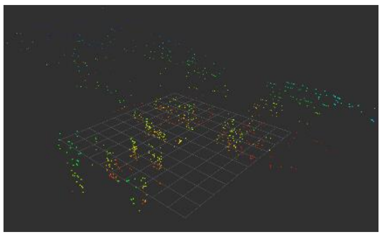


Figure 14: 10% transmitted point cloud

8 Conclusion

In this document, it was shown that data transfer over TCP is unsuitable in situations, where the latency is high and network packets might get lost. This is especially the case in situations, where Wi-Fi is used with several repeaters to increase operation range of a tele operated robot. Contrary it is shown, that UDP is not as much affected by large latencies and packet loss. However, transmitting large messages such as images or point clouds using UDP does not work well with the used Robot Operating System, because a single lost packet makes the messages unreadable for the receiver. Therefore, different mechanisms were introduced to split up a large message into several smaller ones, which are then reassembled at the receiver side. This way, parts of the transmitted sensor data can still be visualized to the operator, even when there is some packet loss, as shown in the experimental results.

A References

[1] M. Langerwisch, M. Reimer, M. Hentschel und B. Wagner, „Control of a Semi-Autonomous UGV using Lossy Low-Bandwidth Communication,“ in *2nd IFAC Symposium on Telematics Applications*, Timisoara, Romania, 2010.

[2] O. Wulf, Virtuelle 2D-Scans - Ein Verfahren zur echtzeitfähigen Roboternavigation mit dreidimensionalen Umgebungsdaten, Hannover, Germany, 2008.