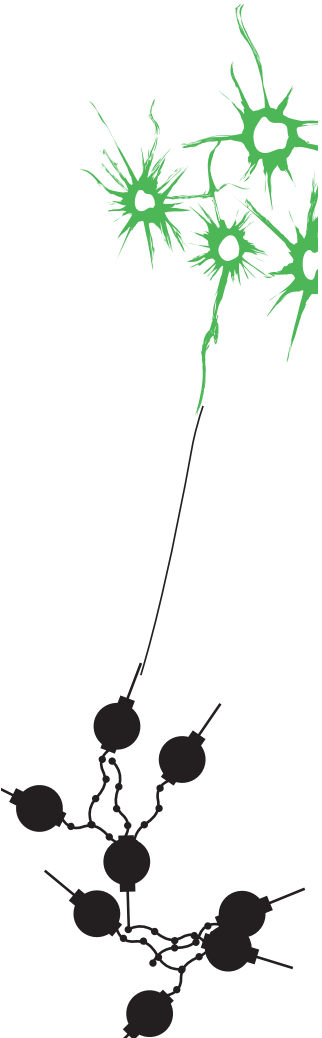


RAM ROBOTICS AND MECHATRONICS

Optimizing wireless video streams for computer vision

F.J. (Frank) van der Hoek

MSC ASSIGNMENT

Committee:
dr.ir. J.F. Broenink
K.H. Russcher, MSc
dr. M. Poel

August, 2019

037RaM2019
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

The Dutch National police increasingly use robots for their operations, for example during observation and surveillance. The robots are equipped with a camera and transmit video data via a wireless video stream to the tele-operater, who uses the video for navigation. The tele-operator can be assisted, or replaced, by algorithms that use computer vision.

However, the video data from the robots cannot be completely transmitted when the bit rate of wireless video streams is larger than the available throughput. This occurs, for example, when the wireless channel switches to a robust coding and modulation scheme, due to external disturbances. The incomplete data causes visible artefacts in the decoded video and computer vision algorithms cannot be effectively applied to such videos.

The goal of this research is to determine how video streams can be optimized for computer vision, when the throughput is limited. The research is focussed on three types of video scaling that reduce data: spatial, temporal, and quality scaling. For these types of scaling, two questions are answered during the research: Can the required throughput of wireless video streams be reduced enough using spatial, temporal, and quality scaling, such that video data can be transferred completely? And how do spatial, temporal, and quality scaling affect computer vision?

The impact of the three types of scaling on required throughput and computer vision, has been determined by analysing bit rate and visual tracking performance for videos generated from the RGB-D and CoRBS datasets, after applying different spatial, temporal, and quality scaling parameters. A custom visual tracking algorithm has been designed for the performance evaluation, based on direct visual simultaneous localization and mapping methods. It uses basic image processing techniques that are used in most other computer vision algorithms, such that the results of the research are generalizable to such algorithms.

The results indicate that combining the three types of scaling reduces the required throughput of a video enough, such that it is below the minimum available throughput of the IEEE 802.11 wifi standards. Of the three types, quality scaling did not impact tracking performance. Spatial scaling had a negative impact on tracking performance, but it also reduced the throughput. Temporal scaling had a bigger impact on tracking performance than spatial scaling, but a smaller impact on the required throughput.

Based on the results, an optimal scaling strategy has been determined, that reduces throughput, while maximizing performance of computer vision algorithms. The optimal strategy is to first apply quality scaling on a video stream, until the lowest quality is reached, followed by spatial scaling, until the lowest resolution is reached, and finally temporal scaling to further reduce the required throughput.

The results can be combined with related research to implement optimal wireless video streams on robots, such that computer vision algorithms can be effectively applied. Further research, on a larger number of videos, is required to determine the optimal scaling strategy for a specific throughput and to verify the optimal strategy in practice on a robot with a wireless video stream.

Preface

In front of you is the thesis “Optimizing wireless video streams for computer vision.” It marks the end of my decade of studying Electrical Engineering at the University of Twente, a university that encourages the entrepreneurial spirit of its students. I worked part-time on the research and writing of this thesis, from the beginning of 2018 to half-way into 2019, while running my own business at the same time. I am grateful to have had the chance to combine both endeavours.

To me, the topic of this thesis, computer vision, has some magic to it. To enable a computer algorithm to “see”, and act upon this vision, is both exciting and challenging. During my research, it was especially challenging to present the results in a meaningful and understandable format. Discussing the results with several people from the Robotics and Mechatronics group, allowed me to see the results from a different perspective and to provide clear answers to the identified questions.

I am thrilled to finally finish my study and my thanks go out to everyone who supported me, and helped me shape my thesis. In particular, I would like to thank ir. K.H. Russcher, my daily supervisor, who supervised me for more than a year and provided insightful feedback on a weekly basis. I also wish to thank dr.ir. J.F. Broenink for his constructive feedback on my thesis, both at the start and at the end of my research and for the suggestion to add more figures and lists. Furthermore, I would like to thank prof.dr.ir. G.J.M. Krijnen, whose feedback greatly helped me turning my research into a meaningful thesis.

To my friends and family: thank you for keeping me motivated. My girlfriend deserves a special note of thanks: without your wise words and support I think I would not have had the perseverance, strength and urgency to finish my thesis.

Frank van der Hoek
Utrecht, 22nd August, 2019

Contents

1 Introduction

1.1 Context

Recently, the Dutch National Police (NPN) have started to use robots for their operations. The NPN use robots for a variety of tasks, such as surveillance and observation. Depending on the task, the NPN may use drones, wheeled robots or other robots. The robots are able to travel to places where it would be dangerous to deploy human personnel, and the robots, especially drones, can travel much faster to an area of interest than a person. Therefore, robots allow the NPN to increase their efficacy and the safety of their employees.

The robots are tele-operated and equipped with a camera. The video data from the cameras is transmitted to the tele-operator via a wireless video stream. The wireless connection allows the NPN to quickly and effectively deploy robots in a variety of environments, where time is sometimes of the essence. The videos are encoded using the widely used H.264 encoder, which is implemented in hardware on most robots for fast and efficient encoding.

In the future, the video stream will be used to assist, or replace, the tele-operator by algorithms that use computer vision. Examples of these are visual simultaneous localization and mapping (SLAM) systems that aid the tele-operator during navigation, and algorithms for dense 3D reconstructions of observed scenes, such as a crime scene. In such systems, streams from multiple robots and body cams can be combined on a centralized system.

The NPN do not design or manufacture the robots themselves, but use commercially available robots, from various manufacturers. Hence, changes to these robotic systems are limited and the NPN rely on the design decisions of the manufacturers.

1.2 Problem

Video data from the cameras cannot be completely transmitted when the required throughput is larger than the throughput available on the wireless channel. This is, for example, the case when the wireless channel switches to a robust coding and modulation scheme, due to external disturbances. It also occurs if multiple videos are streamed of the network, such as when multiple robots perform cooperative SLAM.

When the data is not completely transmitted, missing data results in visible artefacts in the decoded video. The artefacts make it difficult for a tele-operator to navigate the robot and inhibit effective use of computer vision on the video.

1.3 Focus

Several solutions to the problem can be thought of, for example:

1. Replacing the wireless connection with a wired connection, which has a higher throughput than a wireless connection.
2. Preventing the wireless channel from switching to coding and modulation schemes with low bit rates. This can be accomplished by increasing the signal-to-noise ratio of the channel using better antennas or signal amplification.
3. Applying the computer vision directly to the video on the robot itself.
4. Reducing the data by discarding part of the data using lossy compression.

Not all these solutions are feasible, given the situation of the NPN. The first option prevents the NPN from using robots to travel large distances unless the tele-operator closely follows the robot. The solution, therefore, takes away the advantages of increased flexibility, speed and

safety that the robots are able to provide. Furthermore, a wire imposes other challenges as it may get stuck and is too heavy to carry for some robots, such as small drones.

The second option is not feasible as well. As explained in Section 1.1, the police relies on the design decisions of manufacturers and cannot easily change parts of the robots. Furthermore, it does not solve the problem if the throughput per video stream is reduced when multiple videos are streamed over the network.

Similar to the second option, option 3 is not possible, because is not realistic to have all manufacturers change the software on the robots. Another disadvantage of the option is that it requires new software implementations for every robot that is, or will be, used by NPN.

Hence, in this thesis the focus is on the fourth option: reducing the required throughput of the video stream, by discarding part of the data using lossy compression.

More specifically, the data reductions that will be considered, must be possible using minor changes to the configuration of the H.264 encoder. The NPN should be able to prescribe these minor changes to the manufacturers of robots and the changes should be easy to implement for the manufacturers, such that it is realistic that manufacturers implement the changes.

1.4 Related work

It is the task of an encoder to reduce data from a video, such that the video file becomes small enough for efficient storage or transmission over a network connection. An encoder uses a variety of techniques to describe the video information using less data, i.e., to compress data. During compression, the encoder is responsible for discarding information with the least visual value first. The visual value, however, might be different for computer vision than for human vision.

1.4.1 Video encoding

H.264 (Wiegand et al., 2003; Ostermann et al., 2004) is the most widely used video compression standard. Amongst others, the standard uses inter and intra frame prediction and motion estimation to only encode shifts of blocks of image data. This greatly reduces the amount of information that needs to be transferred.

Certain implementations of the H.264 encoder, such as the open source x264 encoder, allow setting a constant rate factor (CRF) (Robitza, 2017a). Using this setting the encoder will apply a constant quality factor to the video. This quality is the *perceived* quality, which means that it will apply different quantization parameters for the compression of each frame, depending on the content. One way in which the encoder optimizes the compression, is by taking motion into account. High motion frames are compressed more than frames with little motion. The resulting video will have a high rate-distortion (RD) performance (Merritt and Vanam, 2007), which is measured as the peak signal-to-noise ratio (PSNR) as a function of average bit rate.

An extension to the H.264 standard was introduced in 2007 (Segall and Sullivan, 2007; Schwarz et al., 2007) to improve support for multiple display resolutions using scalable video coding (SVC). SVC encodes scaled versions of the video in subsets of the bit stream. These subsets can be derived by dropping packets from the main bit stream. The scaled video data that is contained in the subset can be either scaled by resolution (spatial scalability), frame rate (temporal scalability), quality (quality scalability) or a combination of these three. Both server and client can switch to a different configuration by dropping packets. Hence, SVC enables a reduction of the required throughput without re-encoding.

It has been shown that SVC can be used to improve quality (Schierl et al., 2007) and bandwidth utilization (Chiang et al., 2008). Combining spatial, temporal, and quality scaling can effectively improve the RD performance (Van der Auwera et al., 2008). When different priorities are

assigned to the packets from different subsets, The quality of the video can be optimized by assigning different priorities to packets from different subsets (Monteiro et al., 2008).

Hence, it is expected that SVC, or more in general, spatial, temporal, and quality scaling, can be used to reduce the required throughput of a video stream, such that reliable transmission is possible over a wireless network connection. However, SVC is not generally supported by H.264 encoders.

1.4.2 Perceived image and video quality

The H.264 encoder can be configured to optimize video encoding for the perceived quality of service (PQOS) for humans. Several researches have been conducted to determine how humans perceive quality of service. Mannos and Sakrison (1974) showed how pseudorandom perturbations in the intensity pattern of a given meaningful image is detectable by a human subject. According to Mannos and Sakrison (1974) humans are more sensitive to some spatial frequencies than other spatial frequencies, and more sensitive to errors in grey areas than in white.

Similar reasoning led several others to the conclusion that a simple error metric, such as the mean squared error (MSE) is not suitable as a quality metric for video encoding (Teo and Heeger, 1994; Eckert and Bradley, 1998; Winkler, 1999; Wang, 2001; Wang and Bovik, 2002; Wang et al., 2002; Pinson and Wolf, 2004). Some suggested different metrics (Teo and Heeger, 1994; Winkler, 1999; Wang, 2001; Wang et al., 2002; Wang and Bovik, 2002; Wang et al., 2004) to objectively describe quality as perceived by the human visual system. An overview and assessment of several systems is given in (Chikkerur et al., 2011). Overall, human perception makes objective image quality assessment a difficult task (Wang et al., 2002).

Network related effects such as jitter and delays do not only affect the perceived quality (Claypool and Tanner, 1999), but also the understanding of video (Ghinea and Thomas, 1998). Additionally, loss of packets results in a lower perceived quality of service and is considered a useful metric in analysing the quality of a video (Lin et al., 2006; Rui et al., 2006; Frnda et al., 2016). Gardikis et al. (2012) showed the limited correlation between network-level quality of service (NQOS) and PQOS.

Hence, it is difficult to express quality of service from the perspective of a human, because the human visual system is highly subjective when perceiving quality. How does this compare to computer vision?

1.4.3 Visual SLAM

An important and extensively researched computer vision topic is visual SLAM. Eade and Drummond (2006) and Davison et al. (2007) were the first to present a successful application of a pure vision-based SLAM method for a monocular camera. Eade and Drummond (2006) used a particle filter and Davison et al. (2007) an extended Kalman filter (EKF) for the camera pose combined with a particle filter for the depth of each feature.

Mouragnon et al. (2006) and later Klein and Murray (2009) showed how bundle adjustment can be used for camera pose estimation and geometrical reconstruction.

As opposed to previous work, Klein and Murray (2009) perform tracking and mapping on separate threads so that it can run on low-end devices. Building on this work, Mur-Artal et al. (2015) proposed ORB-SLAM, which uses ORB features and performs loop closing and other optimizations.

All these approaches estimate 3D geometry based on matches of keypoints. The reprojection error for matched keypoints is minimized to obtain 3D geometry information. As they do not directly operate on the image intensity, these types of methods are referred to as *indirect* meth-

ods. Besides being indirect, the resulting map for these methods is sparse and prior knowledge about the reconstruction is not used during estimation.

Other methods, referred to as *direct* methods, work directly on the pixel intensity. Such methods minimize the difference in pixel intensity between frames, the photometric error. As these methods do not require feature extraction, but operate directly on pixel intensity, they can generate denser maps using less computation. Furthermore, dense reconstruction allows for the use of a regularization filter to optimize depth estimates by smoothing the generated reconstruction. Examples of direct methods are DTAM (Newcombe et al., 2011), LSD-SLAM (Engel et al., 2014) and DSO (Engel et al., 2017).

In summary, visual SLAM is based either on matching features, or directly comparing pixel intensities. As opposed to the human visual system, computer vision is, at least for visual SLAM, not more sensitive to specific spatial frequencies or pixel intensities than others.

Hence, it is expected that computer vision algorithms experience a different perceived quality of service than humans, and that the techniques that encoders apply to optimize compression for humans do not optimize encoding for computer vision. Research is missing regarding the perceived quality of service from the perspective of computer vision algorithms.

1.5 Research questions

To solve the problem for the NPN, the data from the video stream of the robots must be reduced without hindering computer vision tasks. Therefore, the goal of this thesis is to determine how wireless video streams can be optimized for computer vision, when the throughput of the wireless channel is limited.

Building on the related work that was presented in the previous section, it is analysed how spatial, temporal, and quality are able to reduce video data and how these types of scaling affect the perceived quality of service of computer vision algorithms. More specifically, the main research question of this thesis is:

How can wireless video streams be optimized for computer vision, when the throughput is limited?

The optimization consists of a trade-off between the data reduction and performance of computer vision algorithms. Hence, the research is subdivided into two parts. First, it is determined whether the required throughput of a video stream can be sufficiently reduced to guarantee successful transmission, even when the available throughput of the wireless connection becomes low. Second, the impact of such data reduction measures on a visual algorithm are examined. Therefore, the main research question is subdivided into two sub questions:

1. *Can the required throughput of video streams be reduced using spatial, temporal, and quality scaling, such that videos can be streamed reliably over a wireless connection?*
2. *How do spatial, temporal, and quality scaling affect computer vision algorithms?*

The sub questions are answered by evaluating the bit rate and performance of a visual tracking algorithm for videos similar to scenarios that robots from the NPN encounter, after applying the three types of scaling using different parameters. Generalizability of the results is ensured by restricting the visual tracking algorithm to basic image processing techniques that are used in most computer vision algorithms.

1.6 Outline

The outline of this thesis is as follows: In Chapter 2, a theoretical background regarding visual tracking and encoding is provided. First, basic camera projection using the pinhole camera model is explained. Next, it is explained how pixel depth can be estimated using tracked points,

based on epipolar geometry. Finally a brief overview of the H.264 video encoding standard is provided.

In Chapter 3, it is explained how the limited wireless connection poses challenges to a wireless video stream. After this, it is analysed how the required throughput can be reduced using spatial, temporal, and quality scaling. Finally, the impact of the different types of scaling is analysed. The analyses in Chapter 3 are qualitative, as quantitative analysis is not possible, because the impact of scaling depends on the content of a video. It is concluded that experiments are needed for a quantitative analysis.

In Chapter 4, it is explained how videos are generated from two datasets using different scaling parameters, and how the bit rate and visual tracking performance for these videos is evaluated using experiments.

The results of these experiments are presented and discussed in Chapter 5. It is shown how spatial, temporal, and quality scaling affect the required throughput of a video stream and the PQOS of a visual tracking algorithm. These results are subsequently used to determine a strategy for optimizing a wireless video stream for a visual tracking algorithm and it is explained how these results apply to computer vision algorithms in general.

In the final chapter, Chapter 6, the work is concluded and topics for further research are recommended.

2 Background

In this chapter a theoretical background regarding visual tracking and H.264 encoding is provided. First, the pinhole camera model is described, which forms the basis for capturing the three dimensional world on a two dimensional image plane. Next, the relationship between a point in one video frame and its projection in another video frame is described using the concept of epipolar geometry. After this, a method to select points to track throughout a video is discussed, based on gradient of pixel intensities. Subsequently, it is discussed how a pixel can be matched between video frames by minimizing the sum of squared differences (SSD) of the photometric error. In Section 2.5 it is described how epipolar geometry and photometric error minimization are used in a visual simultaneous localization and mapping (SLAM) method to build a map of the environment. Finally, a brief introduction to the H.264 encoder is given, such that the impact of video compression can be understood as well as the ways in which the trade-off between bit rate and video quality can be controlled using different rate control factors.

2.1 Camera projection using the pinhole camera model

The pinhole camera model is a widely used model that mathematically describes the relationship between a point in 3D and its projection on a 2D image plane. It is depicted in Figure2.1.

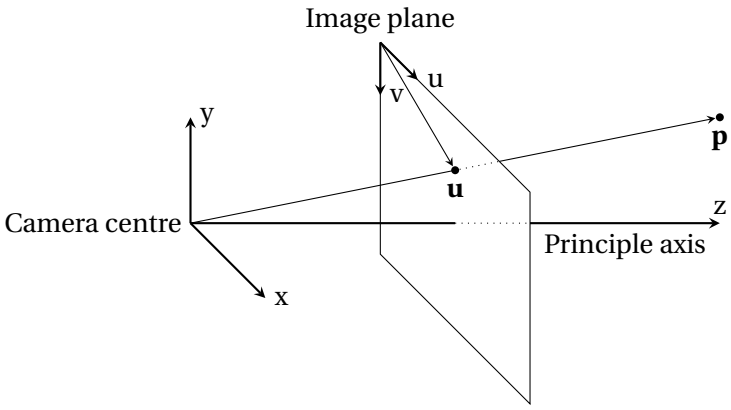


Figure 2.1: The pinhole camera model. A point **p** in 3D is projected as a pixel at location **u** in the image plane.

For a point $\mathbf{p} \in \mathbb{R}^3$ the pinhole camera model is described by:

$$\lambda \mathbf{u} = \mathbf{KRp} + \mathbf{Kt} \tag{2.1}$$

Where

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.2}$$

describes the 2D pixel location $\begin{bmatrix} u & v \end{bmatrix}^T$ in homogeneous coordinates,

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

contains the camera parameters with focal lengths f_x, f_y and principle axis location $\begin{bmatrix} c_x & c_y \end{bmatrix}^T$ and $\mathbf{R}$ and $\mathbf{t}$ the rotation matrix and translation vector that map the world reference frame coordinates to coordinates with respect to the camera reference frame. λ is a scaling factor that scales the homogeneous coordinates such that the bottom value in $\mathbf{u}$ is equal to 1.

2.2 Epipolar geometry

In Equation 2.1 the pose of the camera is described by $\mathbf{R}$ and $\mathbf{t}$. When the pose of a camera changes, $\mathbf{R}$ and $\mathbf{t}$ will change along. If a point $\mathbf{u}_i$ describes the pixel of point $\mathbf{p}$ in frame i with corresponding rotation $\mathbf{R}_i$ and translation $\mathbf{t}_i$, the projection of $\mathbf{p}$ in frame j is given by

$$\lambda_j \mathbf{u}_j = \mathbf{K}\mathbf{R}_j \mathbf{p} + \mathbf{K}\mathbf{t}_j \tag{2.4}$$

By expressing $\mathbf{p}$ in terms of $\mathbf{u}_i, \mathbf{R}_i, \mathbf{t}_i$ and λ_i using Equation 2.1, Equation 2.4 can be expressed as:

$$\lambda_j \mathbf{u}_j = \mathbf{K}\mathbf{R}_j (\lambda_i \mathbf{R}_i^T \mathbf{K}^{-1} \mathbf{u}_i - \mathbf{R}_i^T \mathbf{t}_i) + \mathbf{K}\mathbf{t}_j \tag{2.5}$$

Which can be rewritten to

$$\lambda_j \mathbf{u}_j = \lambda_i \mathbf{K}\mathbf{R}_j \mathbf{R}_i^T \mathbf{K}^{-1} \mathbf{u}_i + \mathbf{K}(\mathbf{t}_j - \mathbf{R}_j \mathbf{R}_i^T \mathbf{t}_i) \tag{2.6}$$

If the rotation and translation of the camera at frame i and j are known, the projection $\mathbf{u}_j$ is described by a line that depends on the depth λ_i.

Expressing Equation 2.6 in the coordinate frame of the camera in video frame i, i.e., $\mathbf{R}_i = \mathbf{I}_3$ and $\mathbf{t}_i = \mathbf{0}$, results in the much simpler equation:

$$\lambda_j \mathbf{u}_j = \lambda_i \mathbf{K}\mathbf{R}_j \mathbf{K}^{-1} \mathbf{u}_i + \mathbf{K}\mathbf{t}_j \tag{2.7}$$

Where λ is equal to the depth z of the point.

The line that is described by Equation 2.7 is referred to as the epipolar line. The epipolar line is depicted in Figure 2.2 as l. In the figure, several possible 3D points, corresponding to pixel $\mathbf{u}_i$ are shown. The pinhole camera model of Figure 2.1 is shown for the camera centre C_1 in frame 1 and the camera centre C_2 in frame 2.

2.3 Matching by minimizing the photometric error

The epipolar line described by Equation 2.7 has to be reduced to a point such that the depth given by λ_i can be estimated. O common approach for finding the best matching pixel on the epipolar line, is minimization of the photometric error.

The photometric error between a pixel $\begin{bmatrix} u_i & v_i \end{bmatrix}^T$ in frame i and another pixel $\begin{bmatrix} u_j & v_j \end{bmatrix}^T$ in frame j is defined by:

$$E = I_i(u_i, v_i) - I_j(u_j, v_j) \tag{2.8}$$

Using a quadratic cost function and a patch N around a pixel, instead of a single pixel, Equation 2.8 can be summed to obtain the SSD corresponding to the two pixels:

$$SSD = \sum_N \left(I_i(u_{i,n}, v_{i,n}) - I_j(u_{j,n}, v_{j,n}) \right)^2 \tag{2.9}$$

The coordinates u_j, v_j can be sampled from the epipolar line given by Equation 2.7.

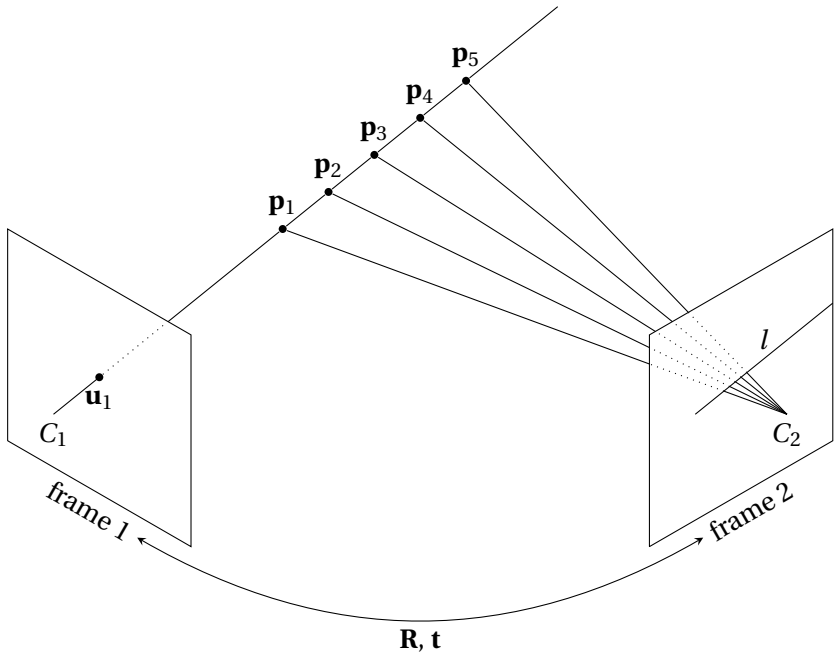


Figure 2.2: Epipolar geometry. A point in frame 1 is projected as a line l in frame 2, when the camera is rotated and or translated between the frame captures. Points $\mathbf{p}_1$ to $\mathbf{p}_5$ are 3D points that correspond to the pixel $\mathbf{u}_1$. The depth can be estimated by finding matching pixel on line l in frame 2.

For an optimal match Equation 2.9 will be minimal. Hence, the matching pixel can be found by finding the pixel on the epipolar line for which Equation 2.9 is minimal. The depth λ_i that corresponds to the minimal SSD is the depth estimate for pixel $\mathbf{u}_i$.

2.4 Gradient-based point selection

Not all pixels in a video frame can be accurately tracked. When pixels surrounding a pixel at $\mathbf{u}_i$ have similar intensities, the intensity difference from Equation 2.8 will be similar for multiple points on the epipolar line given by Equation 2.7. Equation 2.9 will hence not provide a clear minimum for the optimal match.

Engel et al. (2017) suggested to track only pixels with high-gradient values. As the gradient is proportional to local pixel differences, high-gradient points provide more distinctive minima for the SSD.

The difference between tracking low-gradient pixels and high-gradient pixels is shown in Figure 2.4. In the figure the SSD along the epipolar line in Figure 2.3b is shown for different image patches from Figure 2.3a.

In Figure 2.4a the SSD along the epipolar line is shown for an image patch with small gradient values and in Figure 2.4b the SSD along the epipolar line is shown for an image patch with larger gradient values. It can be seen that the high-gradient patch results in a clear minimum value for the SSD, whereas the low-gradient patch has multiple minimum values.

2.5 A brief introduction to visual SLAM

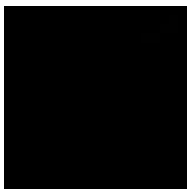
SLAM is the process during which a map of the environment is created, while simultaneously localizing the camera within this map. Besides vision-based methods, there are other methods that use lasers, sound, odometry or a combination of such techniques.

There are two different approaches regarding visual SLAM: indirect methods, that operate on features and minimize their reprojection error, and direct methods, that operate directly on

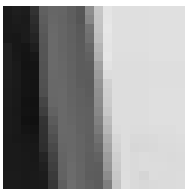
(a) Frame 1. The areas around the selected points are indicated by the two rectangles.

(b) Frame 2.

(c) The area of the image within the left rectangle of (a).



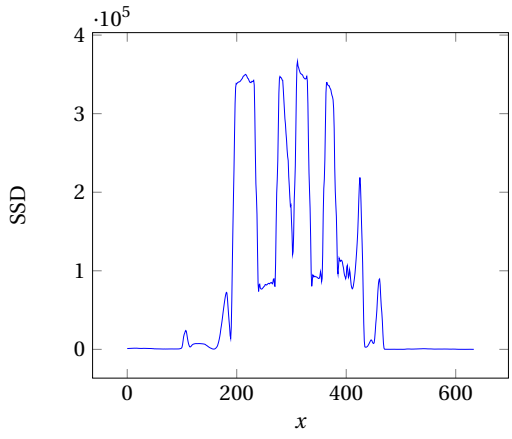
(d) The gradient of the area (c).



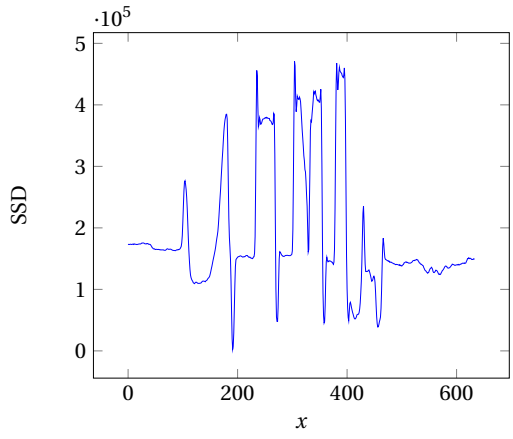
(e) The area of the image within the right rectangle of (a).

(f) The gradient of the area (e).

Figure 2.3: Two frames of video sequence. In (c) and (e) two areas of the frame of (a) are shown. In (d) it can be seen that the gradient in the left rectangular area of (a) is low. The gradient of the right rectangular area of (a) is shown in (f) and is larger around the two edges of (e). The images are part of the RGB-D dataset (Sturm et al., 2012).



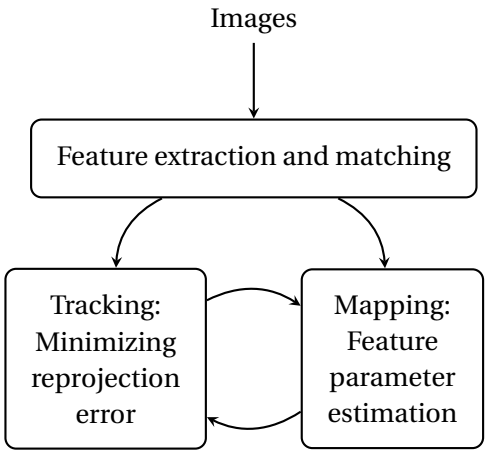
(a) The SSD along the epipolar line in Figure 2.3b for an image patch within Figure 2.3c. There is no clear minimum. Therefore, the pixel cannot be matched accurately.



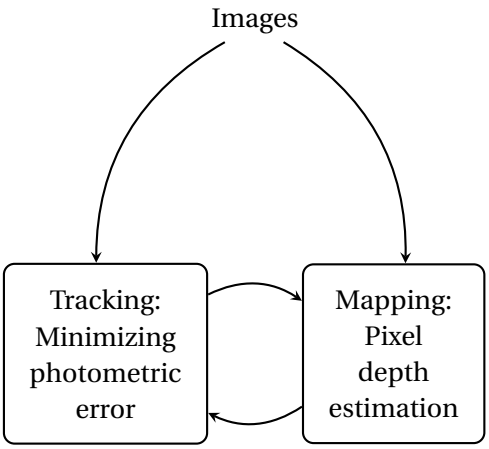
(b) The SSD along the epipolar line in Figure 2.3b for an image patch within Figure 2.3e. There is a clear absolute minimum value around $x = 190$. Therefore, the pixel can be matched accurately.

Figure 2.4: The SSD along the epipolar line in Figure 2.3b for image patches from both regions of Figure 2.3a. Only the patch from the high-gradient region can be matched accurately.

pixel intensity and minimize the photometric error. The difference between the methods is shown in Figure 2.5.



(a) In indirect SLAM features are extracted and used for tracking and mapping. The reprojection error of features is minimized during the tracking process.



(b) Direct SLAM methods operate directly on the pixel intensities of the image. The photometric error between pixel matches is minimized during the tracking process.

Figure 2.5: The difference between direct and indirect SLAM methods.

2.5.1 Indirect methods

Eade and Drummond (2006) and Davison et al. (2007) where the first to present a successful application of a pure vision-based SLAM method for a monocular camera. In these methods features are extracted from video frames and matched in subsequent frames. Based on these matches the estimated pose of the camera is updated together with the 3D locations of the features.

Eade and Drummond (2006) used a particle filter for this, where for each landmark multiple hypotheses for the inverse depth are maintained and updated using the matched features. The inverse depth here is used as the resulting likelihood is better approximated by a Gaussian distribution.

Davison et al. (2007) used an extended Kalman filter (EKF) for the camera pose combined with a particle filter for the depth of each feature, where the particles are uniformly distributed between a minimum and maximum depth.

Mouragnon et al. (2006) and later Klein and Murray (2009) showed how bundle adjustment can be used for camera pose estimation and geometrical reconstruction. Bundle adjustment optimizes the reprojection error of features over multiple frames simultaneously.

As opposed to previous work, Klein and Murray (2009) perform tracking and mapping on separate threads so that it can run on low-end devices. Multi-threading allows the bundle adjustment algorithm to run in the background. Because of this, accurate 3D reconstructions can be generated periodically, whereas the camera pose is updated every frame.

Building on this work, Mur-Artal et al. (2015) proposed ORB-SLAM, which uses ORB features and performs loop closing and other optimizations.

All these approaches estimate 3D geometry based on matches of keypoints. The reprojection error for matched keypoints is minimized to obtain 3D geometry information. As these types of methods do not directly operate on the image intensity, these methods are referred to as

indirect methods. Besides being indirect, the resulting map for these methods is sparse and prior knowledge about the reconstruction is not used during estimation.

2.5.2 Direct methods

Other methods, referred to as direct methods, work directly on the pixel intensities. Such methods minimize the difference in pixel intensity between frames. This difference is referred to as the photometric error.

As direct methods do not require feature extraction, such methods can generate denser maps using less computation. Furthermore, the dense reconstruction allows for the use of a regularization filter to optimize depth estimates by smoothing the generated reconstruction.

Examples of direct methods are DTAM (Newcombe et al., 2011), LSD-SLAM (Engel et al., 2014) and DSO (Engel et al., 2017).

The techniques used to create a map in the direct methods are similar to those discussed in Sections 2.1–2.3. When enough points are tracked, both the depth λ_i and the pose defined by $\mathbf{R}_j, \mathbf{t}_j$ in Equation 2.7 can be optimized simultaneously using for example the Gauss-Newton algorithm (Engel et al., 2017).

2.6 A brief introduction to the H.264 encoder

From the moment that videos were stored digitally on DVDs and the like, compression techniques were used to increase storage efficiency. The technology, either hardware or software based, that is responsible for the compression and decompression of raw video data, is referred to as a codec.

There is a wide variety of these video codecs available. The most widely used compressed format is H.264, also known as MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) Wiegand et al. (2003) and was developed in 2003 by the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group to enable transfer of high definition television signals.

The H.264 standard defines two layers for encoding; the network abstraction layer (NAL) and the video coding layer (VCL).

2.6.1 The network abstraction layer

The NAL is used to prepare the encoded data for distribution on a variety of data transport layers such as RTP or IP, several file formats and broadcasting services.

Encoded data is distributed via small packets of data that are referred to as NAL units. NAL units can either contain video data (VCL NAL units), or additional information (non-VCL NAL units). An example of such additional information is a parameter set, which contains information about the VCL NAL units that is expected to rarely change. Such that this information does not have to be sent with each individual VCL NAL unit.

A single picture can cover multiple NAL units. To recover from loss or data corruption, additional VCL NAL units containing redundant coded pictures can be added to the picture data.

2.6.2 The video coding layer

Where the NAL prepares the data for distribution, the VCL is responsible for the actual encoding of the raw video data.

H.264 follows the block-based hybrid video coding approach. Each picture is divided into macroblocks, which can be encoded in an efficient way.

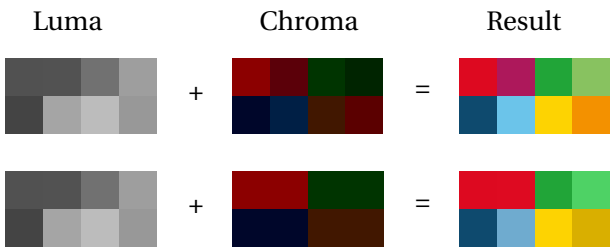


Figure 2.6: Chroma subsampling. The chroma components of the second row are subsampled. Only one chroma sample is used for every set of two consecutive pixels. The colour of the result is slightly different, but the brightness is not affected.

Chroma subsampling

Data in the macroblocks is stored in a different format than the standard Red, Green, Blue (RGB) format and is subsampled using chroma subsampling. This means that the resolution of chroma information, i.e., colour, is lowered with respect to the luma information, i.e., luminance, of a frame, by subsampling. The principle of chroma sub sampling is shown in Figure 2.6.

The reasoning behind chroma subsampling is that the human vision system is more sensitive to differences in luminance than colour. Chroma subsampling can therefore be used to decrease the file size of image information.

To make use of chroma subsampling, the pixel information must be converted from RGB format into $Y'C_BC_R$ format, where Y' is the luma component and C_B and C_R the blue-difference and red-difference chroma components respectively. For analog signals, the chroma parts are indicated by P_B and P_R and are computed using the following equations:

$$\begin{aligned} Y' &= K_R \cdot R' + K_G \cdot G' + K_B \cdot B' \\ P_B &= \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B} \\ P_R &= \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_R} \end{aligned} \tag{2.10}$$

Where $K_R + K_G + K_B = 1$ are the constants ordinally derived from the RGB colour space. For 8-bit samples, the digital values can be obtained using:

$$\begin{aligned} Y &= 16 + 219 \cdot Y' \\ C_B &= 128 + 224 \cdot P_B \\ C_R &= 128 + 224 \cdot P_R \end{aligned} \tag{2.11}$$

This results in scaled versions of the luma ranging from 16 to 235 and scaled versions of the chroma ranging from 16 to 240.

The extra room at the begin and end of the values are called the footroom and headroom respectively and are used for overshoot or undershoot of the processed signal.

Macroblock prediction using I,P and B frames

Frames can be coded using different coding types. As shown in Figure 2.7, there are I, P and B type frames. The samples of each macroblock within these frames are either spatially or temporally predicted and the resulting prediction is encoded using transform encoding.

For I frames, only intra predictions are used, which exploit spatial redundancy. This means that a macroblock is predicted based on correlation with pixels that were coded already.

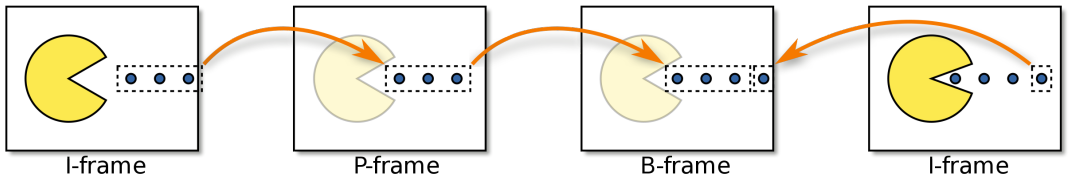


Figure 2.7: The difference between I, P and B frames (Wikipedia, 2019). An I frame encodes an entire image. P frames encode differences with respect to a previous frame. B frames are similar to P frames, but also use information from future frames.

P and B frames are coded using inter predictions as well, which exploit temporal redundancy, i.e. corresponding macroblocks between frames are encoded using a motion vector based on motion estimation. Using the motion vector, these frames thus encode differences with respect other frames.

A B frame is similar to a P frame, but it encodes differences with respect to both the previous frame and the next frame. This allows for more compression than the P frame.

Transform encoding

After encoding, all luma and chroma samples either spatially or temporally, the residual image, i.e., the difference between the encoded and raw image, is encoded using transform encoding with a separable integer transform with similar properties as a 4×4 discrete consine transform.

The resulting coefficients are quantized according to a quantization parameter, which is a trade-off between image quality and compression. The quantized transform coefficients are then encoded using entropy encoding with a context-adaptive variable length coding scheme.

De-blocking filter

One of the artefacts in a block-based coding format is the blockiness of the decoded signal. Block-like structures are visible in the decoded video. An example of such blockiness is shown in Figure 2.8.

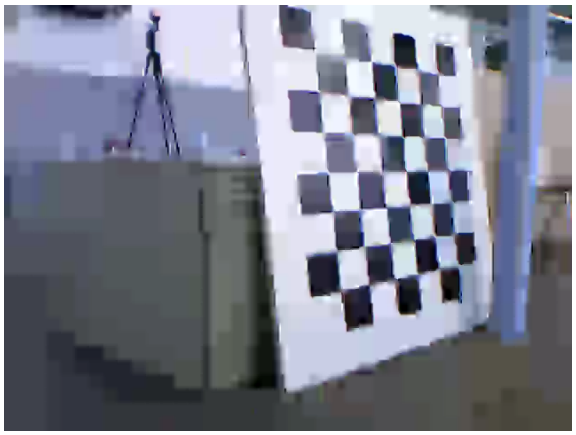


Figure 2.8: Blockiness due to encoding on an image from the RGB-D dataset (Sturm et al., 2012).

To remove this blockiness from the output, a de-blocking filter is applied in the decoder. The de-blocking filter reduces the blockiness without decreasing the sharpness of the pictures. The filter tries to estimate whether the blockiness is caused by quantization or represents an actual edge, based on multiple thresholds.

2.6.3 Rate control for H.264 encoding

As explained in the previous paragraphs, the coefficients of the transform encoding are quantized using a quantization parameter, which is a trade-off between image quality and compression. There are multiple ways in which this quantization parameter can be configured (Robitza, 2017b). Each configuration results in a different encoding strategy and will influence both the quality of the video, as well as the resulting bit rate. The configurations are referred to as rate control methods, as they allow control of the bit rate.

Constant quantization parameter

The constant quantization parameter (CQP) applies the same quantization parameter to every frame. Therefore, the same compression is applied to every frame. As the residual image and entropy is not equal for every frame, the resulting bit rate is not constant, but will hugely vary.

Average bit rate

To obtain a less varying bit rate, the average bit rate (ABR) control option can be used. Using this rate control option the encoder will estimate the required quantization parameter to reach a desired average bit rate. The resulting bit rate is more constant. However, during the first frames, while the encoder is still trying to reach the average bit rate, the bit rate will vary more.

Constant bit rate

An even stricter constant bit rate can be obtained using the constant bit rate (CBR) option. This enforces the encoder to generate a constant bit rate by varying the amount of compression. The encoder does not generate a lower bit rate and hence wastes bandwidth for frames that could be compressed further. As a result of the constant bit rate, the quality will highly fluctuate. Hence, for high-entropy frames, artefacts such as blockiness are more prevalent.

Multi-pass average bit rate

As an encoder cannot predict the compression ahead of time, it cannot compress a video using an optimal trade-off between quality and bit rate. To solve this, the encoder can try the encoding two or more times when a multi-pass average bit rate is configured. This improves the trade-off between quality and bit rate at the cost of computation time.

Constant rate factor

A constant rate factor (CRF) setting instructs the encoder to use different quantization parameters for different frames to create a constant perceived quality, while optimizing the compression ratio. It allows the encoder to make smart decisions such as applying more compression to high-motion frames, which uses the fact that the human visual system is not able to notice quality differences that well when a frame contains motion. While the perceived quality will be more constant, the resulting bit rate will fluctuate. Each increment of 6 for the CRF roughly halves the bit rate Robitza (2017b).

Video buffer verifier

To cope with a varying bit rate, a video buffer verifier (VBV) can be used to create a more constant bit rate without compromising on quality. The VBV uses a hypothetical buffer at a decoder to limit overflow and underflow at the decoder. This technique is useful when a video is encoded for a decoder with a constant reading rate, such as a DVD player.

The concept is a bit counter intuitive. If the bit rate is too high, it will result in an underflow error at the buffer. This is because the decoder, which will read at constant rate from the buffer, will read data to fast for the buffer to fill itself. If the bit rate is too low, the decoder will not read the data from the buffer fast enough, which will result in an overflow error at the buffer.

The mechanism allows the resulting encoding to have short spikes in bit rate and short low bit rate periods, as long as the buffer does not over or underflow. The VBV can be used in combination with the other rate control settings.

2.7 Summary

In this chapter theoretical background was provided regarding visual tracking and H.264 encoding. It was explained how high-gradient points can be selected from a video frame and tracked throughout subsequent frames by minimizing the SSD of a patch around the point along the epipolar line. In a brief introduction to visual SLAM it was explained how this tracking is used in direct methods to estimate depth of these points as well as the pose of the camera. Finally, in the brief introduction to H.264 encoding it was explained how the H.264 encoder reduces video data using chroma subsampling and inter and intra prediction, exploiting spatial and temporal redundancy. The resulting encoded video may contain artefacts such as blockiness. The quality of the video can be controlled using several rate control methods.

In the next chapter the ways in which video data can be optimized for throughput are analysed as well as the impact of throughput reductions on computer vision.

3 Analysis

In the previous chapter, a theoretical background regarding visual simultaneous localization and mapping (SLAM) and H.264 encoding was provided. In this chapter, the theoretical background is used to analyse the main problem of a wireless video stream, which is that the throughput is not always large enough to transmit all video information. After defining the cause of this problem, three types of scaling are analysed that can be used to solve the problem. Subsequently, the impact of these types of scaling on visual tracking algorithm is discussed.

3.1 The limited throughput of a wireless connection

The throughput of a wireless connection is limited and varies depending on the environment. External disturbances, such as signal interference and multipath fading lower the signal-to-noise ratio (SNR), which results in loss of data.

To cope with the lower SNR, IEEE 802.11 wifi standards use adaptive coding and modulation (ACM). With ACM, the coding and modulation scheme is changed to a configuration that is more robust to interference when the SNR of the channel decreases. This robustness comes at the cost of data rate. In the extreme case, where interference is very high, the resulting data rate can become as low as 6.5 Mbps (Perahia and Stacey, 2013).

The data rate of 6.5 Mbps is a theoretical maximum data rate. Protocols, such as the user datagram protocol (UDP) and the real-time transport protocol (RTP), add additional data to the video data in order to transmit it via the network. Therefore, the throughput for video data is much lower than 6.5 Mbps when the SNR of the wireless channel is low.

Furthermore, the available data rate is shared when multiple video streams are present on the same wireless channel. For two or three simultaneous streams, the data rate reduces to 3.25 Mbps and 2.17 Mbps respectively.

A typical full HD H.264 video stream requires 5 to 12 Mbps on average. Peak bit rates are much higher, because not all frames can be compressed to the same extent. Such a video stream cannot always be fully transmitted over the wireless connection.

The loss of data causes visible streaming artefacts, of which an example is shown in Figure 3.1. Such artefacts impact the performance of computer vision, because some parts of the images are not visible, have different pixel intensities, or are displaced.

To optimize the video stream for a visual SLAM algorithm while preventing streaming artefacts, data must be strategically discarded. In this thesis, three types of scaling are considered for reducing the required throughput: *spatial* scaling, *temporal* scaling and *quality* scaling.

In the next sections, the impact of each of these three types of scaling on the required throughput is discussed, as well as the impact on the performance of computer vision. The latter is analysed qualitatively by considering the use case of a visual tracking algorithm. Subsequently, the combination of different types of scaling is analysed, such that a trade-off between types can be made.

A quantitative analysis is not possible without conducting experiments, because the impact of encoding and scaling on bit rate and visual tracking performance depends on the content of the videos. At the end of the chapter, it is determined, which experiments are needed, based on the qualitative analysis, for a quantitative analysis of the impact of each type of scaling on bit rate and visual tracking performance.

(a) A decoded video frame without artefacts.

(b) A decoded video frame with artefacts, obtained by randomly altering 50 bytes in a video file of 13 MB.

Figure 3.1: A decoded video frame without artefacts and the same frame in a corrupted video file. In the corrupted frame, it can be seen that some parts of the image are displaced or distorted. The images are part of the RGB-D dataset (Sturm et al., 2012).

3.2 Spatial scaling

Spatial scaling reduces the size of a video by scaling the *resolution* of a video. Since the number of pixels for each frame of de video decreases when the resolution decreases, the video can be represented using less data. The principle of spatial scaling is shown in Figure 3.2.

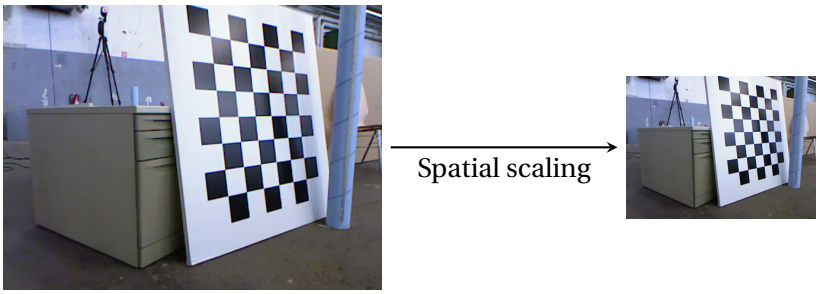


Figure 3.2: Spatial scaling reduces the number of pixels and hence the number of points that can be selected for tracking. The images are part of the RGB-D dataset (Sturm et al., 2012).

3.2.1 Impact on throughput requirements

When the width and height of a video frame are scaled to half of the initial width and height, only a quarter of the original video data is left. In theory spatial scaling can, therefore, reduce the data size quadratically. The encoder, however, applies several advanced methods to optimize video compression.

As explained in Section 2.6, an encoder tries to present the same information using less data during compression by exploiting spatial redundancy. Spatial redundancy can be considered as a measure for information density. When there is a lot of spatial redundancy in a video frame, it can be said that the information density is low, as a lot of visual information in a frame is redundant.

The information density of frames with a higher resolution is often lower than that of frames with lower resolution, as these frames represent the same visual information. When more

pixels are used to represent the same visual information, the probability that pixels convey redundant information increases.

Hence, it is expected that video frames at lower resolutions can be compressed less than frames at higher resolutions, as the encoder can exploit less spatial redundancy. The resulting encoded video will therefore require more than a quarter of the original data, when the width and height of each frame is scaled to half of the initial width and height. The other way around, it is expected that a higher resolution will not result in a quadratic increment in the bit rate of the video.

3.2.2 Impact on visual tracking

Spatial scaling impacts the performance of a visual tracking algorithm in multiple ways. First of all, the quadratic change in available pixels impacts the amount of points that can be selected for tracking. Since most tracking algorithms do not track pixels that have low gradients, as they do not convey much information, the relation between the amount of trackable points and the number of available pixels is not expected to be linear. As scaling the resolution down in most cases leads to a reduction of information, i.e., the number of high-gradient pixels decreases, the number of trackable points most likely decreases when spatial scaling is applied to a video.

A second way in which spatial scaling affects tracking performance, is that it affects the quantization error in the depth estimate. Even though it is possible to perform sub pixel matching using interpolation, the uncertainty in the estimated pixel location results in a larger uncertainty in the corresponding depth estimate when the resolution is smaller. The uncertainty in the depth estimate is therefore expected to grow when the resolution is scaled down and to decrease when the resolution is scaled up.

The third way in which resolution affects tracking performance, is that scaling the resolution down has the effect of a low-pass filter. Most resolution scaling algorithms do not just discard pixels. Instead, such algorithms take multiple pixel intensities into account. The pixel intensities in the scaled video frame represent weighted averages of a group of pixel intensities in the unscaled video frame. Averaging over a group of pixels shifts the pixel intensities closer to a local mean of the area surrounding a pixel. As a result, the gradient of these pixels decreases. In Section 2.4 it was explained that pixels with lower gradients are more difficult to track. Hence, it is expected that spatial scaling increases the probability of a mismatch.

The fourth way in which spatial scaling affects tracking performance, is through noise in the image. Higher resolution images contain relatively more photometric noise. Photometric noise can result in high-gradient values that do not correspond to real-world features. Such noise can then be wrongly selected as point of interest or wrongly matched to a point that is tracked. For lower resolution images, this noise is filtered out by the low-pass filtering effect of the scaling. As explained in Section 2.6.1, an encoder uses transform encoding to encode the residual image. As a result, an encoder filters out high-frequency components during transform encoding. Therefore, it is expected that photometric noise will not be present in encoded videos. Hence, photometric noise will not affect tracking performance.

3.3 Temporal scaling

Temporal scaling reduces the size of a video by discarding entire frames from the video. This allows a video stream to take more time for the transmission of each frame, such that it has enough time to transmit an entire frame before the next frame has to be transmitted. While it decreases the required throughput, it also decreases the overlapping area between frames, as shown in Figure 3.3.

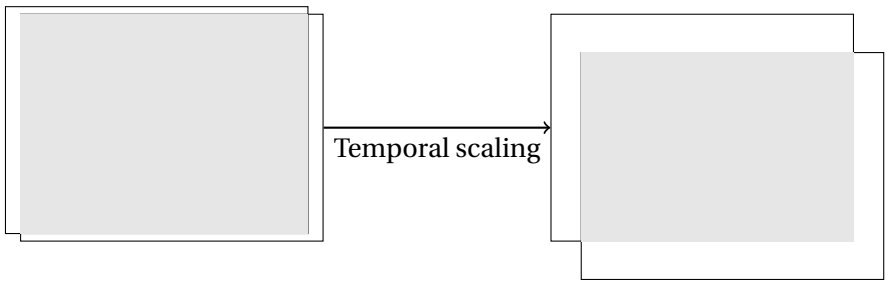


Figure 3.3: The overlapping area between two consecutive frames. Temporal scaling reduces the overlapping area between frames when the camera moves with respect to the observed scene.

3.3.1 Impact on throughput requirements

When the frame rate is halved, by discarding every other frame, the video data is theoretically reduced to half the original size. However, similar to how spatial scaling does not achieve the theoretical maximum data reduction, discarding every other frame does not result in a 50% data reduction after compression.

This is because encoders make use of intra frame encoding. As explained in Section 2.6.2, the H.264 encoder encodes only the difference between frames in P or B frames. When a camera moves with respect to the scene, or observes a dynamic scene, the overlap between frames decreases when the time difference between frames increases. Hence, if frames are dropped as a result of temporal scaling, there is less overlap between consecutive frames. The encoder, therefore, needs more data to encode the difference between frames. The required throughput after encoding is thus expected to be reduced by less than 50% when the frame rate is halved.

3.3.2 Impact on visual tracking

Temporal scaling affects the performance of a visual tracking algorithm in multiple ways. As the overlapping area between frames is related to the frame rate, scaling the frame rate also scales the overlapping area between frames. When a constant velocity model is considered, the size of the overlapping area between frames is directly proportional to the frame rate of the video. Since only the points in this overlapping area can be successfully tracked between frames, the amount of trackable points is also directly related to the frame rate. Fewer points can be tracked when the frame rate is downscaled.

The frame rate also affects the update frequency of matched points. The uncertainty of a depth estimate, that corresponds to a matched point, decreases each time a point is successfully and accurately matched, until it converges to the measurement uncertainty. Therefore, as long as the video is encoded using a high enough quality, the uncertainty of the depth estimates converges faster when the frame rate increases. However, when the encoding quality of the video is low, a higher frame rate actually has a negative impact on tracking performance, because the measurement error, that is related to the encoding quality, increases. As a result, points are matched at random depths and the uncertainty of the depth estimate increases. This prevents the algorithm from building an accurate 3D map.

The uncertainty does not only affect the depth estimate of a pixel, it also affects the size of the search area along the epipolar line. Assuming a Gaussian distributed likelihood, the search area should cover three times the standard deviation in both directions of the epipolar line, such that the probability that the pixel is inside this search area is 99.7%. The search area for a pixel in a frame is thus proportional to the uncertainty of a pixel. As the probability of a mismatch increases when a search is performed across more pixels, the probability of a mismatch

increases when the uncertainty is larger. Hence, when the frame rate is increased while the camera is moving with respect to the observed scene.

3.4 Quality scaling

Quality scaling reduces the size of a video by changing the compression that is applied to the video. Increasing the compression ratio reduces the amount of data that is used to represent a video frame. The data reduction comes at the price of video quality. As shown in Figure 3.4, quality scaling introduces visual distortions, such as blockiness.

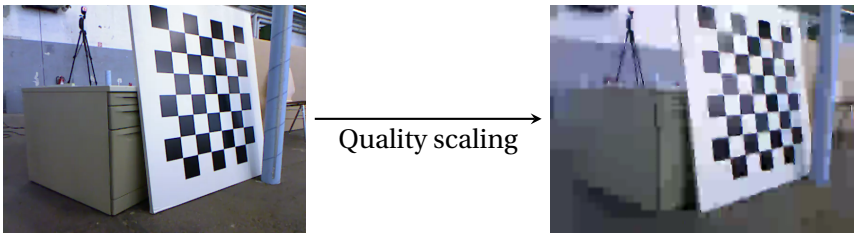


Figure 3.4: Quality scaling increases the visible distortions such as blockiness. The images are part of the RGB-D dataset (Sturm et al., 2012).

3.4.1 Impact on throughput requirements

The quality of an encoded video is not directly related to its bit rate. Some frames require more data for the same perceived quality than others. For example, a completely black frame can be compressed far more than a frame that contains a lot of detail.

Besides the complexity of the image, the bit rate depends on the amount of computation time that an encoder has. For example, when an encoder can do two or more passes on the video data, it can optimize the compression ratio far better than in a single pass. Hence, the bit rate of an encoded video depends on the settings of the encoder. In general, reducing the quality reduces the bit rate, but the amount by which the encoder is able to reduce the bit rate is difficult to predict, as it depends on the video.

As explained in Chapter 2, there are multiple ways to control the quality of the encoding. The output can be indirectly scaled by setting a constant bit rate (CBR) or a target average bit rate (ABR). These settings directly result in an average bit rate, however, there may still exist large peaks in the bit rate. Furthermore, there is no control over the resulting quality.

It is also possible to control the quality more directly by setting a constant rate factor (CRF). The encoder then optimizes the perceived quality. However, the resulting bit rate varies and cannot be determined analytically. Each increment of 6 for the CRF roughly halves the bit rate Robitza (2017b).

3.4.2 Impact on visual tracking for the static case

As explained in Section 2.6.2, the H.264 encoder applies transform encoding to encode residual images. Increasing the compression ratio reduces the amount of data that is used for the transform encoding. Hence, higher frequency components are filtered out. As high-gradient pixels, which correspond to higher frequencies, are better trackable than low-gradient pixels, it is expected that reducing quality increases the probability of mismatches.

The impact of quality scaling on visual tracking, however, is mainly caused by the streaming distortions that are present when the quality is decreased. The distortions cause local changes in image intensity, which increase probability of a mismatch.

Besides that, distortions such as blockiness introduce artificial edges in the video frame. These artificial edges can end up being wrongly selected as point to track.

In general, frames with low quality should not be used for point selection, and ideally not for matching either.

3.5 Trade-off between types of scaling

Now that the impact of spatial, temporal and quality scaling on both the required throughput and the visual tracking performance is analysed, it is time to compare these types of scaling with each other and look at their combined impact. First, the combination of quality scaling and each of the other two types of scaling is discussed. After that, spatial and temporal scaling are compared.

3.5.1 Quality and spatial scaling

As discussed in Section 3.2, the number of points that can be tracked relates to the resolution of the video. When the resolution is increased, more points can be tracked and the uncertainty in the depth estimate becomes smaller.

When the encoding quality of a high resolution video is low however, the added value of the extra trackable points is counteracted by the distortions. Therefore, it is not beneficial to increase the resolution when the encoding quality is low.

When the encoding quality is higher, the extra tracked points can be used effectively to improve the visual tracking. Hence, there is a minimum quality factor for each resolution for which it becomes beneficial to increase the resolution. Until this quality factor is reached, the resolution should not be increased. The reverse is also true, i.e., the resolution should only be scaled down when the minimum quality for the current resolution is reached.

3.5.2 Temporal and quality scaling

As discussed in Section 3.3, a higher frame rate is only beneficial for the uncertainty of depth estimates when the encoding quality is high. The same reasoning applies to the added value of a larger overlapping area between frames.

Increasing quality at the cost of frame rate reduces the uncertainty of tracked points. However, it also reduces the number of trackable points when the camera is moving.

Similar to spatial scaling, there will be a minimum quality factor for which a further decrease in quality would prevent accurate tracking for more points than added by the overlapping area as a result of a higher frame rate. This factor will depend on the velocity of the camera with respect to the observed scene, as the overlapping area is a function of the velocity of the camera with respect to the observed scene and the frame rate.

3.5.3 Spatial and temporal scaling

Both spatial and temporal scaling affect the amount of points that can be tracked between frames. For spatial scaling this amount is more or less directly related to the resolution. For temporal scaling, the relation between the amount of trackable points depends on both the frame rate and the velocity of the camera with respect to the observed scene.

When the velocity of the camera with respect to the observed scene is high, most of the extra points that are added by the higher resolution fall outside the overlapping area between frames. Hence, in such a situation increasing the frame rate is more beneficial for the tracking performance than increasing the resolution, as increasing the overlapping area has a larger impact on the amount of trackable points. This suggests that there is an optimal trade-off between spatial

and temporal scaling that depends on the velocity of the camera with respect to the observed scene.

When a static scene is assumed, this trade-off can be derived analytically, using the angular and linear velocity and the field of view of the camera, by assuming a uniform distribution of points in each frame.

3.6 Conclusion

The three types of scaling that have been discussed all impact the required throughput and visual tracking performance in a different way. As the impact of the H.264 encoder is difficult to predict, impact can only be analysed qualitatively. Experiments are required for a qualitative analysis.

The impact of the three types of scaling on throughput is difficult to predict for all types of scaling, as the encoder optimizes compression using a variety of advanced methods.

The quadratic and linear reductions in data that are theoretically possible for respectively spatial and temporal scaling are not expected to be reached in reality after encoding.

For quality scaling the relationship between bit rate and quality is even harder to predict. However, the bit rate can be effectively set by configuring a CBR in the encoder.

For all types of scaling the required throughput is be reduced when scaling down. To find the magnitude of this reduction, the bit rate of scaled and encoded videos should be analysed via experiments.

The expected results are:

- Spatial scaling reduces data less than quadratically.
- Temporal scaling reduces data less than linearly.
- Quality scaling halves the data each time the CRF is incremented by 6.

Regarding the performance of a visual tracking algorithm, it can be concluded that the distortions induced by lowering the quality counteract the performance gains of higher resolutions and frame rates.

It is expected that for each combination of resolution and frame rate a minimum quality factor exists that should be reached before applying spatial or temporal scaling. To find this minimum quality factor, the performance of a visual tracking algorithm should be evaluated for each combination of frame rate and resolution while varying this quality factor.

The impact of temporal scaling depends on the velocity of the camera with respect to the observed scene, as it determines the size of the overlapping area between frames. The impact of temporal scaling increases when the velocity increases. Therefore, the trade-off between spatial scaling and temporal scaling depends on the velocity as well, because the size of the overlapping area determines the impact of spatial scaling.

Experiments are needed to quantify the impact of spatial, temporal, and quality scaling on visual tracking performance. Based on the qualitative analysis, the expectations are:

- There exists a minimum quality that should be reached before applying any other type of scaling.
- Distortions induced by quality scaling counteract benefits of higher resolutions and frame rates.
- The trade-off between spatial and temporal scaling depends on the velocity of the camera with respect to the observed scene. For higher velocity videos it is better to apply more spatial scaling, and for lower velocity videos it is better to apply more temporal scaling.

4 Test design

In the previous chapter, it was concluded that it is difficult to predict the impact of spatial, temporal and quality scaling on required throughput, because of the compression methods that the encoder uses, which depend on the content of the video. Therefore, the actual reduction should be measured through experiments.

Besides the expected effects on throughput, the expected effects of each type of scaling on visual tracking performance has been discussed. It was concluded that there is a trade-off between spatial and temporal scaling and that for each combination of resolution and frame rate there is a certain threshold for the quality factor that must be reached before changing the spatial and temporal scaling factors.

The actual effects of scaling on the required throughput, as well as the effects of combined scaling on visual tracking performance, can only be quantitatively analysed through experiments. The threshold value for the quality and the trade-off between spatial and temporal scaling can be obtained by analysing the results of these experiments.

In this chapter the design of these experiments is explained. First, an overview of the test framework is given, after which the implementation and used algorithms are further explained. The source code of the scripts that are referred to in this chapter is available via the GitLab repository of the Robotics and Mechatronics group of the University of Twente. A copy of the `README` file is presented in Appendix B.

4.1 Overview of the setup

An overview of the experimental setup is shown in Figure 4.1.

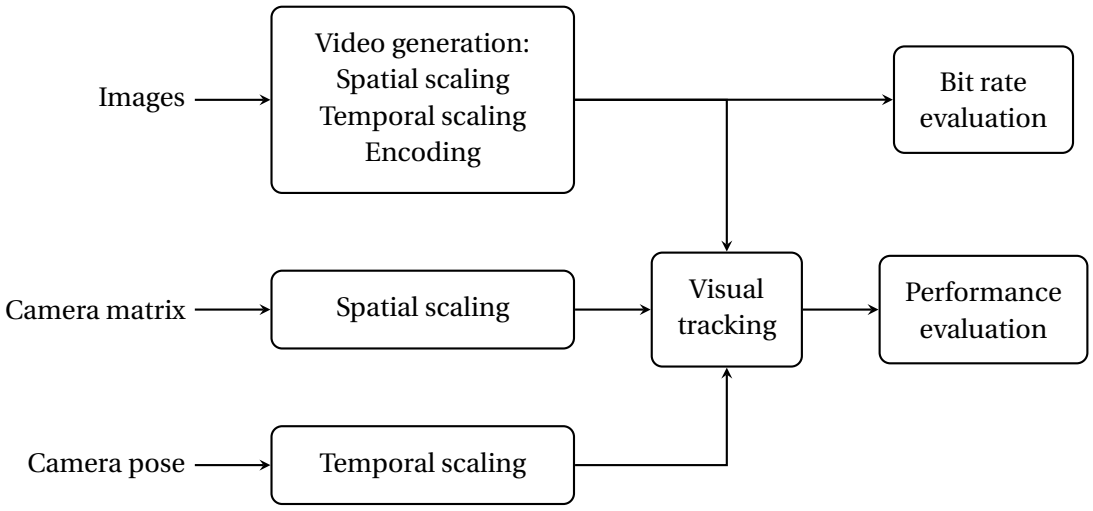


Figure 4.1: The experimental setup

Image sequences from a variety of datasets are converted into encoded videos using `ffmpeg`, which is a widely used, free and open-source project that can create encoded video streams. It is supported by most operating systems, such that it can be used on a robot, as well as on a workstation during experiments. More information regarding the datasets that are used for generating the videos, is provided in Section 4.8.

Spatial and temporal scaling is applied to the videos, by changing the parameters of `ffmpeg`, as explained in Section 4.2.

The same temporal scaling is applied to the file containing the camera pose, which is provided by the datasets, such that the resulting file contains a camera pose for each frame of the video. This process is explained in Section 4.4.

The camera matrix, which was discussed in Section 2.1, is scaled using the same spatial scaling parameters as used for the video, such that the visual tracking algorithm can project 3D points to the correct pixel locations. In Section 4.3, it is explained how spatial scaling is applied to the camera matrix.

The setup is used for each combination of spatial, temporal, and quality scaling. The range of resolutions and frame rates that are used for generating the videos are based on the available frame rate and resolution of each dataset and are discussed in Section 4.9.

The visual tracking algorithm, which is presented in Section 4.5, runs offline, as the software does not yet have to run in realtime on a robot. For similar reasons, the visual tracking algorithm is not applied to a wireless video stream.

For each block of Figure 4.1 a Node.js script is created to automate the process. This language was chosen by the author because of its development speed.

There is also a script that automates the whole process for a given video sequence. This script generates videos, camera poses and camera matrices for all combinations of scaling. It then runs the visual tracking algorithm on all these videos and outputs the result so that it can be analysed.

In the next sections, the process within each block of Figure 4.1 is explained in more detail. Subsequently, the datasets that are used in the experiments are presented, followed by the chosen scaling parameters for each type of scaling.

4.2 Video generation

`ffmpeg` can be used via the command line interface (CLI) to generate videos. The image sequences from a dataset can be encoded into a video by specifying the paths of the images to `ffmpeg` as a glob using the `-i` option in combination with the `-pattern_type` option set to 'glob'. Each image that matches the specified glob is encoded as a single frame in the video.

For a folder named `images` containing PNG images, the code for generating a video using the H.264 encoder is:

```
ffmpeg -pattern_type glob -i "images/*.png" \
       -codec:v libx264 output_filename.mp4
```

Using additional options, spatial, temporal and quality scaling can be applied on the images before they are encoded into a video.

4.2.1 Spatial scaling with `ffmpeg`

Spatial scaling can be applied to the video with `ffmpeg` via the CLI, by specifying a `-filter:v` option `scale`, where the `:v` means that the filter should be applied on the **v**ideo stream.

`ffmpeg` contains several options for the scaling algorithm, such as bilinear, nearest neighbour, Gaussian, Lanczos and bicubic spline. For this experimental setup, the bicubic scaling algorithm is used, as it is a good trade-off between computation time and image quality. This makes it a suitable scaling method for application on a robot as well.

Given a folder containing PNG images named `images`, a video with a spatial scaling factor of $\frac{1}{2}$ can be obtained using:

```
ffmpeg -pattern_type glob -i "images/*.png" \
       -filter:v "scale=iw/2:ih/2" -sws_flags bicubic \
```

```
        -codec:v libx264 output_filename.mp4
```

4.2.2 Temporal scaling with `ffmpeg`

Temporal scaling can be realized by dropping certain frames. Frames can be filtered using `ffmpeg` via the CLI, by providing a `-filter:v` option `select`.

Using `select="not(mod(n 3))"` as a filter option, every third frame is included in the resulting video. This corresponds with a temporal scaling of $\frac{1}{3}$.

In order for the video to have the correct frame rate it is also necessary to configure the input frame rate and the desired output frame rate.

Given that the PNG images in a folder named `images` were obtained with frame rate of 30 Hz, a video with a temporal scaling factor of $\frac{1}{3}$ can be obtained using:

```
ffmpeg -r 30 -pattern_type glob -i "images/*.png" \
       -filter:v "select=not(mod(n\,3))" \
       -codec:v libx264 -r 10 output_filename.mp4
```

4.2.3 Quality scaling using `ffmpeg`

As explained in Section 2.6.3, there are multiple ways to configure the quality of the H.264 encoder. For the experiments in this research, the constant rate factor (CRF) is used to control the quality of the encoding, as it results in a constant video quality. As a constant bit rate is not required for the experiments, a video buffer verifier (VBV) is not used.

To configure a CRF the `-crf` option can be set to a factor between 0 and 51, where 23 is the default factor. An example is given by:

```
ffmpeg -r 30 -pattern_type glob -i "images/*.png" \
       -codec:v libx264 \
       -crf 23 \
       output_filename.mp4
```

4.2.4 Combining spatial, temporal and quality scaling in `ffmpeg`

The settings discussed in Sections 4.2.1–4.2.3 can be combined into a single `ffmpeg` command, which is given by:

```
ffmpeg -r 30 -pattern_type glob -i "images/*.png" \
       -filter:v "select=not(mod(n\,3)),scale=iw/2:ih/2" \
       -sws_flags bicubic \
       -c:v libx264 -crf 23 \
       -r 10 output_filename.mp4
```

Using this command, a video can be generated for each combination of spatial, temporal and quality scaling, by executing the command with different values for the provided options.

To use the generated video for visual tracking, the camera matrix and camera pose sequences have to be scaled along, which is covered in the next sections.

4.3 Spatial scaling of the camera matrix

When spatial scaling is applied to a video, the dimensions of each video frame change. The original camera matrix should be rescaled such that points in 3D are projected to the correct 2D pixel locations. As discussed in Section 2.1, the projection $\begin{bmatrix} u, v \end{bmatrix}^T$ of a point $\mathbf{p} \in \mathbb{R}^3$ with

respect to the camera frame with camera matrix **K** is given by

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{Kp} \tag{4.1}$$

Spatial scaling can be defined as an operation that, given a scaling factor s_s, maps $\begin{bmatrix} u, v \end{bmatrix}^T$ to $\begin{bmatrix} u', v' \end{bmatrix}^T$, where $u' = s_s u$ and $v' = s_s v$.

Substituting this into Equation 4.1 results in

$$\lambda \begin{bmatrix} s_s^{-1} u' \\ s_s^{-1} v' \\ 1 \end{bmatrix} = \mathbf{Kp} \tag{4.2}$$

This can be rewritten as

$$\lambda \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} s_s & 0 & 0 \\ 0 & s_s & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Kp} \tag{4.3}$$

Hence, the camera matrix $\mathbf{K}'$ of a scaled video is given by

$$\mathbf{K}' = \begin{bmatrix} s_s & 0 & 0 \\ 0 & s_s & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{K} \tag{4.4}$$

A scaled version of the camera matrix as given in Equation 4.4 is computed for each spatial scaling factor, such that it can be used as input for the visual tracking algorithm.

4.4 Temporal scaling of the camera pose

When certain images from sequences are dropped to simulate temporal scaling, the camera pose sequence has to be scaled along, i.e. the camera pose for each frame must be extracted from the camera pose data that is provided by the datasets.

Most datasets contain a list of the images in the sequence, together with the timestamp of the moment of capture. Likewise, a list of camera poses with a timestamp is provided. The timestamps of the captured images do not always correspond with the timestamps of the provided camera poses. The latter are often captured at a higher frequency. Therefore, the camera poses for the images have to be filtered from the list.

This algorithm is straightforward. Given the temporal scaling factor s_t, the list of images L_I and the list of camera poses L_T, the list of temporally scaled camera poses L_{T,s_t} can be obtained using Algorithm 1.

4.5 Visual tracking

After generating a video and scaling the camera matrix and camera pose sequence, these three files are used by a visual tracking algorithm to perform visual tracking.

The impact of spatial, temporal and quality scaling on computer vision is determined by analysing their effects on a visual tracking algorithm. It is expected that the effects on the performance of such an algorithm is similar to the effects on performance of other computer vision algorithms, because a visual tracking algorithm uses basic image processing techniques that are also used in most other computer vision algorithms.

For example, when tracked points are directly used to generate a map and estimate the pose of the camera, the error in these tracked points propagates to the map and pose estimates.

Algorithm 1: Temporal scaling of the camera pose T

Data: set of images L_I, set of camera poses L_T, scaling factor s_t
Result: set of temporally scaled camera poses L_{T,s_t}

$L_{T,s_t} \leftarrow \emptyset$;
Sort L_I and L_T by their timestamp;
foreach *timestamp* $t_{I,i}$ *of entry* i *in* L_I **do**
  if $i \equiv 0 \pmod{s_t^{-1}}$ **then**
    foreach *timestamp* $t_{T,j}$ *of entry* j *in* L_T **do**
      if $t_{T,j} > t_{I,i}$ **then**
        if $j = 0$ *or* $t_{T,j} - t_{I,i} > t_{I,i} - t_{T,j-1}$ **then**
          add T_j to L_{T,s_t}
        else
          add T_{j-1} *to* L_{T,s_t}
        end
        break;
      end
    end
  end
end

Hence, the performance of a complete visual simultaneous localization and mapping (SLAM) algorithm is directly related to the performance of a visual tracking algorithm.

The visual tracking algorithm that is used for the experiments is based on algorithms that are also used in direct implementations of visual SLAM. Recently, direct methods have been part of extensive research. As explained in Section 2.5.2, one of the advantages of direct methods is that they operate directly on the pixel intensities. As a result, direct methods can create dense maps of an environment, while still operating in realtime. The high amount of tracked points makes direct methods also more robust to noise and fluctuations in the quality of the video stream. Furthermore, as direct methods do not depend on a specific choice of feature, the outcome of a direct visual tracking algorithm is applicable to a broader range of tracking algorithms than a feature-based implementation. These advantages of direct methods over indirect methods have led to the conclusion that direct methods are more useful for the use case covered by this research, without sacrificing on generalizability.

Several open source implementations of direct visual SLAM methods exist, such as (Engel et al., 2017, 2014; Forster et al., 2014). However, after experimenting with several of these implementations, none of the available implementations were found suitable for this research. The algorithms either depend on the Robot Operating System (ROS), were not compiling on the available hardware or were crashing during runtime.

After trying to fix errors and rewrite available open source implementations to the specifics of this research, existing implementations were abandoned and a custom implementation was developed instead. This implementation is based on a combination of (Engel et al., 2017; Forster et al., 2014; Vogiatzis and Hernández, 2011; Civera et al., 2008).

As shown in Figure 4.2, it consists of two main algorithms. First, a point selection algorithm is used to determine which points should be tracked. This algorithm selects points based on their gradient, but also optimizes for a uniformly distributed collection of points in a frame. After points have been selected, a visual tracking algorithm performs the actual tracking using search along the epipolar line.

The number of visible points is evaluated for each frame, after tracking points. When the number visible points is below a target value, new points are selected in the current frame, before the next frame is read from the video stream.

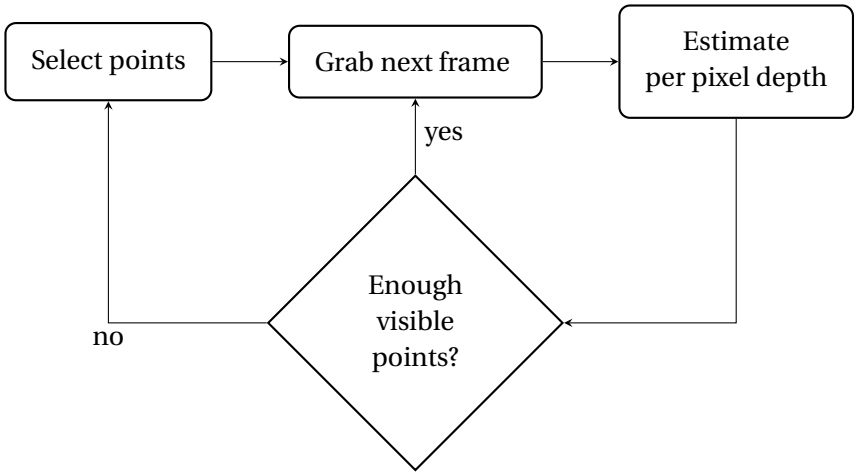


Figure 4.2: The visual tracking algorithm consists of a point selector and a depth estimator.

4.5.1 Pixel selection

Even though it is possible to track all pixels in a frame, not all pixels can be tracked with the same accuracy. As explained in Section 2.4, areas with low gradients are difficult to track, because surrounding pixels have similar intensities. The photometric error is about the same for each pixel within such an area, resulting in inaccurate tracking.

To solve this problem, some visual SLAM methods apply methods such as regularization to the depth estimates (Newcombe et al., 2011; Engel et al., 2013).

Another solution, is to prevent low-gradient points from being tracked, by adding a pixel selection process. Besides increasing tracking accuracy, this also saves a lot of computation time since fewer points have to be matched.

For this reason, a pixel selection process is added to the visual tracking algorithm that is used during the experiments, instead of applying regularization. The algorithm that is used to select pixels for tracking is presented in Algorithm 2. It is based on the point selection algorithm from Engel et al. (2017).

The point selection algorithm optimizes for both distinctiveness and a uniform distortion of points. First, a frame is divided into rectangular regions. For each region a threshold is determined, based on the median of the gradient magnitude within this region.

After determining a threshold for each region, the frame is divided into smaller regions for a second time. For each of these regions, the pixel with the largest gradient is selected as a point for tracking, when it is larger than the threshold for its region.

To obtain an even more uniform distribution, this last process is repeated two times using a lower threshold value and a larger region. This adds a smaller amount of pixels with a lower gradient to the collection of points.

4.5.2 Depth estimation

The depth estimation part of the tracking algorithm is based on a combination of direct visual SLAM methods (Engel et al., 2017; Forster et al., 2014; Vogiatzis and Hernández, 2011; Civera et al., 2008). The algorithm searches an optimal match between a selected point from a pre-

Algorithm 2: The point selection algorithm for the visual tracker

Data: frame $\mathbf{I} \in \mathbb{R}^{q \times r}$, threshold region size ρ_{th}, threshold offset θ, selection region size d
Result: Set of selected points L_p

$L_p \leftarrow \emptyset$;
$\mathbf{T} \leftarrow \mathbf{0}_{q,r}$;
$\mathbf{G}_{abs} \leftarrow AbsoluteGradient(\mathbf{I})$;
;
Define thresholds;
for $i \leftarrow 0$ **to** $\frac{q}{\rho_{th}} - 1$ **do**
  for $j \leftarrow 0$ **to** $\frac{r}{\rho_{th}} - 1$ **do**
    $m_{abs} \leftarrow median(\mathbf{G}_{i:i+\rho_{th}, j:j+\rho_{th}})$;
    for $k \leftarrow 0$ **to** $\rho_{th} - 1$ **do**
      for $l \leftarrow 0$ **to** $\rho_{th} - 1$ **do**
        $T_{i\rho_{th}+k, j\rho_{th}+l} \leftarrow m_{abs} + \theta$
      end
    end
  end
end
;
Select points;
for $i \leftarrow 0$ **to** $\frac{q}{d} - 1$ **do**
  for $j \leftarrow 0$ **to** $\frac{r}{d} - 1$ **do**
    $g_{max} \leftarrow 0$;
    $\mathbf{p}_{max} \leftarrow \begin{bmatrix} -1 & -1 \end{bmatrix}^T$;
    for $k \leftarrow 0$ **to** $d - 1$ **do**
      for $l \leftarrow 0$ **to** $d - 1$ **do**
        if $G_{id+k, jd+l} > g_{max}$ *and* $G_{id+k, jd+l} > T_{id+k, jd+l}$ **then**
          $g_{max} \leftarrow G_{id+k, jd+l}$;
          $\mathbf{p}_{max} \leftarrow \begin{bmatrix} id+k & jd+l \end{bmatrix}^T$;
        end
      end
    end
    if $g_{max} > 0$ **then**
      add $\mathbf{p}_{max}$ to L_p
    end
  end
end

vious frame and candidate points on the epipolar line in the current frame by minimizing the sum of squared differences (SSD) of the pixel intensities.

Using the location of the matched pixel, the corresponding depth is estimated. The depth estimate is then used to constrain the search area along the epipolar line in the next frame.

When the uncertainty of the depth estimate is under a certain threshold, the corresponding point in 3D is added to the map. The points that are tracked are referred to as seeds, until they reach the uncertainty threshold and are added to the map.

The algorithm is described in Algorithm 3 and is explained in more detail in the next paragraphs.

Algorithm 3: The depth estimation algorithm for the visual tracker

Data: set of frames L_I, set of camera poses L_T, target #seeds N_s, camera matrix $\mathbf{K}$
Result: map of points M_p

$l_v \leftarrow 0$;
$L_s \leftarrow \emptyset$;
$M_p \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** $|L_I|$ **do**
  if $l_v < N_s$ **then**
    $L_{s^+} \leftarrow$ CreateSeeds(SelectPoints($L_{I,i-1}, N_s - l_v$));
    $L_s \leftarrow L_s \cup L_{s^+}$
  end
  $l_v \leftarrow 0$
  foreach $s_j \in L_s$ **do**
    if *IsVisible*($s_j, \mathbf{L}_{I,i}, \mathbf{K}, \mathbf{L}_{T,i}$) **then**
      $L_c \leftarrow$ ComputeEpipolarLinePoints($s_j, \mathbf{L}_{T,i}, \mathbf{K}$);
      $SSD_{min} \leftarrow \infty$;
      $c_{opt} \leftarrow \begin{bmatrix} -1 & -1 \end{bmatrix}^T$;
      foreach $\begin{bmatrix} u_c & v_c \end{bmatrix}^T \in L_c$ **do**
        $SSD \leftarrow$ ComputeSSD($s_j, \begin{bmatrix} u_c & v_c \end{bmatrix}^T, \mathbf{L}_{I,i}$);
        if $SSD < SSD_{min}$ **then**
          $SSD_{min} \leftarrow SSD$;
          $c_{opt} \leftarrow \begin{bmatrix} u_c & v_c \end{bmatrix}^T$
        end
      end
      $s_j \leftarrow$ UpdateSeedDepthAndUncertainty($s_j, \mathbf{K}, \mathbf{L}_{T,i}, c_{opt}$);
      if *UncertaintyBelowThreshold*(s_j);
      then
        $\mathbf{p} \leftarrow$ Extract3DPoint(s_j);
        Add $\mathbf{p}$ to M_p;
        Remove s_j from L_s
      else
        $l_v \leftarrow l_v + 1$
      end
    end
  end
end

Seeds

Seeds are generated using the points that are selected by the point selection algorithm, and updated using the matched pixel. The concept of these seeds is based on the work of Vogiatzis and Hernández (2011), who introduced a Gaussian + uniform mixture model for vision-based depth estimates:

$$p(x_n|Z,\pi) = \pi N(x_n|Z,\tau^2) + (1-\pi)U(x_n|Z_{min},Z_{max}) \tag{4.5}$$

Where x_n is the matched pixel location, Z is the depth, τ^2 the variance of a good measurement and π the probability of an inlier, which can be modelled by the inlier count. Z_{min} and Z_{max} contain the prior knowledge about the minimum and maximum observable depth in a scene.

The Gaussian distribution in Equation 4.5 models the depth estimate for good matches and the uniform distribution models the depth estimate for mismatches. The posterior of the depth estimate is:

$$p(Z,\pi|x_0,\ldots,x_n) \propto p(Z|\pi)\prod_n p(x_n|Z,\pi) \tag{4.6}$$

which can be approximated by a Gaussian × beta distribution:

$$q(Z,\pi|a_n,b_n,\mu_n,\sigma_n) = beta(\pi|a_n,b_n)N(Z|\mu_n,\sigma_n^2) \tag{4.7}$$

Where a_n and b_n are counts for respectively the number of inliers and the number of outliers, to model the inlier probability π, and μ_n and σ_n^2 represent the mean and variance of the Gaussian distributed depth estimate.

When a new depth measurement is added to Equation 4.7, the updated posterior becomes:

$$C \times p(x_n|Z,\pi)q(Z,\pi|a_{n-1},b_{n-1},\mu_{n-1},\sigma_{n-1}) \tag{4.8}$$

Where C denotes a constant. This equation is not a Gaussian × beta distribution, but can be approximated as one by matching the moments of Equation 4.8 and Equation 4.7.

The variance of the posterior, σ_n^2, is used to determine if a depth estimate converged to an accurate enough depth estimate. This is considered the case when the variance is lower than $\frac{Z_{min}-Z_{max}}{10000}$ and the inlier ratio $\frac{a_n}{a_n+b_n}$ larger than 0.1.

In contrast with Vogiatzis and Hernández (2011), the visual tracking algorithm estimates the *inverse* depth instead of the depth, similar to Newcombe et al. (2011) and Forster et al. (2014). This is because the inverse depth is better parametrized by a Gaussian distribution than the depth (Civera et al., 2008). Furthermore, it is less complex to describe a point at infinity in software using an inverse depth of zero.

Similar to Vogiatzis and Hernández (2011), seeds are initialized such that the depth range is covered by three times the standard deviation in both directions, using $a_0 = 10, b_0 = 10, \mu_0 = \frac{d_{max}-d_{min}}{2}, \sigma_0^2 = \left(\frac{d_{max}-d_{min}}{6}\right)^2$, where d_{max} and d_{min} are the maximum and minimum inverse depths.

Epipolar geometry

Each seed is projected onto to the next frame using the spatially scaled camera matrix and the temporally scaled camera pose. Using the basic equation for the pinhole camera model,

which was presented in Section 2.1, the projection of a point $\mathbf{p} \in \mathbb{R}^3$, expressed in the coordinate system of the camera at frame i, is given by:

$$z \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{Kp} \tag{4.9}$$

Where $\mathbf{K}$ is the camera matrix, z the depth and u_i, v_i the pixel coordinates of the projection in frame i. The projection of the same point in frame j is given by:

$$\lambda \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = \mathbf{KRp} + \mathbf{Kt} \tag{4.10}$$

Where $\mathbf{R}$ and $\mathbf{t}$ denote respectively the relative rotation and translation of the camera in frame j, with respect to the pose of the camera in frame i.

Substituting Equation 4.9 into Equation 4.10 gives:

$$\lambda \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = z\mathbf{KRK^{-1}} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} + \mathbf{Kt} \tag{4.11}$$

Multiplying Equation 4.11 by the inverse depth $d = z^{-1}$ results in:

$$d\lambda \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = \mathbf{KRK^{-1}} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} + d\mathbf{Kt} \tag{4.12}$$

which expresses the epipolar line of a point from frame i in frame j, using the relative rotation and translation of the camera and the camera matrix as a function of the inverse depth d. The constrained search area for a pixel match in frame j is determined using Equation 4.12, by projecting the minimum $\mu_{j-1} - 3\sigma_{j-1}$ and maximum $\mu_{j-1} + 3\sigma_{j-1}$ inverse depths.

Sum of squared differences

Candidate pixels are selected by sampling the line segment of the epipolar line between the projections of the estimated minimum and minimum inverse depth, with a discrete step size of one pixel. For each of these candidates pixels, the SSD between the pixel intensities of a patch around the point in frame i and the projection of this patch in frame j is computed using:

$$SSD(d_c) = \sum_N \left(I_i(\mathbf{u}_{i,n}) - I_j(\mathbf{P}(\mathbf{u}_{i,n}, d_c, \mathbf{K}, \mathbf{R}, \mathbf{t})) \right)^2 \tag{4.13}$$

Where $I_j(\mathbf{u})$ retrieves the intensity of pixel $\mathbf{u}$ in frame j, and $\mathbf{P}(\mathbf{u}_{j,n}, d_c, \mathbf{K}, \mathbf{R}, \mathbf{t})$ is the projection function that maps a point $\mathbf{u}_i$ to frame j following Equation 4.12. The candidate with the minimum SSD is selected as matching pixel and used to update the posterior parameters $a_n, b_n, \mu_n, \sigma_n$ of the seed.

4.6 Bit rate evaluation

To determine the impact of spatial, temporal, and quality scaling on the required throughput, the bit rate of each generated video is evaluated. `ffmpeg` contains a CLI tool for this: `ffprobe`.

`ffprobe` can be configured to analyse the video stream by setting the `-select_streams` option to `v`. Frame information can be extracted using the `-show_frames` option. By specifying `csv` as `-print_format`, the output can be written as comma-separated values (CSV). The columns that should be present in the CSV can be selected using the `-show_entries` option.

An example output from `ffprobe` is shown in Table 4.1. This table shows the first 10 entries of the output generated by:

```
ffprobe -loglevel error -select_streams v \
        -show_frames -show_entries \
        frame=pkt_pts_time,pkt_size,pict_type \
        -print_format csv corbs1_1920x1080_30_29
```

Timestamp	Size in bytes	Frame type
0.000000	93020	I
0.033333	6367	B
0.066667	7595	B
0.100000	30478	B
0.133333	24985	P
0.166667	4609	B
0.200000	7449	B
0.233333	5511	B
0.266667	34180	P
0.300000	5008	B

Table 4.1: Example data obtained from `ffprobe`.

The bit rate throughout a video can be computed from the results, using the timestamps and the size of the frames.

4.7 Performance evaluation

To evaluate the performance of the visual tracking algorithm, two metrics are used:

- The average number of points that can be selected for tracking per frame.
- The number of points that are added to the map.

The number of points that can be selected indicate the potential density of the map that can be obtained via tracking. Since in general, a denser map is more useful, the number of trackable points are considered a performance indicator.

The number of trackable points is determined by applying the point selection algorithm from Algorithm 2 to several frames of each generated video, using region size $d = 1$. The results are then averaged over the number of frames.

The density of the map is also affected by the uncertainty of the visual tracking algorithm. As explained in Section 4.5.2, each time an inverse depth is estimated, the parameters of the corresponding probability distribution are updated. When the resulting uncertainty is below a threshold, the depth estimate is considered accurate enough to add the point to the 3D map. Hence, the total amount of points in the map indicates the performance of the visual tracking algorithm.

The number of points in the map is analysed, by applying the visual tracking algorithm to each generated video, using a target number of visible points of 200. By using the same tar-

get number of visible points, the impact of the different scaling parameters on the uncertainty of tracked points can be evaluated.

4.8 Selected datasets

Video processing algorithms are generally evaluated on standardized datasets. This allows for objective comparison between algorithms. There exist several benchmark datasets that contain image sequences captured by a camera. For the performance evaluation of visual SLAM and visual tracking algorithms, such sequences also contain recordings of the camera trajectory. The camera trajectory can be used to determine the pose of the camera for each video frame. This pose is referred to as the ground-truth camera pose, as it is accurate enough to consider it the ground-truth.

Two datasets are used for the experiments to analyse the impact of spatial, temporal, and quality scaling on bit rate and tracking performance: the TUM RGB-D dataset (Sturm et al., 2012) and the CoRBS dataset (Wasenmüller et al., 2016). Both datasets contain image sequences and high-frequency camera pose information, collected by high-accuracy motion-capture equipment, for different scenes.

Videos captured by the Dutch National Police (NPN) are not available and hence not used for the experiments. The two selected datasets were chosen to be as close to a realistic scenario as possible.

4.8.1 TUM RGB-D dataset

The RGB-D dataset from the Technical University of Munich (Sturm et al., 2012), is a benchmark dataset that can be used to evaluate visual SLAM methods and contains a large set of image sequences obtained using a Microsoft Kinect.

Images are recorded at a frame rate of 30 Hz with a resolution of 640×480 pixels. The images are provided in portable network graphics (PNG) format, which is a lossless format that contains 8-bit red, green and blue pixel intensities.

The ground-truth camera trajectory is obtained at a frequency of 100 Hz using a high-accuracy motion-capture system.

Image sequences are provided in several categories, such as handheld SLAM, 3D Object Reconstruction and Robot SLAM. For this thesis the robot pioneer sequences 1, 2 and 3 have been selected, as they contain images captured by a robot that is navigated through a maze of tables, containers and walls. Hence, these sequences contain videos similar to the situation of the NPN. There are, however, no moving objects observed by the camera. All three sequences contain between 2 and 3 minutes of video. In Figure 4.3 a few example frames of these sequences are shown.

(a) **(b)** **(c)**

Figure 4.3: A few video frames from the RGB-D dataset (Sturm et al., 2012).

(a) (b) (c)

Figure 4.4: A few frames from the CoRBS Desk sequence (Wasenmüller et al., 2016).

4.8.2 CoRBS dataset

The CoRBS dataset (Wasenmüller et al., 2016) is another benchmark dataset, recorded using the Microsoft Kinect v2. Similar to the RGB-D dataset, images are recorded at a frame rate of 30 Hz and provided in PNG format. However, the images in the CoRBS dataset are recorded at a high resolution of 1920×1080 pixels.

The ground-truth camera trajectory is obtained using an external motion capture system at a frequency of 100 Hz. The motion capture has a sub-millimetre precision.

The observed scene is less close to the situation of the NPN, however the high-resolution images allow for the evaluation of a larger range of realistic spatial scaling parameters. The dataset contains recordings of four different scenes. The camera is moved through a room while observing a static scene of a mannequin, a cabinet, a desk and or a racing car.

It was determined that only a limited number of sequences were needed for the evaluation. As the desk observing scene contains the fewest reflections, two desk observing sequences were selected for the experiments, a relatively short sequence of 23.14 s and a longer sequence of 81.3 s. In Figure 4.4 a few example frames of these sequences are shown.

4.9 Choice of parameters

The impact of spatial, temporal, and quality scaling on required throughput and tracking performance is analysed for a variety of scaling parameters, using the setup from Section 4.1.

The spatial scaling is applied using parameter $s = \frac{1}{n}$ for $n \in \mathbb{N}^+$ as long as the height of the resulting video stays above 240 pixels, as a smaller resolution are too small for a tele-operator. The spatial scaling parameters allow for an evenly divisible resolution.

The temporal scaling parameter t is varied using $t = \frac{1}{m}$ for $m \in \{1,2,3,6\}$, resulting in a frame rate of 30 Hz, 15 Hz, 10 Hz and 5 Hz. 30 Hz is the maximum available frame rate and it a minimum frame rate of 5 Hz is required for a tele-operator to navigate the robot. The values in between are chosen such that the frame rate can be scaled by discarding entire frames.

The quality is varied using the CRF values 17, 20, 23, 26 and 29, where a larger value indicates a lower quality. 23 is the default and recommended value and each increment of 6 roughly doubles the resulting file size Robitza (2017a). Hence, the minimum value applies roughly $\frac{1}{4}$ of the maximum quality scaling, which is a similar scaling range as used for the spatial scaling.

Overviews of the applied scaling parameters and resulting resolution, frame rate, and quality factor, for respectively the RGB-D dataset and the CoRBS dataset, are provided in Table 4.2 and Table 4.3.

Parameter	Scaling	Value
Resolution	1	640 × 480
Resolution	1/2	320 × 240
Frame rate	1	30 Hz
Frame rate	1/2	15 Hz
Frame rate	1/3	10 Hz
Frame rate	1/6	5 Hz
CRF	N/A	17
CRF	N/A	20
CRF	N/A	23
CRF	N/A	26
CRF	N/A	29

Table 4.2: The different scaling parameters used for the RGB-D dataset

Parameter	Scaling	Value
Resolution	1	1920 × 1080
Resolution	1/2	960 × 540
Resolution	1/3	640 × 360
Resolution	1/4	480 × 270
Frame rate	1	30 Hz
Frame rate	1/2	15 Hz
Frame rate	1/3	10 Hz
Frame rate	1/6	5 Hz
CRF	N/A	17
CRF	N/A	20
CRF	N/A	23
CRF	N/A	26
CRF	N/A	29

Table 4.3: The different scaling parameters used for the CoRBS dataset

5 Results and discussion

In the previous chapter, it was explained how the impact of spatial, temporal, and quality scaling on required throughput and visual tracking performance can be analysed qualitatively using experiments. In this chapter, the results of these experiments are presented and discussed.

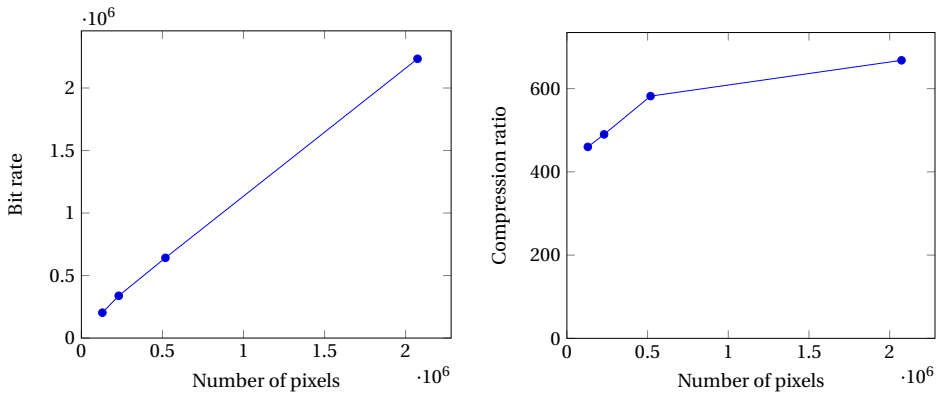
First, the results the bit rate evaluation is presented and discussed. Then, the results of the visual tracking performance evaluation are presented and discussed. Finally, it is discussed how the results can be used to define an optimal scaling strategy for the visual tracking algorithm. Subsequently, it is discussed to what extent the results apply to computer vision in general. At the end of this chapter, the key findings from the experiments are summarized.

5.1 Bit rate evaluation

For each sequence a combination of spatial, temporal and quality scaling has been applied to the image sequences, while encoding them into a video using `ffmpeg` as explained in Section 4.2. Subsequently, the bit rate has been analysed using `ffprobe`, as explained in Section 4.6.

5.1.1 Results

In Figures 5.1a–5.3a, the bit rates of different videos generated from the Desk 1 sequence of the CoRBS dataset are shown. In Figure 5.1a, it is shown how the bit rate varies depending on the resolution, in Figure 5.2a, it is shown how the bit rate varies depending on the frame rate, and in Figure 5.3a, it is shown how the bit rates varies depending on the quality. The bit rates for other sequences and scaling parameters follow a similar course and are shown in Appendix A.

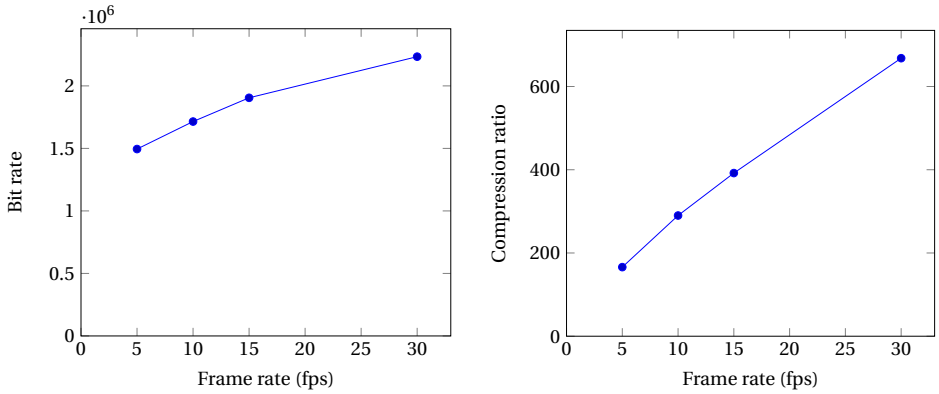


(a) The bit rate for videos with different resolutions.

(b) The compression rate for videos with different resolutions.

Figure 5.1: The bit rate and compression ratio for videos generated from the CoRBS Desk 1 sequence with different resolutions, at a frame rate of 30 Hz and a constant rate factor (CRF) of 29. The bit rate in (a) changes almost linearly with respect to the number of pixels. In (b), the compression ratio increases when the resolution is increased, which means that the relationship between the bit rate and the number of pixels is non-linear. The increase in compression ratio declines when the resolution is increased.

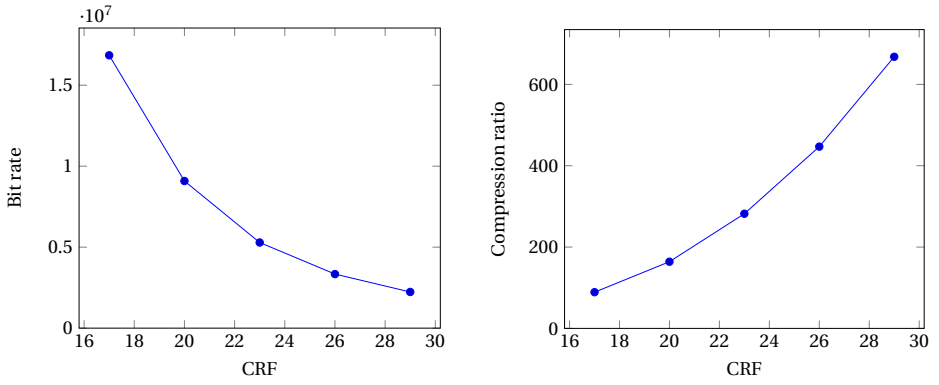
In Figure 5.1a the bit rate increases linearly with respect to the number of pixels when the resolution increases. The bit rate also increases when the frame rate increases as shown in Figure 5.2a. However, the increase in bit rate seems to decrease when the frame rate is increased. This suggests that the bit rate converges asymptotically to an upper bound when the frame rate is further increased beyond 30 Hz.



(a) The bit rate for videos with different frame rates.

(b) The compression ratio for videos with different frame rates.

Figure 5.2: The bit rate and compression ratio for videos generated from the CoRBS Desk 1 sequence with different frame rates, at a resolution of 1920 × 1080 and a CRF of 29. In (a), the bit rate increases less when the frame rate is increased. This corresponds with an increased compression ratio in (b).



(a) The bit rate for videos with different qualities, configured using the CRF.

(b) The compression ratio for videos with differnt qualities, configured using the CRF.

Figure 5.3: The bit rate and compression ratio for videos generated from the CoRBS Desk 1 sequence with different qualities, at a resolution of 1920 × 1080 and a frame rate of 30 Hz. In (a), the bit rate decreases exponentially when the quality is decreased by increasing the CRF. This corresponds with an exponentially growing compression ratio in (b).

For the quality factor, the relationship is quadratic, as shown in Figure 5.3a. When the quality is decreased, which corresponds to a higher CRF, the bit rate also decreases. However, the effect decays when the CRF is increased.

The compression applied by the encoder is shown in Figures 5.1b–5.3b for the Desk 1 sequence of the CoRBS dataset. In these figures, the compression ratio, $r_{compression}$, is defined as the ratio between the video data rate before encoding and the resulting bit rate, f_{bit}, after encoding:

$$r_{compression} = \frac{3 \cdot 8 \cdot \#pixels \cdot f_{frame}}{f_{bit}} \tag{5.1}$$

The video data rate, in Equation 5.1, is defined as the product of the data in each frame, measured as three bytes per pixel, one for each colour, and the frame rate f_{frame}. The resulting compression ratio is, hence, a dimensionless number that is equal to 1, when no compression is applied, and larger, when compression is applied to the video.

In Figure 5.1b, it is shown how the compression ratio varies depending on the resolution, in Figure 5.2b, it is shown how the compression ratio varies depending on the frame rate, and in Figure 5.3b, it is shown how the compression ratio varies depending on the quality. In all three figures, the compression ratio is larger than 1, which indicates that the encoder compresses the video data.

For both the resolution and the frame rate, the compression ratio increases as the resolution and frame rate are increased. For the resolution, this effect decays as the resolution is increased and for the frame rate this relationship is linear at least up until a frame rate of 30 Hz. When the quality is increased, which corresponds to a smaller CRF, the compression ratio decreases, as shown in Figure 5.3b.

To limit the number of figures, the bit rate and compression ratio in Figures 5.1–5.3 are shown for a limited number of scaling parameters and only for the Desk 1 sequence of the CoRBS dataset. The rest of the figures contain similar results and are presented in Appendix A.

5.1.2 Discussion of the results

From the results, it can be concluded that the bit rate increases linearly with the number of pixels, which means that the bit rate changes quadratically with the applied spatial scaling. This is in accordance with the expectations discussed in Chapter 3. The impact was expected to be influenced by the intra frame encoding that the encoder applies. The compression ratio in Figure 5.1b indeed indicates that the compression ratio increases when the resolution is increased.

For the temporal scaling, a linear relationship between frame rate and bit rate was expected. In Figure 5.2a, it is shown that this relationship is not exactly linear, but that the bit rate grows logarithmically when the frame rate is increased. This can be attributed to the inter frame encoding that the encoder applies. As increasing the frame rate results in a larger overlapping area between frames, when the camera moves with respect to the observed scene, the encoder has to encode less differences between frames. Hence, the compression ratio increases when the frame rate increases. From Figure 5.2b it can be concluded that this is indeed the case.

Encoding quality was expected to impact the bit rate as well. It was expected that each increment of 6 of the CRF would roughly halve the bit rate. From the results, it can be concluded that the relationship between the CRF and the bit rate is indeed quadratic, but that the impact on bit rate is larger for each increment of 6 of the CRF. The compression ratio increases quadratically when the CRF is increased.

Comparing the three types of scaling, it can be concluded that both quality and spatial scaling can be used to effectively reduce the required throughput of a video. The effect of tem-

poral scaling is less than that of quality and spatial scaling, because the encoder can effectively use inter frame encoding to increase the compression ratio for higher frame rates. The additional frames are relatively cheap, considering the amount of data that is required to add them. Hence, reducing the frame rate does reduce the required throughput as much as spatial and quality scaling.

5.2 Visual tracking performance evaluation

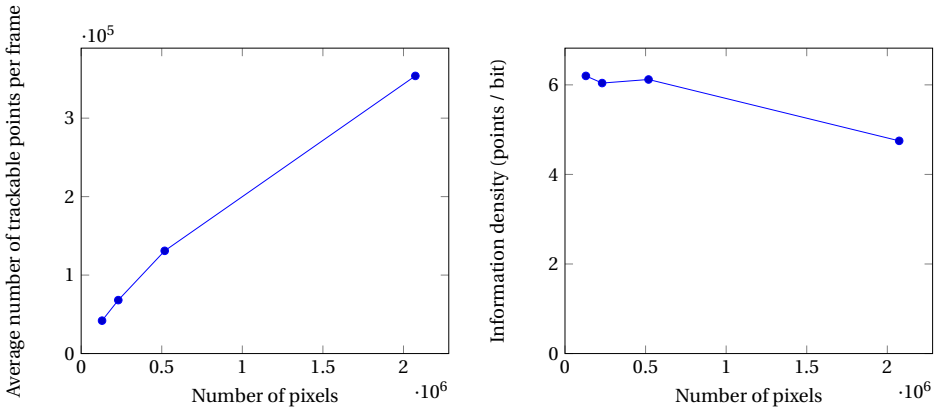
For each video, generated using `ffmpeg`, the number of trackable points has been determined and the visual tracking algorithm has been applied.

The amount of trackable points has been obtained by running the point selection algorithm on one frame every second throughout the video.

A 3D map has been generated for each video using the visual tracking algorithm. The algorithm was configured with a target minimum number of visible seeds of 200, such that it initialized another 200 seeds, using the point selection algorithm, as soon as less than 200 seeds were visible. The generated 3D map was saved during the visual tracking process, such that it can be analysed.

5.2.1 Results

The average number of points that can be selected in a frame by the point selection algorithm is shown in Figures 5.4a–5.6a for the Desk 1 sequence of the CoRBS dataset. These points have been selected based on their gradient as discussed in Chapter 3. The number can be considered as an indication of the amount of information that is present in a frame for a computer vision algorithm.

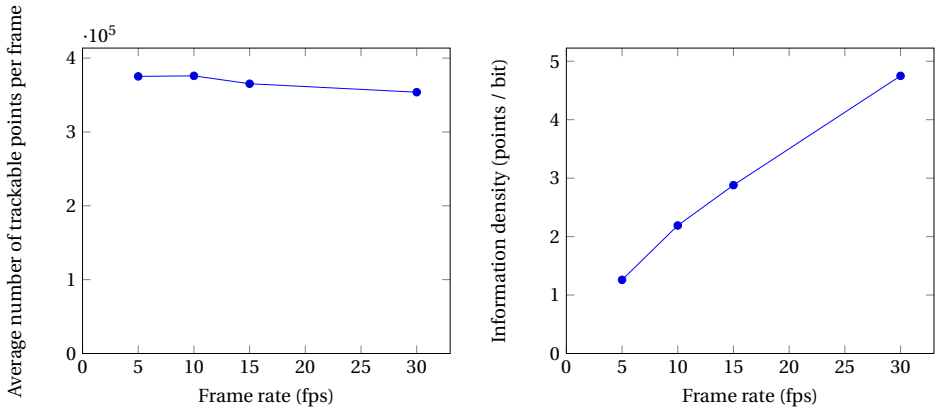


(a) The average number of trackable points per frame for videos with different resolutions.

(b) The information density of videos with different resolutions.

Figure 5.4: The average number of trackable points per frame and the information density for videos generated from the CoRBS Desk 1 sequence with different resolutions, at a frame rate of 30 Hz and a CRF of 29. The number of trackable points in (a) increases less when the resolution is increased. The resulting information density in (b) stays rather constant around lower resolutions, suggesting a linear relationship between the number of trackable points and the number of pixels. When the resolution is further increased, the information density declines.

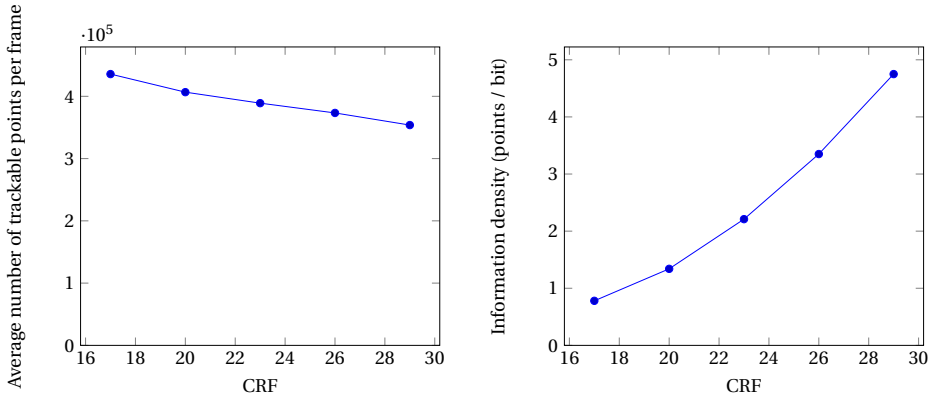
In Figure 5.4a, the number of trackable points increases when the number of pixels is increased. Each time the resolution is increased, the effect becomes smaller. This suggests that there is some maximum number of trackable points that is reached when the resolution is further increased beyond 1920 × 1080.



(a) The average number of trackable points per frame for videos with different frame rates.

(b) The information density of videos with different frame rates.

Figure 5.5: The average number of trackable points per frame and the information density for videos generated from the CoRBS Desk 1 sequence with different frame rates, at a resolution of 1920×1080 and a CRF of 29. In (b), the number of trackable points slightly decreases in a linear fashion when the frame rate is increased. As the compression ratio increases when the frame rate is increased, the information density also increases in (b) when the frame rate is increased.



(a) The average number of trackable points per frame for videos with different CRFs.

(b) The information density of videos with different qualities, configured using the CRF.

Figure 5.6: The average number of trackable points per frame and the information density for videos generated from the CoRBS Desk 1 sequence with different CRFs, at a frame rate of 30 Hz and a resolution of 1920×1080. The number of trackable points in (a) slightly decreases when the CRF is increased. As the compression ratio increases quadratically when the quality is decreased, i.e. when the CRF is increased in (b), the information density also increases quadratically when the CRF is increased.

The number of trackable points in Figure 5.5a changes linearly with respect to the frame rate. When the frame rate is increased, the number of trackable points decreases slightly. The difference between frame rates is relatively low compared to the number of trackable points. Compared to Figure 5.4a, the effect of temporal scaling on the number of trackable points is much lower than that of spatial scaling. Even though the impact of temporal scaling on the number of trackable points is relatively small, it is consistent throughout the results, as shown Appendix A.

In Figure 5.6a, there is a linear relationship between the number of trackable points and the quality of the encoding. The number of trackable points increases when the quality is increased, i.e. when the CRF is decreased.

In Figures 5.4b–5.5b, the information density of the generated videos from the CoRBS Desk 1 sequence is shown. The information density $\rho_{information}$, in these figures is defined as the ratio, between the information rate and the bit rate f_{bit}, where the information rate is defined as the product of the number of trackable points per frame and the frame rate f_{frame}:

$$\rho_{information} = \frac{\#trackablepoints \cdot f_{frame}}{f_{bit}} \tag{5.2}$$

The resulting number is expressed in number of trackable points per bit and indicates how efficient the video is compressed with respect to the information that a computer vision algorithm uses.

In Figure 5.4b, the information density increases slightly, when the resolution is increased. However, at a resolution of 1920×1080 the information density is much lower, indicating a less efficient compression with respect to the information that is useful for a computer vision algorithm.

The information density changes as a result of temporal scaling, as shown in Figure 5.5b. As opposed to the resolution, the information density increases when the frame rate is increased. This effect seems to decline slightly when the frame rate is increased. Even though the decline is relatively small, it is visible for other resolutions as well, as shown in Appendix A.

The relationship between quality and information density is quadratic in Figure 5.6b. The information density decreases when the quality increases, i.e. when the CRF is decreased.

To limit the number of figures, the number of trackable points and the information density in Figures 5.4–5.6 are shown for a limited number of scaling parameters and only for the Desk 1 sequence of the CoRBS dataset. The rest of the figures contain similar results and are presented in Appendix A.

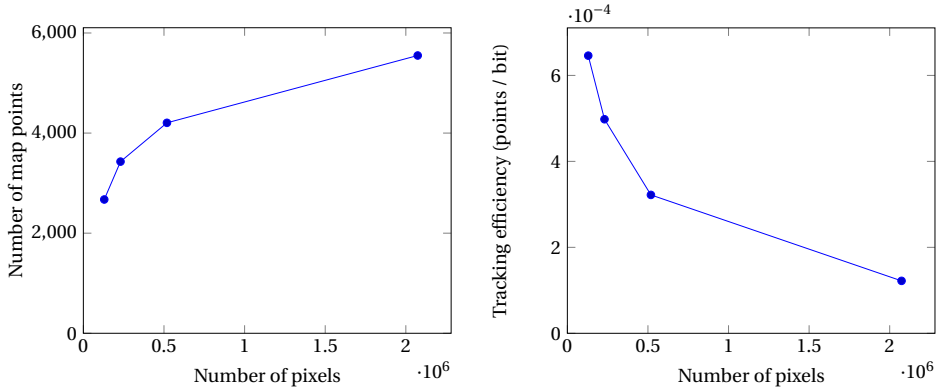
In Figures 5.7a–5.9a the number of map points, generated by the visual tracking algorithm are shown. In Figure 5.7a the number of map points increases when the resolution is increased. However, this effect decreases each time the resolution is increased.

The effect of temporal scaling on the number of map points is similar to that of spatial scaling. The number of map points increases when the frame rate is increased, as shown in Figure 5.8a. This effect decays when the frame rate is increased.

For the lowest frame rate, 5 Hz, the number of map points for the Desk 1 sequence of the CoRBS does not seem to vary with the resolution anymore, as shown in Figure 5.10. For other frame rates, the effects are similar to that of Figure 5.7a.

In Figure 5.9a there is no visible effect of quality scaling on the number of points in the generated map.

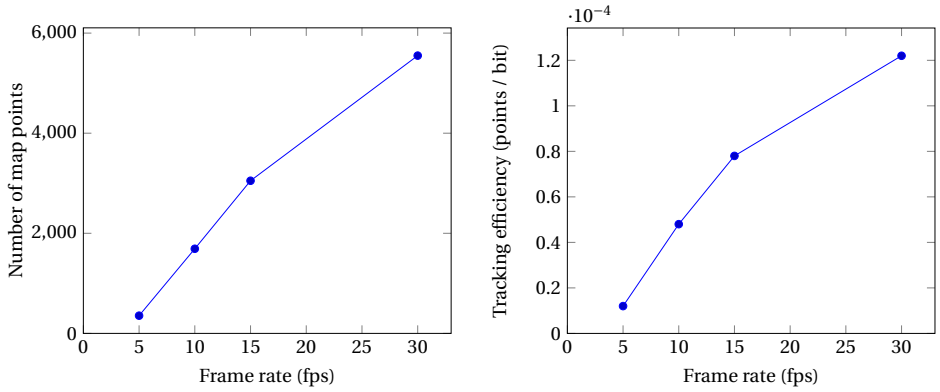
In Figure 5.7b–5.9b the tracking efficiency is shown. The tracking efficiency $\eta_{tracking}$ is defined as the ratio between the number of points that are added to the map by the visual tracking



(a) The number of map points obtained by the visual tracking algorithm for videos with different resolutions.

(b) The tracking efficiency obtained by the visual tracking algorithm for videos generated for different resolutions.

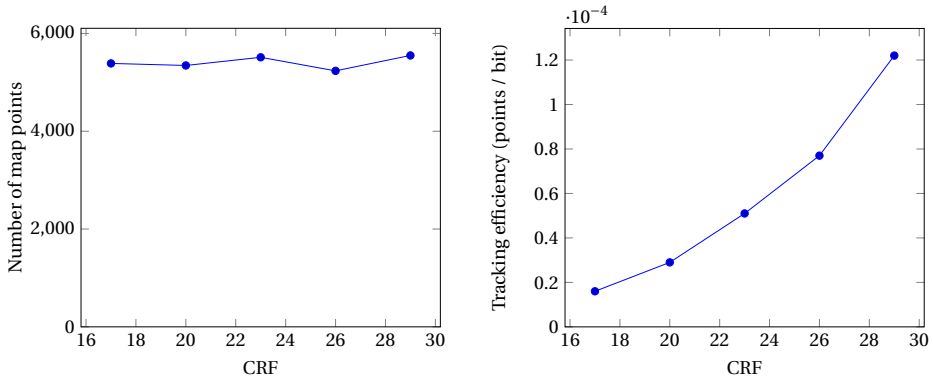
Figure 5.7: The number of map points and tracking efficiency obtained by the visual tracking algorithm for videos generated from the CoRBS Desk 1 sequence with different resolutions at a frame rate of 30 Hz and a CRF of 29. The number of map points increases in (a) when the resolution is increased. However the effect decreases when the resolution is increased, suggesting a saturation at higher resolutions. The tracking efficiency in (b) decays exponentially when the resolution is increased.



(a) The number of map points obtained by the visual tracking algorithm for videos with different frame rates.

(b) The tracking efficiency obtained by the visual tracking algorithm for videos with different frame rates.

Figure 5.8: The number of map points and the tracking efficiency obtained by the visual tracking algorithm for videos generated from the CoRBS Desk 1 sequence with different frame rates at a resolution of 1920×1080 and a CRF of 29. In (a) the number of map points increases when the frame rate is increased. Similar to the effect of resolution, the increment in the number of map points decreases when the frame rate is increased. However, this effect is less strong than for the spatial scaling. Furthermore, since the compression ratio decreases when the frame rate is increased, the tracking efficiency in (b) also increases when the frame rate is increased.



(a) The number of map points obtained by the visual tracking algorithm for videos with different qualities, configured using the CRF

(b) The tracking efficiency obtained by the visual tracking algorithm for videos with different qualities, configured using the CRF.

Figure 5.9: The number of map points and the tracking efficiency obtained by the visual tracking algorithm for videos generated from the CoRBS Desk 1 sequence with different CRFs at a frame rate of 30 Hz and a resolution of 1920 × 1080. The number of map points in (a) does not consistently change when the CRF is changed. As the compression ratio increases when the quality is decreased by increasing the CRF, the tracking efficiency in (5.9b) also increases when the quality is decreased.

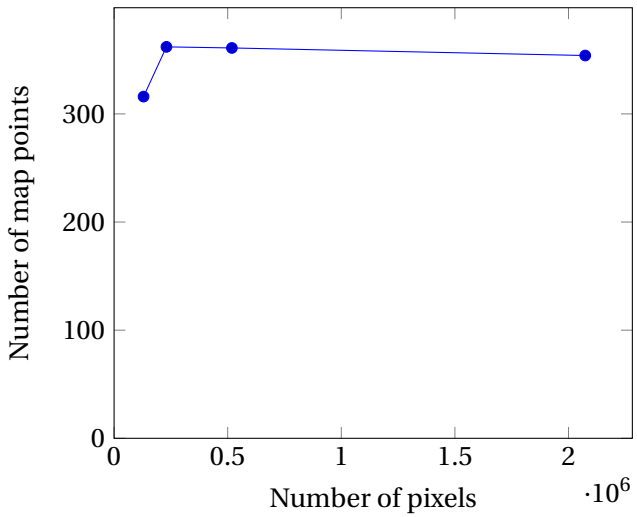


Figure 5.10: The number of map points obtained by the visual tracking algorithm for videos generated from the CoRBS Desk 1 sequence with different resolutions at a frame rate of 5 Hz and a CRF of 29. Compared to Figure 5.7a the number of map points does not change consistently when the resolution is changed.

algorithm and the data that was required to obtain these points, i.e. the size of the encoded video:

$$\eta_{tracking} = \frac{\#mappoints}{f_{bit} \cdot duration} \tag{5.3}$$

In Figure 5.7b the tracking efficiency is shown for the Desk 1 sequence of the CoRBS dataset with different resolutions. The efficiency decreases exponentially when the resolution is increased.

As shown in Figure 5.8b, the reverse is true for the frame rate: the efficiency increases when the frame rate is increased. This effect decreases each time the frame rate is increased, suggesting a maximum efficiency when the frame rate is further increased beyond 30 Hz.

In Figure 5.9b the tracking efficiency increases when the quality of the encoding is decreased, i.e. when the CRF is increased. This relationship appears to be quadratic.

To limit the number of figures, the number of map points and tracking efficiency in Figures 5.7–5.9 are shown for a limited number of scaling parameters and only for the Desk 1 sequence of the CoRBS dataset. The rest of the figures contain similar results and are presented in Appendix A.

5.2.2 Discussion of the results

From the results it can be concluded that the number of trackable points increases when the resolution of the video is increased. This is in agreement with the expectation. As there are more pixels available when the resolution is increased, there are more points that can be tracked.

However, as discussed in Chapter 3, the information density decreases when the resolution increases, as shown in Figure 5.4b. This means that, when more pixels are added to the frames of the video, by increasing the resolution, there are increasingly more pixels added that are not trackable. Hence, the number of trackable points in Figure 5.4a increases less each time the resolution is increased.

It was not expected that the number of trackable points would change depending on the frame rate. The frame rate should not affect the pixels in a single frame. However, from Figure 5.5a it can be concluded that the number of trackable points slightly decreases when the frame rate is increased. An explanation for this phenomenon is that the encoder applies more compression when the frame rate is increased. The increased compression means that less values are used in the discrete cosine transform, resulting in the removal of the higher frequency parts. This low-pass filtering reduces the gradients, as explained in Chapter 3. Since points are selected based on their gradient, less points will have the required gradient value to be selected as a trackable point.

The small reduction in available trackable points does not result in a lower information density. As shown in Figure 5.5b, the information density increases because the frame rate can be increased using relatively little data, due to inter frame encoding.

The effect of quality scaling on the number of trackable points is comparable to that of temporal scaling. Decreasing the quality of the encoding increases the compression, such that less points can be selected for tracking, as shown in Figure 5.6a. The effect of quality scaling on the number of trackable points is larger than that of temporal scaling, which is in accordance with the expectations.

The information density increases quadratically when the quality is decreased, as shown in Figure 5.6a. The reason for this is that the impact of quality scaling on required throughput is larger than the impact on the number of trackable points.

Regarding the number of map points, it can be concluded from 5.7a that increasing the resolution increases the number of points that the visual tracking algorithm can add to the generated map. An explanation for this is that a variance of 1 pixel is used to model the uncertainty of the matched pixel. The uncertainty in the depth estimates decreases when the resolution is increased, because the uncertainty in pixels then corresponds with a smaller uncertainty in the depth estimate. Hence, the points can be added to the map sooner. This is in accordance with the expectations. The relationship between the number of map points and the number of pixels is logarithmic. This is due to the fact that the uncertainty, corresponding to the variance of 1 pixel in any direction, decreases with the square root of the number of additional pixels in a frame.

The tracking efficiency decreases when the resolution is increased, as shown in Figure 5.7b. This is because the number of map points increases logarithmically with respect to the number of pixels, whereas the bit rate increases linearly with the number of pixels when the resolution is increased.

Increasing the frame rate increases the number of points that are added to the map by the visual tracking algorithm as well, as shown in Figure 5.8a. This is in accordance with the expectations. It was expected that the uncertainty would decrease, because there are more measurements per tracked point when the frame rate is increased. The impact of temporal scaling decreases, when the frame rate is increased. It is expected that the relation is bounded by the target number of seeds and the number of frames.

Compared to the resolution, the frame rate has a larger impact on the number of map points than the resolution. Furthermore, can be concluded that increasing the frame rate is relatively cheap considering the required throughput, because the tracking efficiency increases when the frame rate is increased, as shown in Figure 5.8b.

In Chapter 3, it was explained that there exists a lower bound for the frame rate. After reaching this lower bound, increasing the resolution does not increase the tracking performance anymore. From Figure 5.10, it can be concluded that this lower bound is reached at a frame rate of 5 Hz for the Desk 1 sequence of the CoRBS dataset.

Regarding the quality, it can be concluded that, at least for a target number of seeds of 200 and a CRF in the range of 17–29, decreasing quality does not affect tracking performance, because, as shown in Figure 5.9a, reducing the quality does not result in a significant decrease of the number of map points. This is not in accordance with the expectations. It was expected that decreasing the quality would have a negative a negative impact on visual tracking performance and hence reduce the number of map points.

Considering the increased tracking efficiency for this CRF, shown in Figure 5.9b, data is used most efficiently, using a CRF 29. However, there are two reasons for using a higher quality and hence a lower CRF. First of all, in Figure 5.6a it is shown that a higher quality allows for tracking slightly more points. Second, it increases the perceived quality of service for the tele-operator.

From the results it can be concluded that the quality can be further decreased than the minimum value used for the experiments, because the no significant impact on tracking performance. It is, however, expected that the quality cannot be decreased indefinitely. At some point it will affect tracking performance. Additional experiments with larger CRFs are required to determine the maximum CRF.

When the three types of scaling are compared, it can be concluded that the CRFs used for the experiments do not affect tracking performance. Increasing frame rate and resolution both increase the number of points in the generated map. However, the impact of frame rate is larger than that of resolution. Furthermore, increasing the frame rate also increases the information density and the tracking efficiency, whereas increasing the resolution decreases the information density and tracking efficiency. This indicates that it is better to use a high frame rate and

to apply spatial scaling before applying temporal scaling. However, the number of trackable points strongly depends on the resolution. Applying spatial scaling therefore comes at the cost of map density.

5.3 Optimal scaling

Using the results of Section 5.1 and Section 5.2, an optimal strategy can be derived for reducing the required throughput of the video stream, while ensuring high-performance visual tracking. Such a strategy can be derived by applying the following steps:

1. Create a table with the scaling parameters, resulting bit rates, and number of map points for each generated video.
2. Sort the table by the number of map points in descending order, because the strategy should optimize for the number of map points.
3. Remove all entries that have a higher bit rate than the entry above it, because these entries have a lower tracking performance, but require more data than the entry above them. Hence, switching to such a configuration is not optimal.

The optimal strategies for all sequences, obtained by applying these steps, are shown in Tables 5.1–5.5. For these tables, only scaling configurations with a CRF of 29 were taken into consideration, as it was concluded that using a higher quality does not improve tracking performance, but does increase the throughput requirements of the video.

Resolution	Frame rate (fps)	Quality (CRF)	Bit rate (bps)	# map points
1920x1080	30	29	2233689	5550
960x540	30	29	641595	4205
640x360	30	29	338207	3430
480x270	30	29	203055	2673
480x270	15	29	177706	1967
480x270	10	29	160651	1254
480x270	5	29	143580	316

Table 5.1: The configurations from the optimal strategy for the Desk 1 sequence of the CoRBS dataset.

Resolution	Frame rate (fps)	Quality (CRF)	Bit rate (bps)	# map points
1920x1080	30	29	2100282	17261
1920x1080	15	29	1867010	12508
960x540	30	29	603820	11837
640x360	30	29	315333	9990
480x270	30	29	189966	8074
480x270	15	29	163588	6368
480x270	10	29	151608	5438
480x270	5	29	138214	3669

Table 5.2: The configurations from the optimal strategy for the Desk 1 sequence of the CoRBS dataset.

All results, except for Table 5.2, indicate that the optimal strategy is to apply spatial scaling before temporal scaling, until the lowest resolution, respectively 480×270 and 320×240, is reached. After reaching the lowest resolution, temporal scaling can be used to further reduce the required throughput.

Since a high quality is only beneficial for a tele-operator, and does not affect tracking performance, quality scaling can be applied before spatial scaling, as the first measure to reduce throughput.

Resolution	Frame rate (fps)	Quality (CRF)	Bit rate (bps)	# map points
640x480	30	29	401542	9031
320x240	30	29	113110	4939
320x240	15	29	98965	1057
320x240	10	29	93658	220
320x240	5	29	85384	40

Table 5.3: The configurations from the optimal strategy for the Pioneer robot sequence 1 of the RGB-D dataset.

Resolution	Frame rate (fps)	Quality (CRF)	Bit rate (bps)	# map points
640x480	30	29	365175	5165
320x240	30	29	104972	3176
320x240	15	29	89682	1432
320x240	10	29	83902	426
320x240	5	29	76913	74

Table 5.4: The configurations from the optimal strategy for the Pioneer robot sequence 2 of the RGB-D dataset.

Resolution	Frame rate (fps)	Quality (CRF)	Bit rate (bps)	# map points
640x480	30	29	419008	5619
320x240	30	29	110357	2806
320x240	15	29	94978	1235
320x240	10	29	88802	419
320x240	5	29	81932	17

Table 5.5: The configurations from the optimal strategy for the Pioneer robot sequence 3 of the RGB-D dataset.

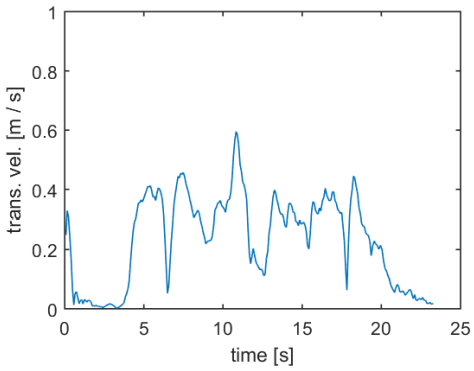
In summary, the optimal scaling strategy is:

1. Apply quality scaling until the minimum quality is reached.
2. Apply spatial scaling until the minimum resolution is reached.
3. Apply temporal scaling until the minimum frame rate is reached.

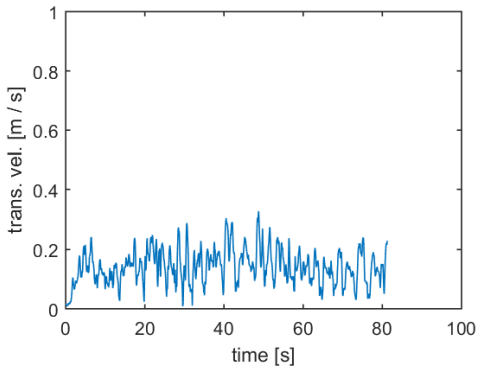
The scaling should be applied until the required throughput is lower than the available throughput, such that the available throughput does not prevent successful transmission of all video data. The required throughput for the different sequences is roughly indicated by the bit rate in Tables 5.1–5.5. As explained in Chapter 3, the actual required throughput is larger due to data that is added by the network protocols. However, the minimum bit rates are more than an order of magnitude smaller than the minimum available throughput of 6.5 Mbps that was presented in Chapter 3. This should be low enough to successfully transmit all video data.

For any given configuration, it can be seen that the bit rate in Tables 5.1–5.5 differs between sequences. Hence, it is not possible to define an optimal scaling strategy for specific values of the throughput for videos in general. More research is required to determine the distribution of bit rate for all scaling parameters for a larger number of videos. After determining how the bit rate is distributed, the optimal scaling strategy can be combined with the work of Capirchio (2017) and implemented and tested on an actual robotic system, that streams video via wireless connection.

The optimal strategy for the Desk 2 sequence of the CoRBS dataset, shown in Table 5.2, differs from the other results. For this sequence it is beneficial to reduce the frame rate, earlier in the process. A possible explanation for this is shown in Figure 5.11. The average velocity in the Desk 2 sequence is low compared to that in the Desk 1 sequence. As predicted in 3, the effect of temporal scaling is larger when the camera moves with a higher velocity with respect to the observed scene. The findings hence confirm that an optimal strategy scaling strategy depends on the velocity of the robots. Slower moving robots can apply more temporal scaling than faster moving robots. The threshold seems to be around an average velocity of 0.2 m/s.



(a) The velocity of the camera during the Desk 1 sequence.



(b) The velocity of the camera during the Desk 2 sequence.

Figure 5.11: The velocities of the camera for the CoRBS dataset (Wasenmüller et al., 2016).

The robots used by the Dutch National Police (NPN) are moving relatively fast with respect to the observed scene, for example, while chasing a person. Hence, the larger velocity of the Desk 1 sequence is a better representation of the situation of the NPN. Therefore, the robots of the NPN should be configured to apply spatial scaling before temporal scaling.

5.4 Limitations and applicability to computer vision in general

In the previous sections, the results of the experiments were discussed and an optimal strategy was derived for each sequence. It was concluded that, the preferred scaling order is quality scaling, followed by spatial scaling, and then temporal scaling.

The results, however, were not obtained using an actual robot with a wireless video stream. Further research is needed to establish whether the optimal strategy prevents streaming artefacts in wireless video streams in practice.

Another limitation of the research is that the visual tracking algorithm used for the experiments, tracked relatively few seeds in the videos. Therefore, it is expected that the results do not apply to computer vision algorithms that require a relatively large number of tracked points. It is expected that the effect of spatial scaling increases when more points must be tracked, because spatial scaling reduces the number of trackable points.

Furthermore, the performance of the tracking algorithm was evaluated by analysing the number of points that were added to the map. This number indicates the uncertainty of the visual tracking, but not its accuracy. The impact of spatial, temporal, and quality scaling on the accuracy of 3D reconstructions was not evaluated.

Also, the tracking algorithm used for the experiments, selects the best trackable points. For points that are less trackable, i.e., points with a lower gradient, the impact of quality scaling is expected to be larger. Furthermore, the algorithm is relatively robust to quality scaling, because it minimizes the photometric error and does not track specific features. Computer vision algorithms that need to track less trackable points or complex features, may require higher quality video encoding.

In general, it is expected that the results of this thesis apply to a wide variety of computer vision algorithms, because the algorithm used for the experiments in this thesis is based on basic image processing techniques, that are used as a basis for most computer vision algorithms, such as the direct visual simultaneous localization and mapping (SLAM) methods.

5.5 Summary

Experiments have been conducted to quantify the impact of spatial, temporal, and quality scaling on bit rate and visual tracking performance. In summary, the key findings are:

- Spatial, temporal, and quality scaling effectively reduce the data.
- The compression ratio increases when either the frame rate or resolution is increased, or when the quality is decreased. This effect is smaller for the resolution than for the frame rate, which indicates that the encoder can apply more inter frame encoding than intra frame encoding.
- The number of trackable points increases when the resolution is increased, decreases slightly when the frame rate is increased, and decreases slightly more when the quality is decreased.
- The visual tracking performance increases when resolution is increased. However, the tracking efficiency decreases when the resolution is increased. Increasing the frame rate increases both the visual tracking performance and the tracking efficiency. Furthermore, the visual tracking algorithm is more sensitive to temporal scaling than spatial scaling. Quality scaling does not seem to affect the visual tracking performance.
- Based on the results, there exists an optimal scaling strategy that reduces required throughput, while maximizing tracking performance. The optimal strategy is to apply quality scaling first, until the lowest quality is reached, followed by spatial scaling, until the lowest resolution is reached, and finally temporal scaling to further reduce the required throughput.

6 Conclusions and recommendations

The goal of this research was to determine how to optimize wireless video streams for computer vision. The research question was divided into two sub questions, that respectively considered reducing the video data and its effects on computer vision. During the research, the bit rate and tracking performance of videos from multiple sequences of the CoRBS and RGB-D datasets, have been evaluated after applying spatial, temporal, and quality scaling using different parameters. Based on these evaluations, it can be concluded that there exists a clear optimal video scaling strategy, that reduces the required throughput, while maximizing visual tracking performance.

According to this strategy, quality scaling is the first measure that should be taken to reduce the required throughput of a video stream, as it has been concluded that quality scaling does not affect tracking performance, but does reduce the required throughput.

When the minimum quality is reached, spatial scaling can be used to further reduce the required throughput of a video. While this reduces the number of trackable points, it reduces the required throughput even more.

Finally, when the lowest resolution is reached as well, temporal scaling can be applied to reduce the required throughput even further.

It can be concluded that the throughput reduction that is realized using the optimal strategy is large enough to allow for successful transmission of all video data.

While the research has only considered the performance of a visual tracking algorithm, using a limited number of seeds, it is expected that the results are generalizable to a wider range of computer vision algorithms, because the visual tracking algorithm used for the experiments is based on basic image processing techniques that form the basis of most computer vision tasks.

Compared to the human visual system, it can be concluded that computer vision perceives quality of service differently. Scaling the encoding quality of a video does not seem to affect computer vision algorithm, whereas it does affect the perceived quality of service for the human visual system.

The results of this thesis can be used to configure throughput-based, adaptive, wireless video streams for mobile robots. Using related research, comprising the design of a system for live-captured video streaming over the data distribution service (DDS) (Capirchio, 2017), a system can be constructed that enables reliable video streaming for the different robots used by the Dutch National Police (NPN) with plug-and-play functionality. Optimizing the video stream for computer vision, enables a variety of tasks, such as the creation of a 3D map to aid the tele-operator during navigation.

As evaluations have been performed offline, on videos that were captured by cameras while moving through static environments. The efficacy of the optimal strategy still has to be verified on an actual robotic system, that streams video over an actual wireless connection while observing more realistic scenarios, such as the chase of a suspect.

Some of the findings of this research can be further analysed during future research. The three main recommendations are:

- *An analysis of the bit rate of videos captured by the NPN.* The strategy that was extracted from the results in this thesis, defines how scaling can be applied to reduce throughput while maximizing visual tracking performance. In practice, the resulting throughput of a video varies, depending on its content. To determine the optimal scaling configuration,

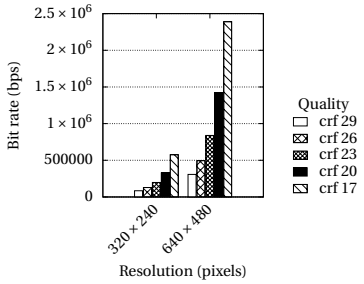
based on the available throughput, it is required to understand which bit rates can be expected as a result of scaling for a wide variety of videos captured by the NPN.

- *An analysis of the efficacy of the optimal strategy on an actual robot.* As stated before, the results of this thesis can be combined with those from related research to construct a system that enables reliable video streaming. Capirchio (2017) analysed how available throughput changes depending on external disturbances. Together with the required throughputs presented in this thesis, a system for reliable video streaming can be implemented and applied to an actual robot, while observing a more realistic scenario, such that the efficacy of the proposed strategy can be verified in a realistic situation.

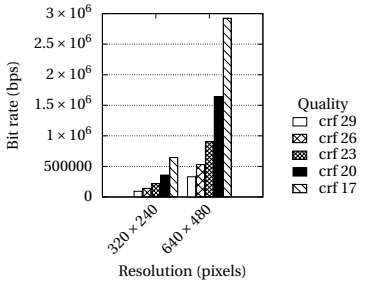
- *An analysis of the accuracy of the generated map.* The visual tracking algorithm that was used for the experiments, used only a limited number of seeds. Furthermore, the performance was evaluated using the number of points in the map, which is a measure of the uncertainty of the tracking algorithm. The impact of spatial scaling on tracking performance is expected to be larger when more points are used. It is recommended to analyse the impact of spatial, temporal, and quality scaling on the accuracy of the generated map, when dense, accurate 3D reconstructions need to be created by a computer vision algorithm, using a larger number of seeds.

A Measurement results

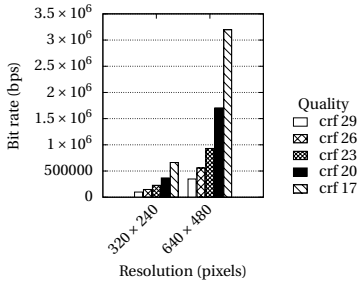
This appendix contains all measurement results, that were not presented in Chapter 5) for readability. The results are shown in Figures A.1–A.30.



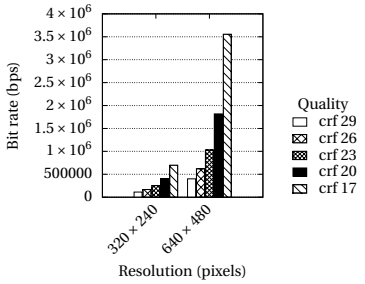
(a) The bit rate at a frame rate of 5 Hz.



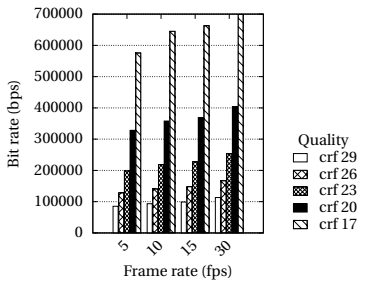
(b) The bit rate at a frame rate of 10 Hz.



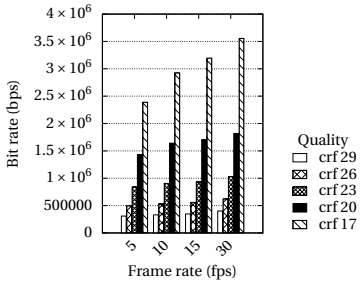
(c) The bit rate at a frame rate of 15 Hz.



(d) The bit rate at a frame rate of 30 Hz.

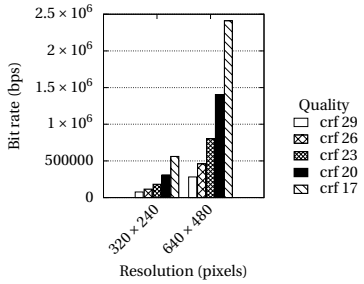


(e) The bit rate at a resolution of 320 × 240.

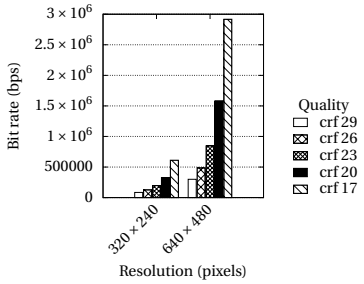


(f) The bit rate at a resolution of 640 × 480.

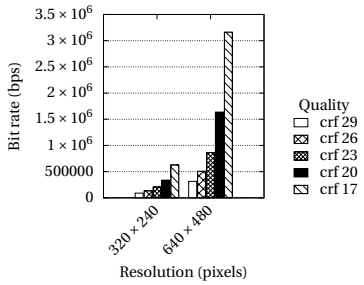
Figure A.1: The bit rate for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the bit rate for all constant rate factor (CRF) factors. (a) – (d) show the bit rates for each resolution at a fixed frame rate configuration. (e) – (f) show how the bit rates vary for a constant resolution.



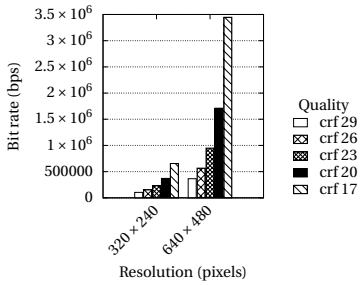
(a) The bit rate at a frame rate of 5 Hz.



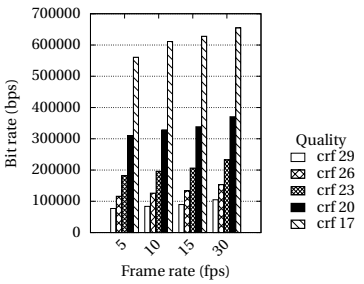
(b) The bit rate at a frame rate of 10 Hz.



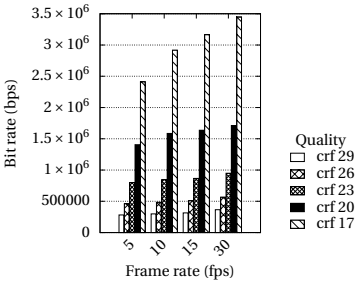
(c) The bit rate at a frame rate of 15 Hz.



(d) The bit rate at a frame rate of 30 Hz.

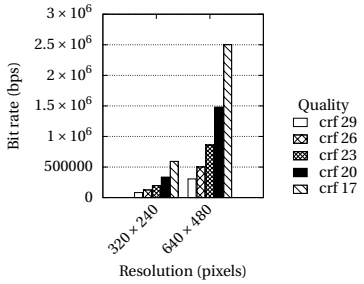


(e) The bit rate at a resolution of 320 × 240.

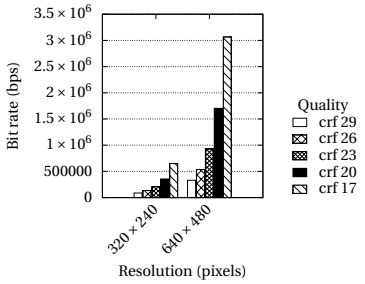


(f) The bit rate at a resolution of 640 × 480.

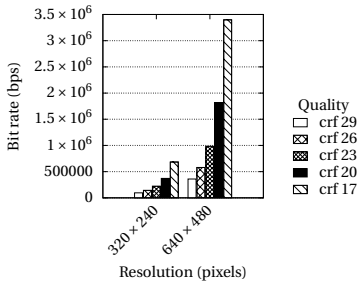
Figure A.2: The bit rate for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the bit rate for all CRF factors. (a) – (d) show the bit rates for each resolution at a fixed frame rate configuration. (e) – (f) show how the bit rates vary for a constant resolution.



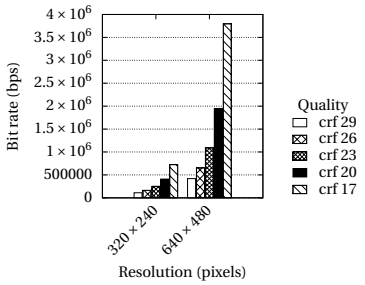
(a) The bit rate at a frame rate of 5 Hz.



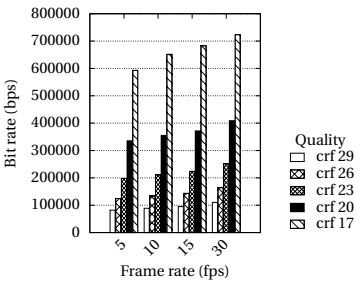
(b) The bit rate at a frame rate of 10 Hz.



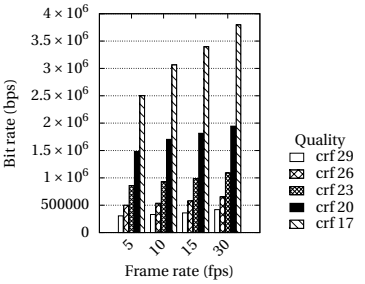
(c) The bit rate at a frame rate of 15 Hz.



(d) The bit rate at a frame rate of 30 Hz.

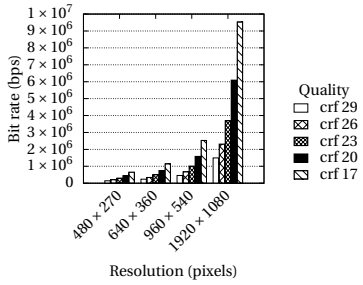


(e) The bit rate at a resolution of 320 × 240.

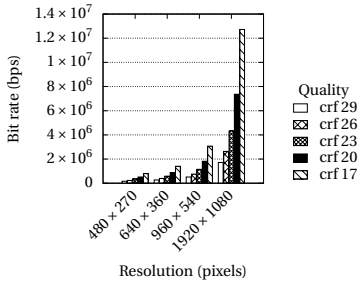


(f) The bit rate at a resolution of 640 × 480.

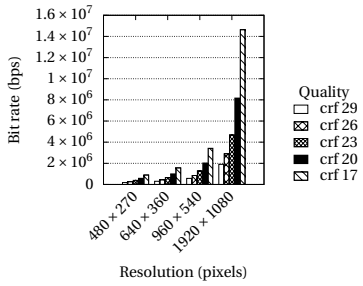
Figure A.3: The bit rate for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the bit rate for all CRF factors. (a) – (d) show the bit rates for each resolution at a fixed frame rate configuration. (e) – (f) show how the bit rates vary for a constant resolution.



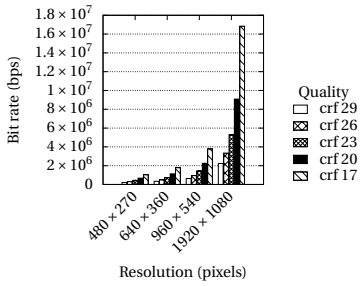
(a) The bit rate at a frame rate of 5 Hz.



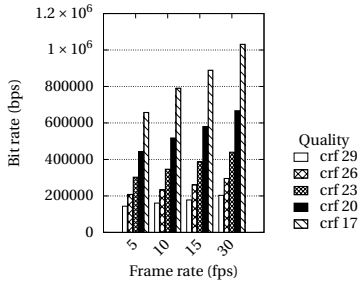
(b) The bit rate at a frame rate of 10 Hz.



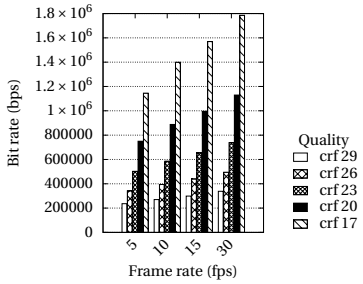
(c) The bit rate at a frame rate of 15 Hz.



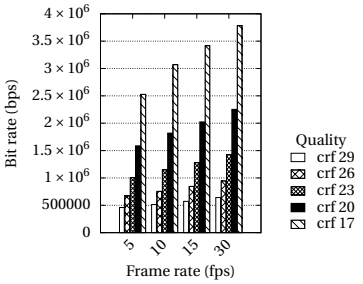
(d) The bit rate at a frame rate of 30 Hz.



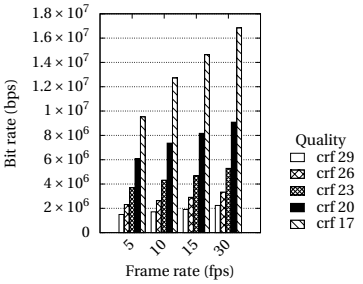
(e) The bit rate at a resolution of 480 × 270.



(f) The bit rate at a resolution of 640 × 360.

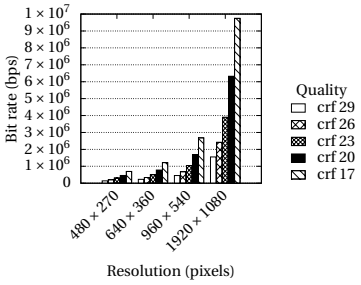


(g) The bit rate at a resolution of 960 × 540.

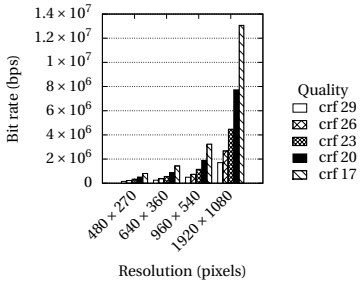


(h) The bit rate at a resolution of 1920 × 1080.

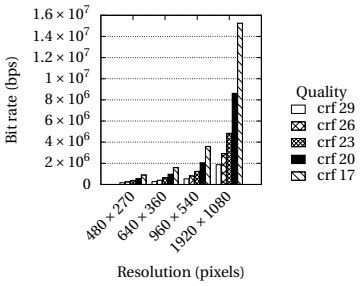
Figure A.4: The bit rate for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the bit rate for all CRF factors. (a) – (d) show the bit rates for each resolution at a fixed frame rate configuration. (e) – (h) show how the bit rates vary for a constant resolution.



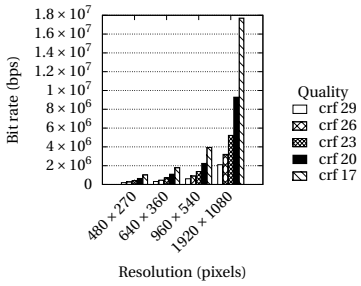
(a) The bit rate at a frame rate of 5 Hz.



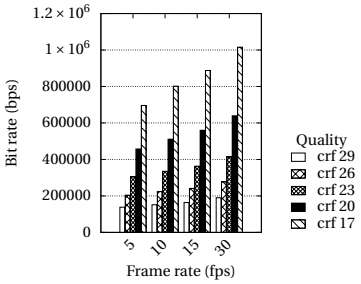
(b) The bit rate at a frame rate of 10 Hz.



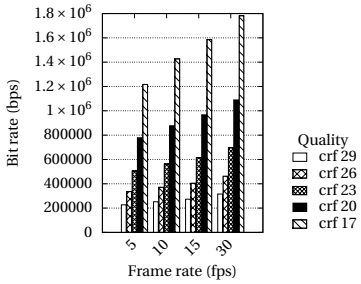
(c) The bit rate at a frame rate of 15 Hz.



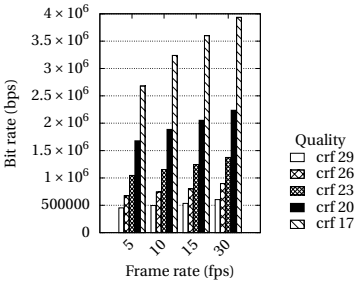
(d) The bit rate at a frame rate of 30 Hz.



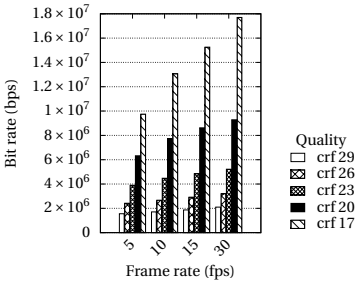
(e) The bit rate at a resolution of 480 × 270.



(f) The bit rate at a resolution of 640 × 360.

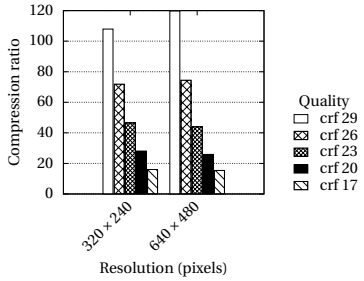


(g) The bit rate at a resolution of 960 × 540.

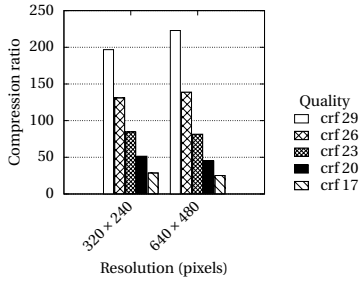


(h) The bit rate at a resolution of 1920 × 1080.

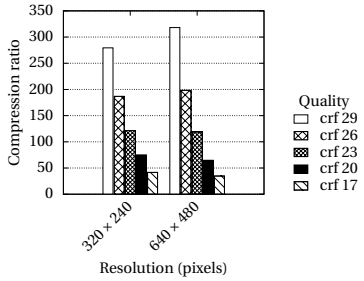
Figure A.5: The bit rate for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the bit rate for all CRF factors. (a) – (d) show the bit rates for each resolution at a fixed frame rate configuration. (e) – (h) show how the bit rates vary for a constant resolution.



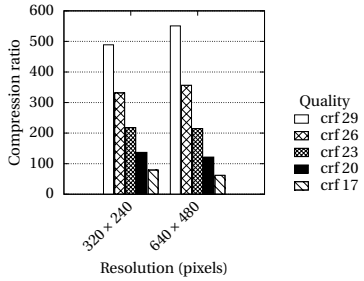
(a) The compression ratio at a frame rate of 5 Hz.



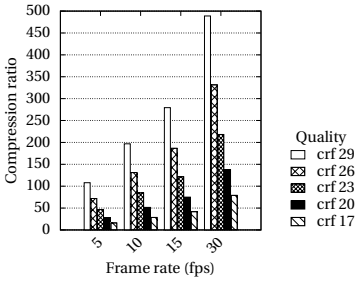
(b) The compression ratio at a frame rate of 10 Hz.



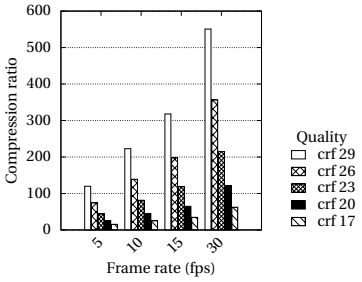
(c) The compression ratio at a frame rate of 15 Hz.



(d) The compression ratio at a frame rate of 30 Hz.

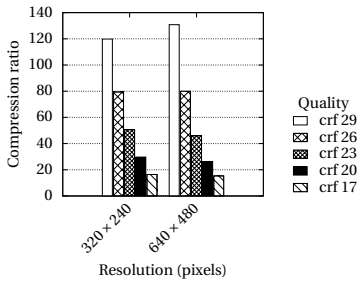


(e) The compression ratio at a resolution of 320 × 240.

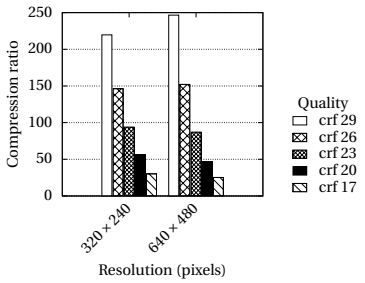


(f) The compression ratio at a resolution of 640 × 480.

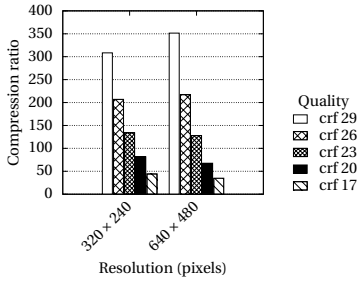
Figure A.6: The compression ratio for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the compression ratio for all CRF factors. (a) – (d) show the compression ratios for each resolution at a fixed frame rate configuration. (e) – (f) show how the compression ratios vary for a constant resolution.



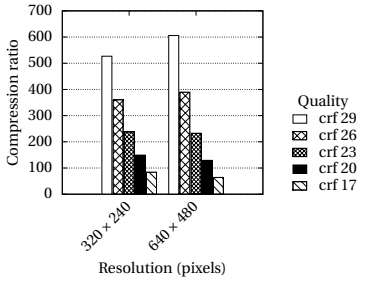
(a) The compression ratio at a frame rate of 5 Hz.



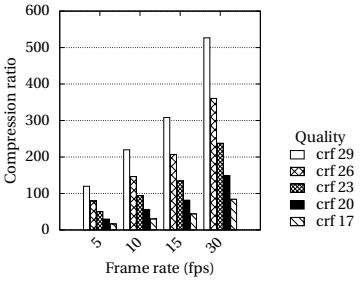
(b) The compression ratio at a frame rate of 10 Hz.



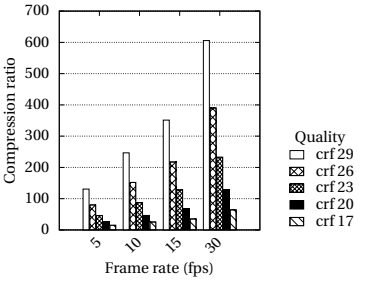
(c) The compression ratio at a frame rate of 15 Hz.



(d) The compression ratio at a frame rate of 30 Hz.

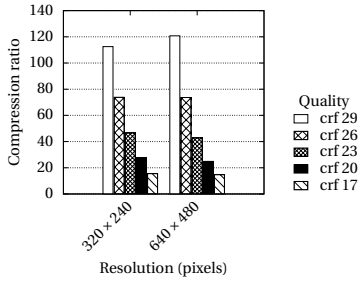


(e) The compression ratio at a resolution of 320 × 240.

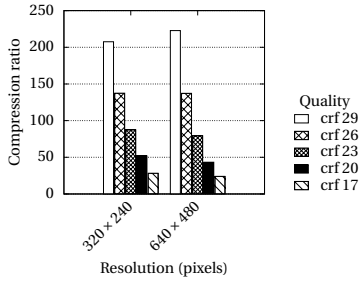


(f) The compression ratio at a resolution of 640 × 480.

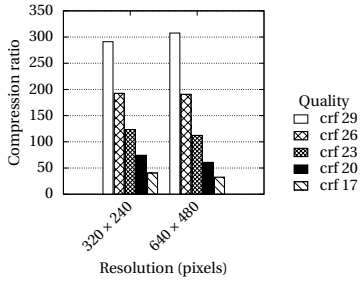
Figure A.7: The compression ratio for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the compression ratio for all CRF factors. (a) – (d) show the compression ratios for each resolution at a fixed frame rate configuration. (e) – (f) show how the compression ratios vary for a constant resolution.



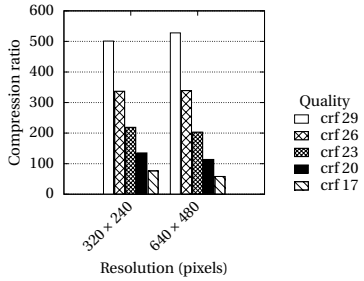
(a) The compression ratio at a frame rate of 5 Hz.



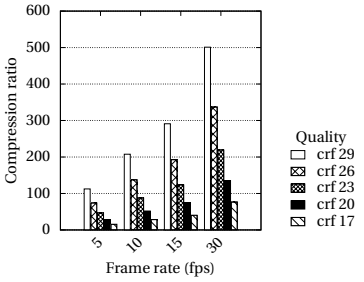
(b) The compression ratio at a frame rate of 10 Hz.



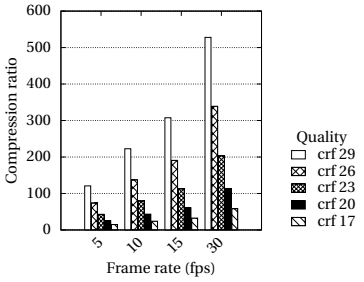
(c) The compression ratio at a frame rate of 15 Hz.



(d) The compression ratio at a frame rate of 30 Hz.

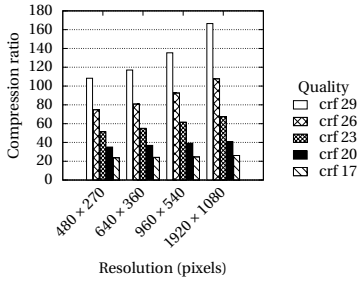


(e) The compression ratio at a resolution of 320 × 240.

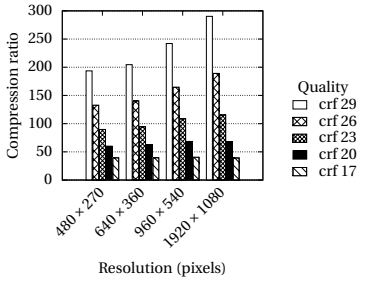


(f) The compression ratio at a resolution of 640 × 480.

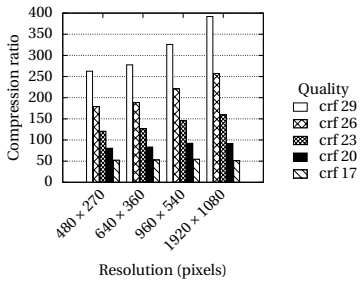
Figure A.8: The compression ratio for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the compression ratio for all CRF factors. (a) – (d) show the compression ratios for each resolution at a fixed frame rate configuration. (e) – (f) show how the compression ratios vary for a constant resolution.



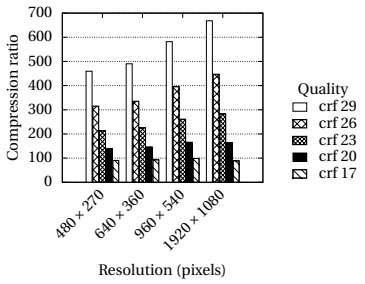
(a) The compression ratio at a frame rate of 5 Hz.



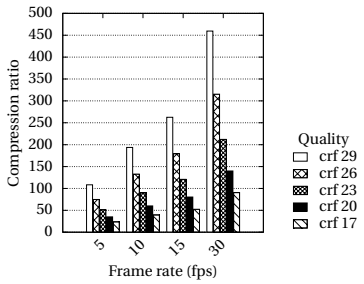
(b) The compression ratio at a frame rate of 10 Hz.



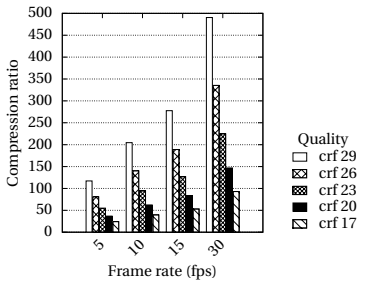
(c) The compression ratio at a frame rate of 15 Hz.



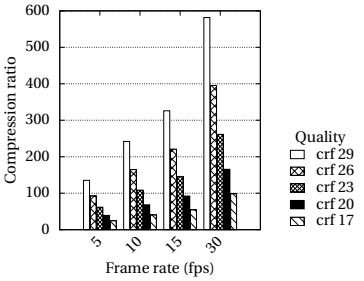
(d) The compression ratio at a frame rate of 30 Hz.



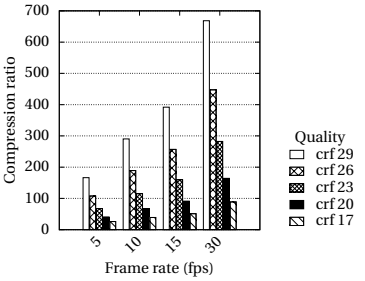
(e) The compression ratio at a resolution of 480 × 270.



(f) The compression ratio at a resolution of 640 × 360.

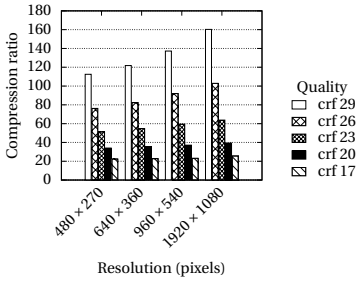


(g) The compression ratio at a resolution of 960 × 540.

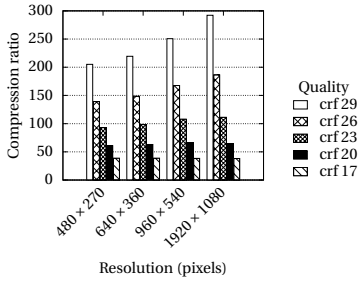


(h) The compression ratio at a resolution of 1920 × 1080.

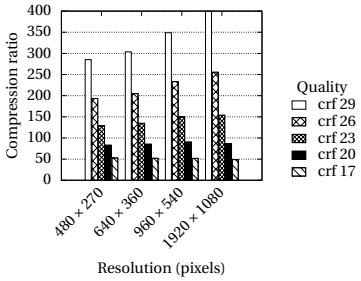
Figure A.9: The compression ratio for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the compression ratio for all CRF factors. (a) – (d) show the compression ratios for each resolution at a fixed frame rate configuration. (e) – (h) show how the compression ratios vary for a constant resolution.



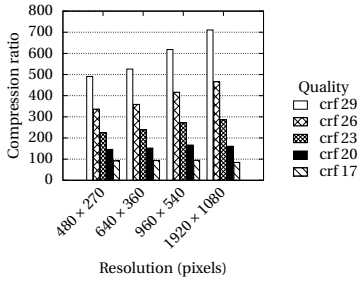
(a) The compression ratio at a frame rate of 5 Hz.



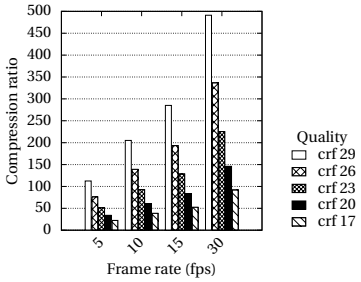
(b) The compression ratio at a frame rate of 10 Hz.



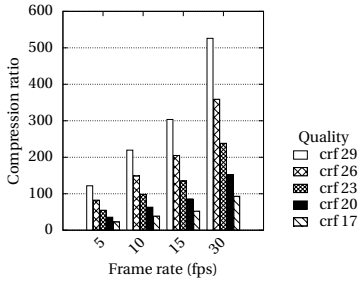
(c) The compression ratio at a frame rate of 15 Hz.



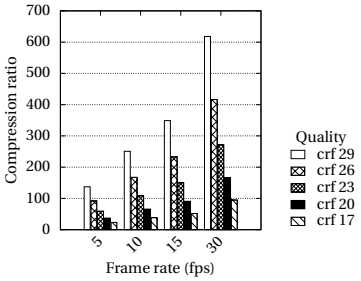
(d) The compression ratio at a frame rate of 30 Hz.



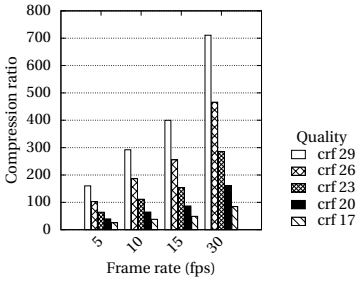
(e) The compression ratio at a resolution of 480 × 270.



(f) The compression ratio at a resolution of 640 × 360.

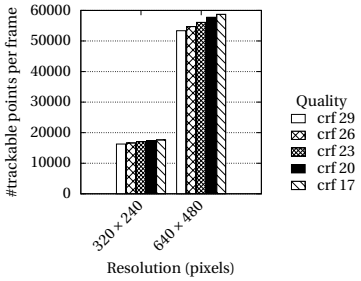


(g) The compression ratio at a resolution of 960 × 540.

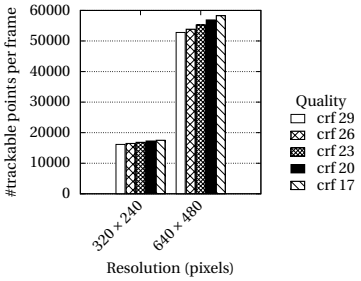


(h) The compression ratio at a resolution of 1920 × 1080.

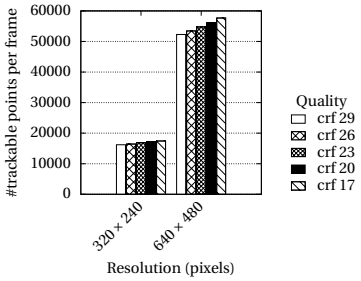
Figure A.10: The compression ratio for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the compression ratio for all CRF factors. (a) – (d) show the compression ratios for each resolution at a fixed frame rate configuration. (e) – (h) show how the compression ratios vary for a constant resolution.



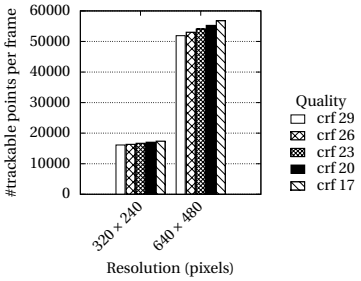
(a) The average number of trackable points at a frame rate of 5 Hz.



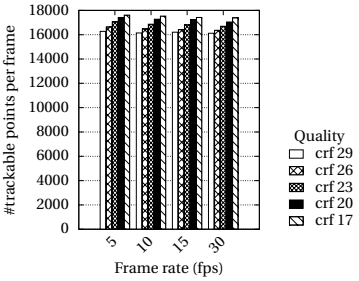
(b) The average number of trackable points at a frame rate of 10 Hz.



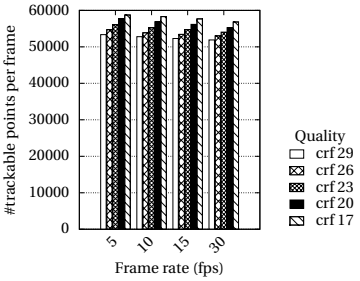
(c) The average number of trackable points at a frame rate of 15 Hz.



(d) The average number of trackable points at a frame rate of 30 Hz.

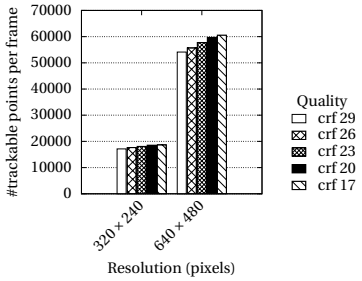


(e) The average number of trackable points at a resolution of 320 × 240.

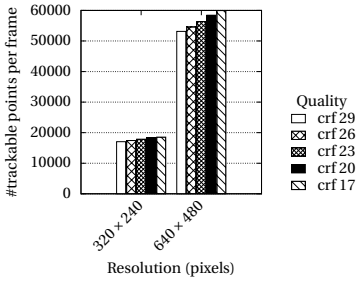


(f) The average number of trackable points at a resolution of 640 × 480.

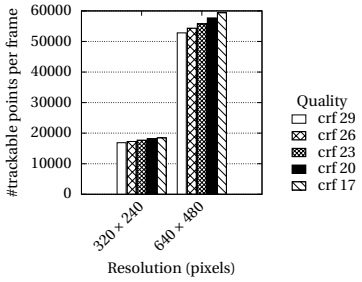
Figure A.11: The average number of trackable points for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the average number of trackable points for all CRF factors. (a) – (d) show the average number of trackable points for each resolution at a fixed frame rate configuration. (e) – (f) show how the average number of trackable points varies for a constant resolution.



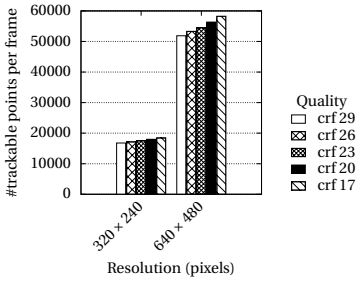
(a) The average number of trackable points at a frame rate of 5 Hz.



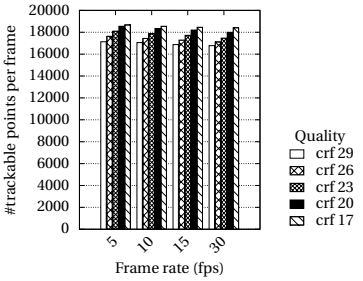
(b) The average number of trackable points at a frame rate of 10 Hz.



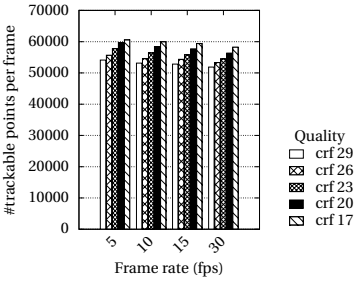
(c) The average number of trackable points at a frame rate of 15 Hz.



(d) The average number of trackable points at a frame rate of 30 Hz.

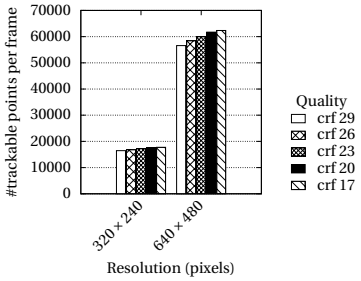


(e) The average number of trackable points at a resolution of 320 × 240.

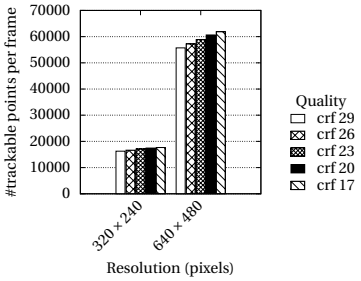


(f) The average number of trackable points at a resolution of 640 × 480.

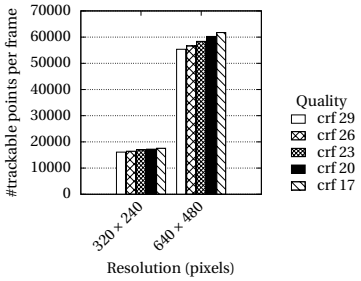
Figure A.12: The average number of trackable points for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the average number of trackable points for all CRF factors. (a) – (d) show the average number of trackable points for each resolution at a fixed frame rate configuration. (e) – (f) show how the average number of trackable points varies for a constant resolution.



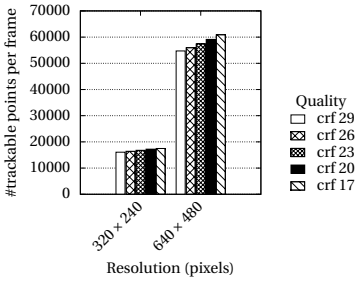
(a) The average number of trackable points at a frame rate of 5 Hz.



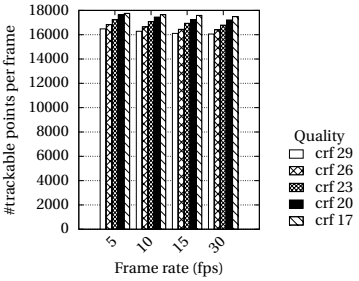
(b) The average number of trackable points at a frame rate of 10 Hz.



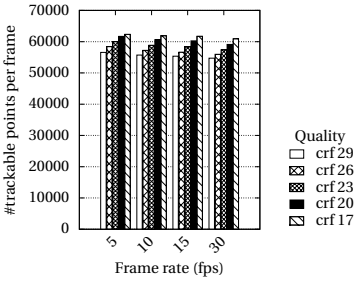
(c) The average number of trackable points at a frame rate of 15 Hz.



(d) The average number of trackable points at a frame rate of 30 Hz.

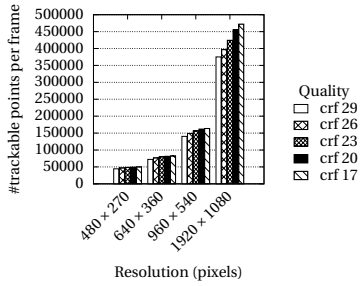


(e) The average number of trackable points at a resolution of 320 × 240.

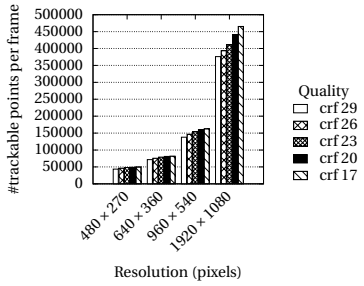


(f) The average number of trackable points at a resolution of 640 × 480.

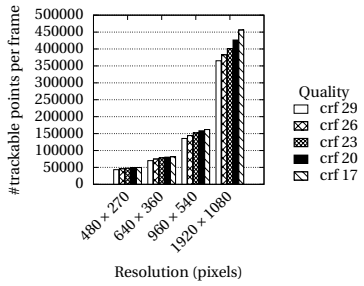
Figure A.13: The average number of trackable points for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the average number of trackable points for all CRF factors. (a) – (d) show the average number of trackable points for each resolution at a fixed frame rate configuration. (e) – (f) show how the average number of trackable points varies for a constant resolution.



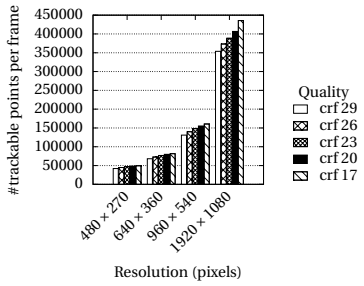
(a) The average number of trackable points at a frame rate of 5 Hz.



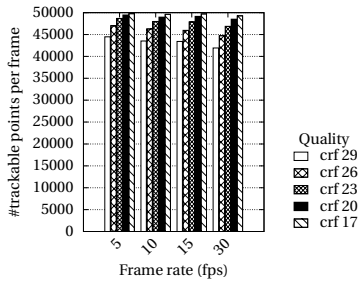
(b) The average number of trackable points at a frame rate of 10 Hz.



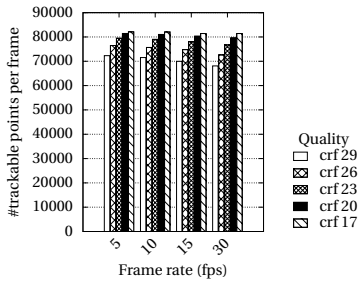
(c) The average number of trackable points at a frame rate of 15 Hz.



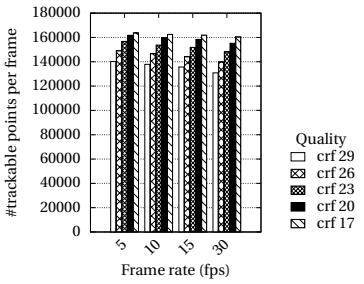
(d) The average number of trackable points at a frame rate of 30 Hz.



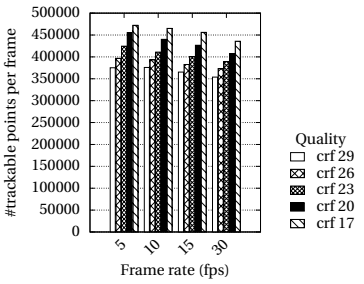
(e) The average number of trackable points at a resolution of 480 × 270.



(f) The average number of trackable points at a resolution of 640 × 360.

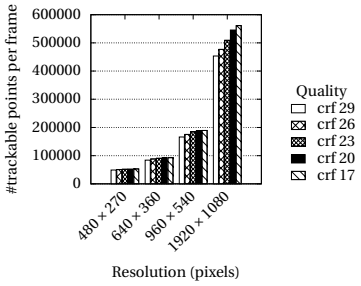


(g) The average number of trackable points at a resolution of 960 × 540.

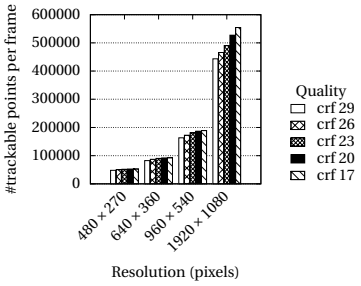


(h) The average number of trackable points at a resolution of 1920 × 1080.

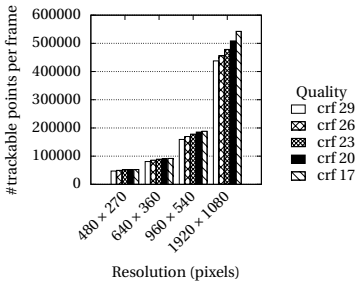
Figure A.14: The average number of trackable points for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the average number of trackable points for all CRF factors. (a) – (d) show the average number of trackable points for each resolution at a fixed frame rate configuration. (e) – (h) show how the average number of trackable points varies for a constant resolution.



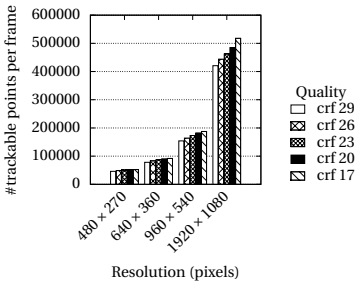
(a) The average number of trackable points at a frame rate of 5 Hz.



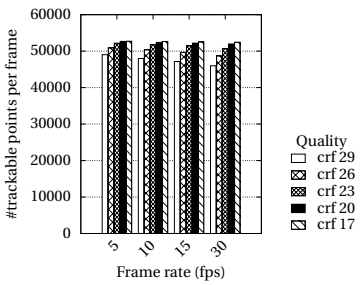
(b) The average number of trackable points at a frame rate of 10 Hz.



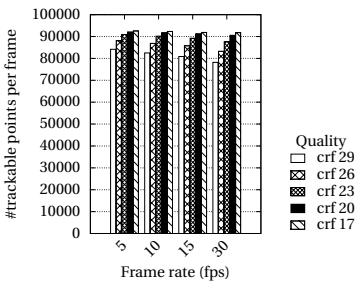
(c) The average number of trackable points at a frame rate of 15 Hz.



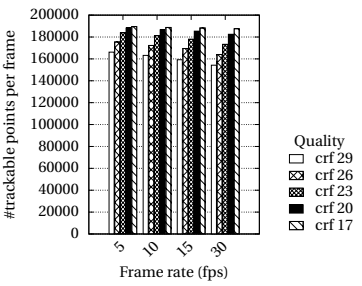
(d) The average number of trackable points at a frame rate of 30 Hz.



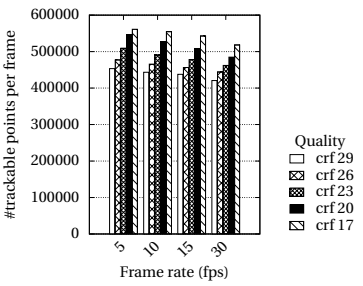
(e) The average number of trackable points at a resolution of 480 × 270.



(f) The average number of trackable points at a resolution of 640 × 360.

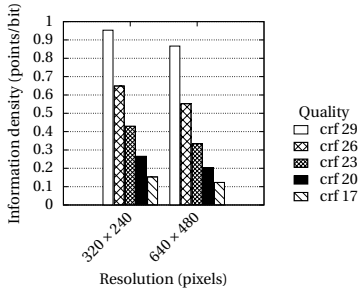


(g) The average number of trackable points at a resolution of 960 × 540.

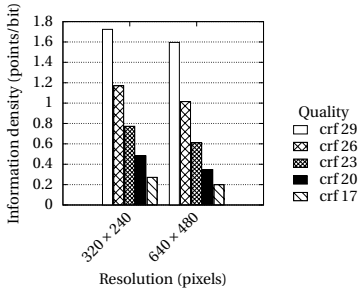


(h) The average number of trackable points at a resolution of 1920 × 1080.

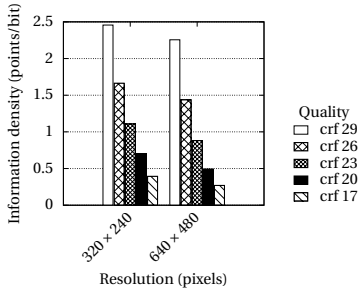
Figure A.15: The average number of trackable points for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the average number of trackable points for all CRF factors. (a) – (d) show the average number of trackable points for each resolution at a fixed frame rate configuration. (e) – (h) show how the average number of trackable points varies for a constant resolution.



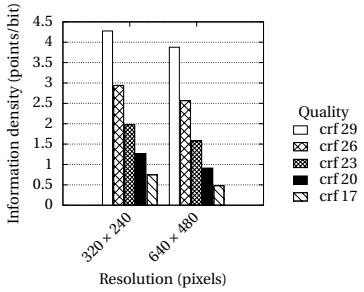
(a) The information density at a frame rate of 5 Hz.



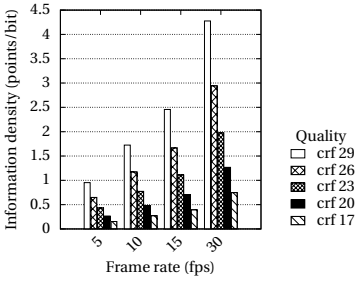
(b) The information density at a frame rate of 10 Hz.



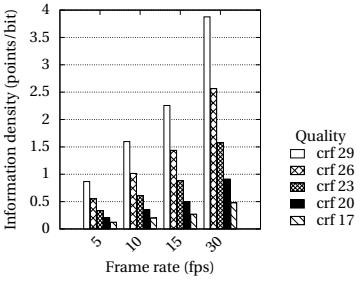
(c) The information density at a frame rate of 15 Hz.



(d) The information density at a frame rate of 30 Hz.

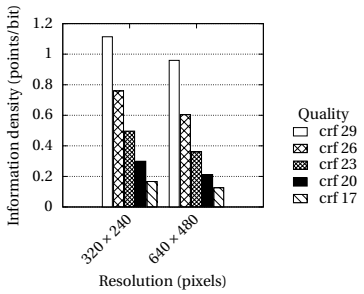


(e) The information density at a resolution of 320 × 240.

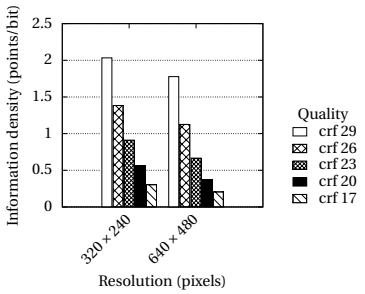


(f) The information density at a resolution of 640 × 480.

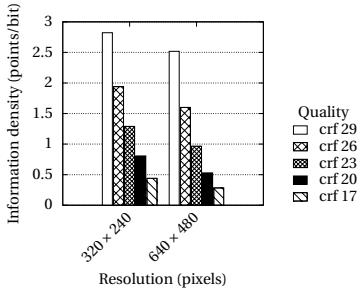
Figure A.16: The information density for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the information density for all CRF factors. (a) – (d) show the information density for each resolution at a fixed frame rate configuration. (e) – (f) show how the information density varies for a constant resolution.



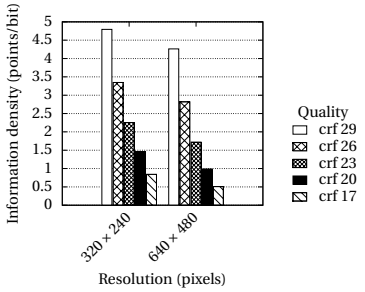
(a) The information density at a frame rate of 5 Hz.



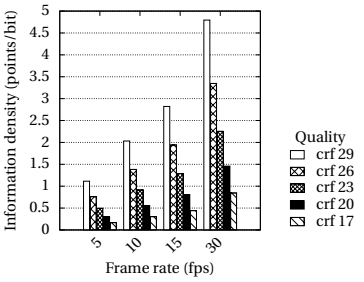
(b) The information density at a frame rate of 10 Hz.



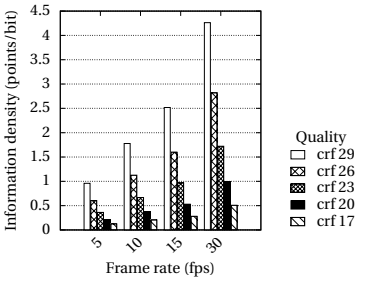
(c) The information density at a frame rate of 15 Hz.



(d) The information density at a frame rate of 30 Hz.

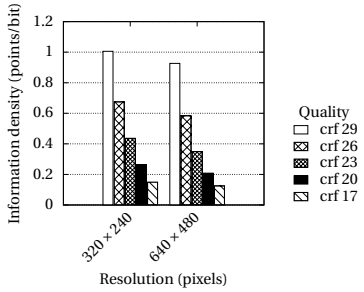


(e) The information density at a resolution of 320 × 240.

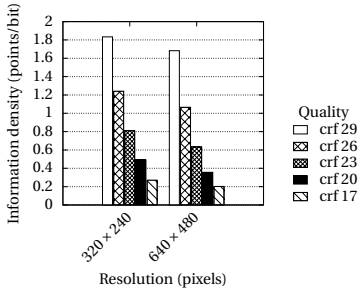


(f) The information density at a resolution of 640 × 480.

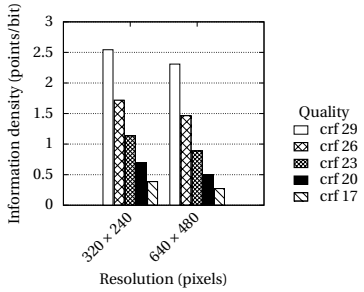
Figure A.17: The information density for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the information density for all CRF factors. (a) – (d) show the information density for each resolution at a fixed frame rate configuration. (e) – (f) show how the information density varies for a constant resolution.



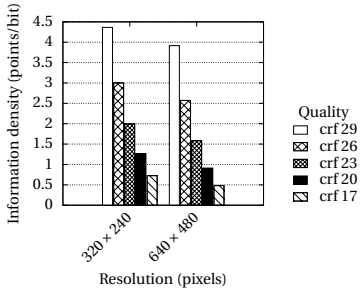
(a) The information density at a frame rate of 5 Hz.



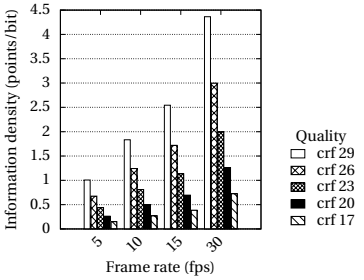
(b) The information density at a frame rate of 10 Hz.



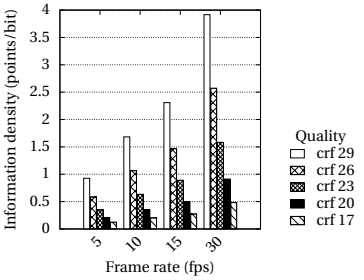
(c) The information density at a frame rate of 15 Hz.



(d) The information density at a frame rate of 30 Hz.

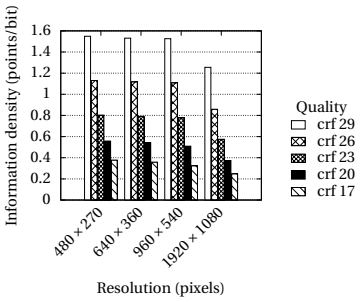


(e) The information density at a resolution of 320 × 240.

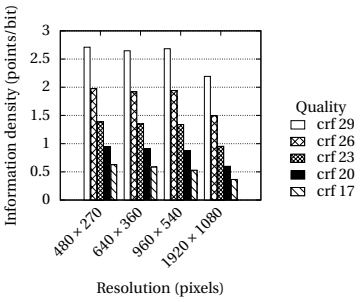


(f) The information density at a resolution of 640 × 480.

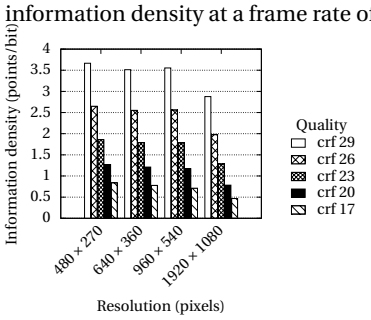
Figure A.18: The information density for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the information density for all CRF factors. (a) – (d) show the information density for each resolution at a fixed frame rate configuration. (e) – (f) show how the information density varies for a constant resolution.



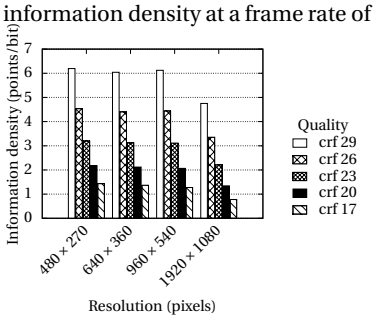
(a) The information density at a frame rate of 5 Hz.



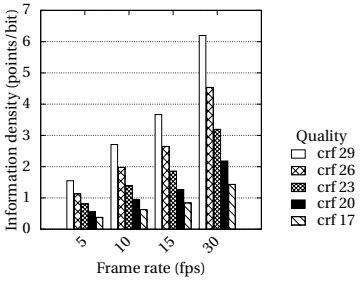
(b) The information density at a frame rate of 10 Hz.



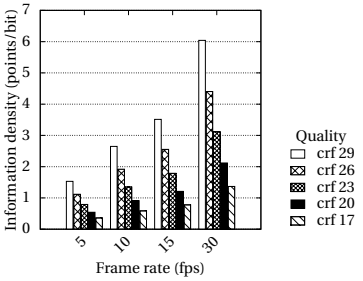
(c) The information density at a frame rate of 15 Hz.



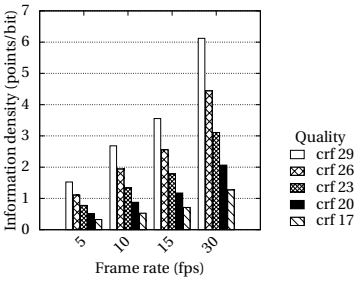
(d) The information density at a frame rate of 30 Hz.



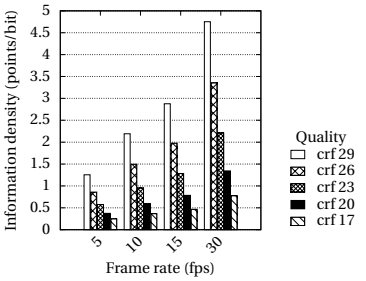
(e) The information density at a resolution of 480 × 270.



(f) The information density at a resolution of 640 × 360.

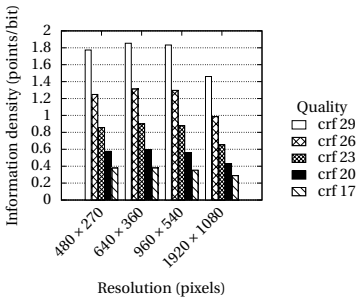


(g) The information density at a resolution of 960 × 540.

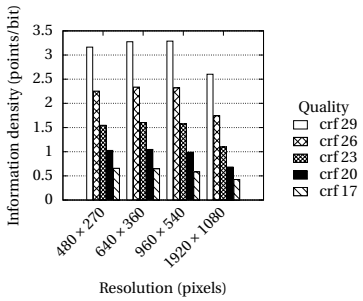


(h) The information density at a resolution of 1920 × 1080.

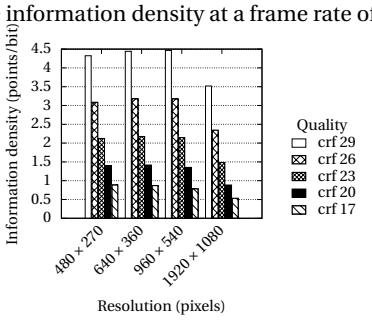
Figure A.19: The information density for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the information density for all CRF factors. (a) – (d) show the information density for each resolution at a fixed frame rate configuration. (e) – (h) show how the information density varies for a constant resolution.



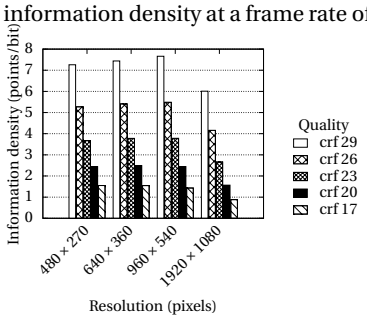
(a) The information density at a frame rate of 5 Hz.



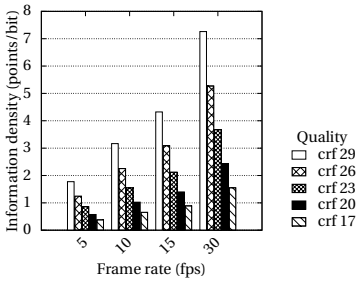
(b) The information density at a frame rate of 10 Hz.



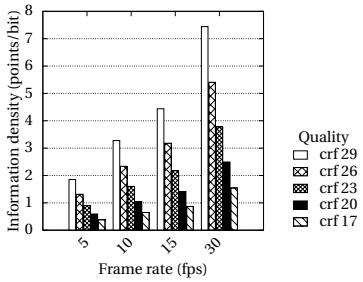
(c) The information density at a frame rate of 15 Hz.



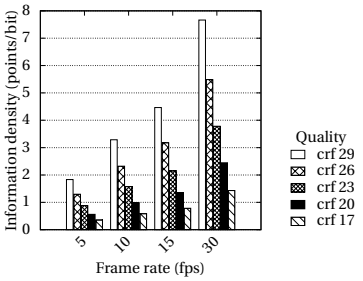
(d) The information density at a frame rate of 30 Hz.



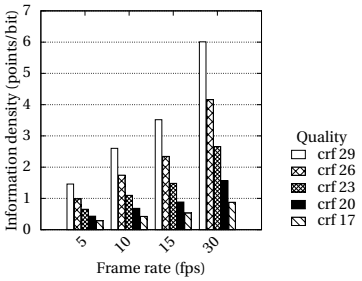
(e) The information density at a resolution of 480 × 270.



(f) The information density at a resolution of 640 × 360.

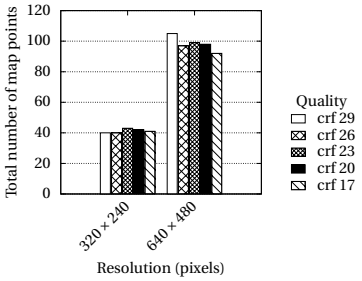


(g) The information density at a resolution of 960 × 540.

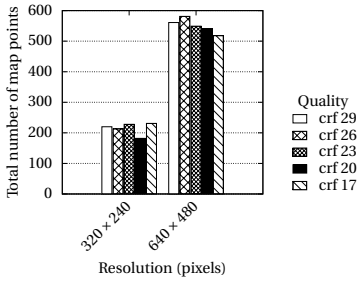


(h) The information density at a resolution of 1920 × 1080.

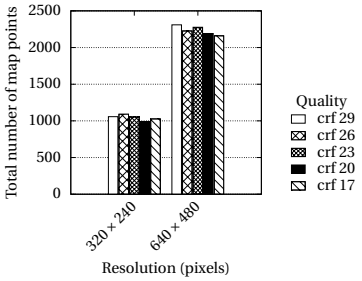
Figure A.20: The information density for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the information density for all CRF factors. (a) – (d) show the information density for each resolution at a fixed frame rate configuration. (e) – (h) show how the information density varies for a constant resolution.



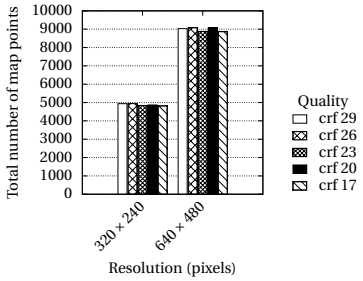
(a) The tracking efficiency at a frame rate of 5 Hz.



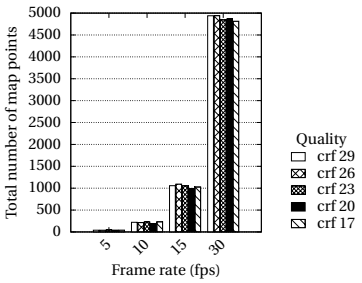
(b) The tracking efficiency at a frame rate of 10 Hz.



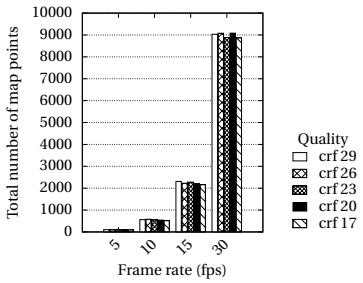
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

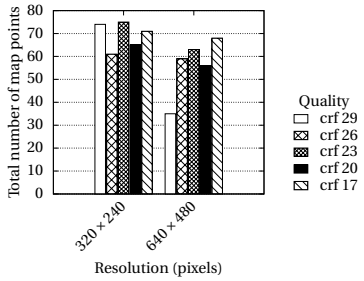


(e) The total number of map points at a resolution of 320 × 240.

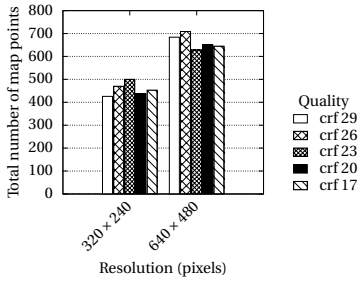


(f) The total number of map points at a resolution of 640 × 480.

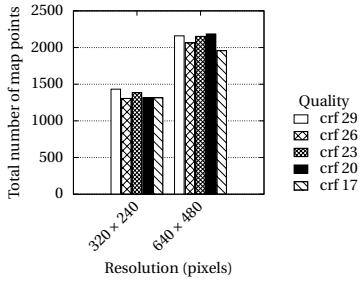
Figure A.21: The total number of map points for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the total number of map points for all CRF factors. (a) – (d) show the total number of map points for each resolution at a fixed frame rate configuration. (e) – (f) show how the total number of map points varies for a constant resolution.



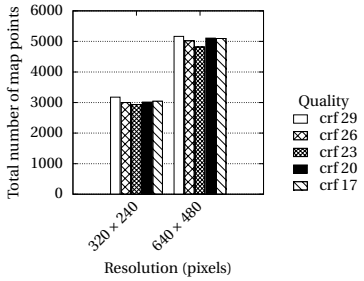
(a) The tracking efficiency at a frame rate of 5 Hz.



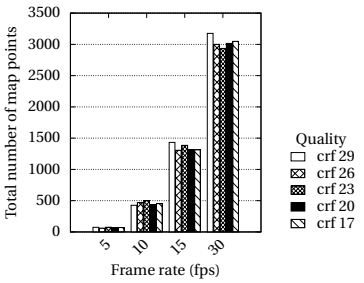
(b) The tracking efficiency at a frame rate of 10 Hz.



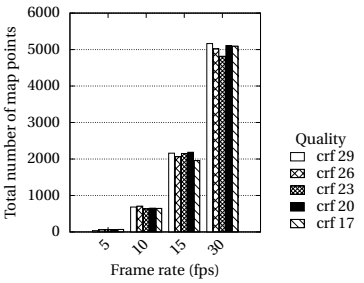
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

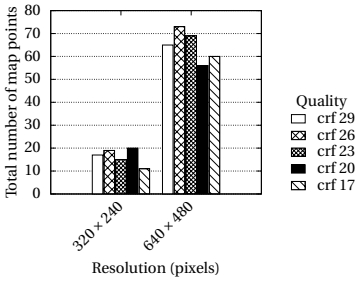


(e) The total number of map points at a resolution of 320 × 240.

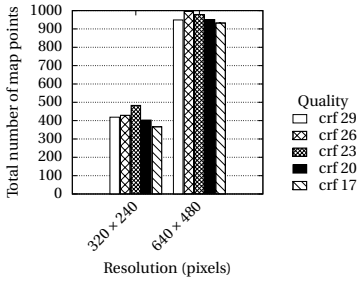


(f) The total number of map points at a resolution of 640 × 480.

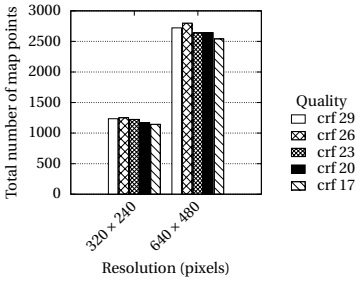
Figure A.22: The total number of map points for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the total number of map points for all CRF factors. (a) – (d) show the total number of map points for each resolution at a fixed frame rate configuration. (e) – (f) show how the total number of map points varies for a constant resolution.



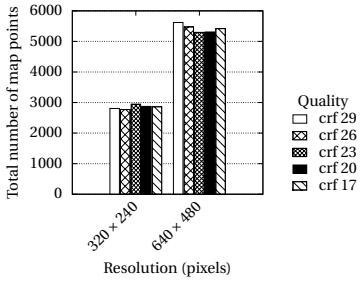
(a) The tracking efficiency at a frame rate of 5 Hz.



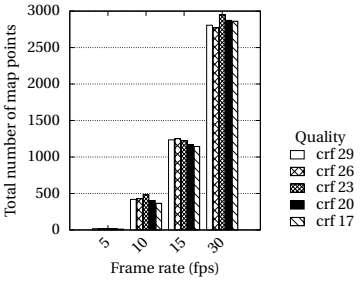
(b) The tracking efficiency at a frame rate of 10 Hz.



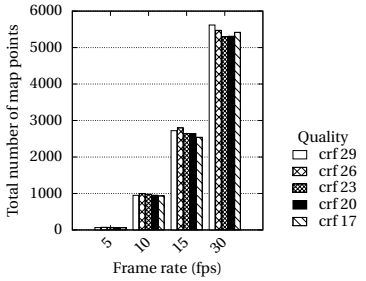
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

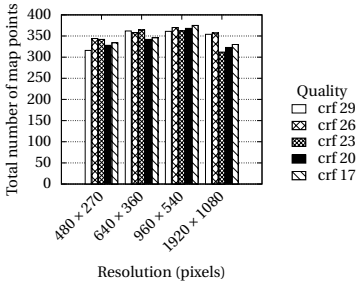


(e) The total number of map points at a resolution of 320 × 240.

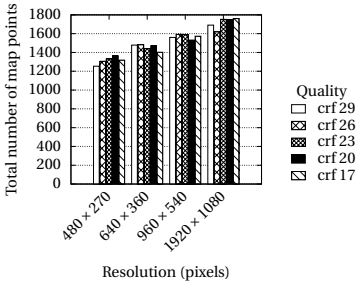


(f) The total number of map points at a resolution of 640 × 480.

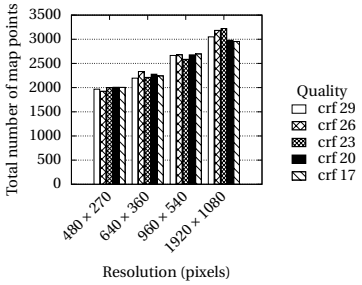
Figure A.23: The total number of map points for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the total number of map points for all CRF factors. (a) – (d) show the total number of map points for each resolution at a fixed frame rate configuration. (e) – (f) show how the total number of map points varies for a constant resolution.



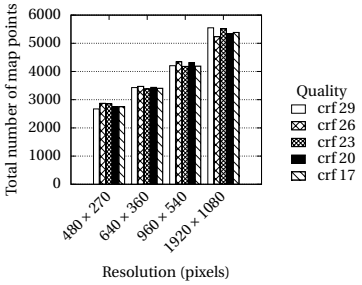
(a) The tracking efficiency at a frame rate of 5 Hz.



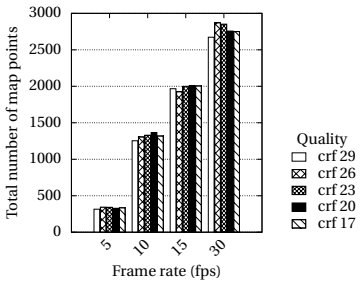
(b) The tracking efficiency at a frame rate of 10 Hz.



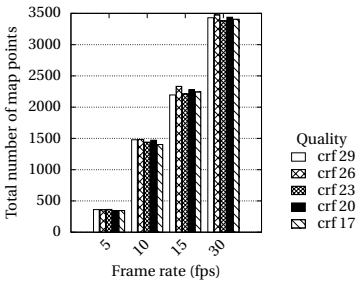
(c) The tracking efficiency at a frame rate of 15 Hz.



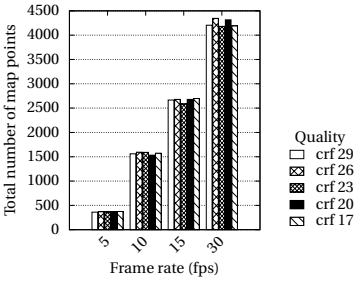
(d) The tracking efficiency at a frame rate of 30 Hz.



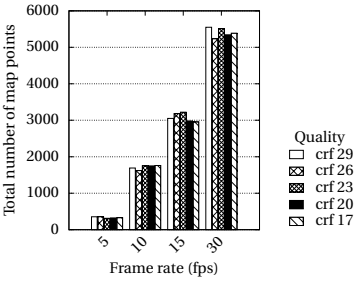
(e) The total number of map points at a resolution of 480 × 270.



(f) The total number of map points at a resolution of 640 × 360.

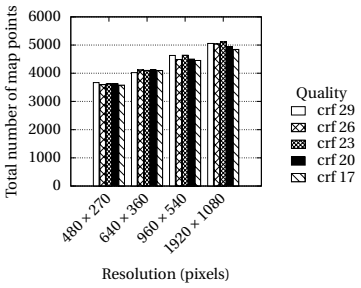


(g) The total number of map points at a resolution of 960 × 540.

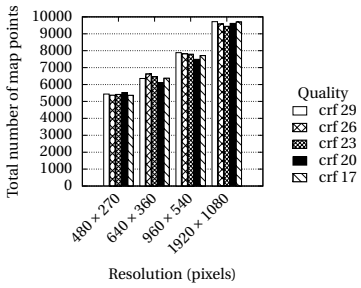


(h) The total number of map points at a resolution of 1920 × 1080.

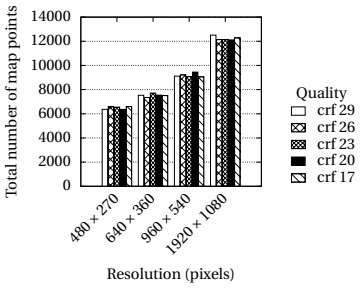
Figure A.24: The total number of map points for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the total number of map points for all CRF factors. (a) – (d) show the total number of map points for each resolution at a fixed frame rate configuration. (e) – (h) show how the total number of map points varies for a constant resolution.



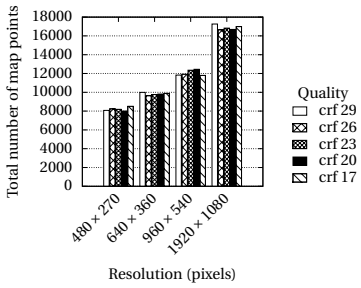
(a) The tracking efficiency at a frame rate of 5 Hz.



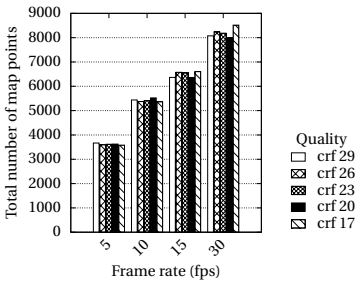
(b) The tracking efficiency at a frame rate of 10 Hz.



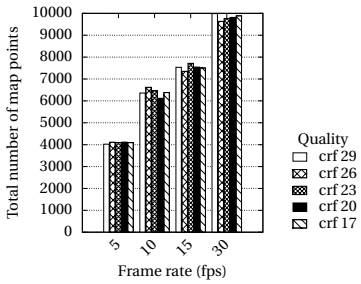
(c) The tracking efficiency at a frame rate of 15 Hz.



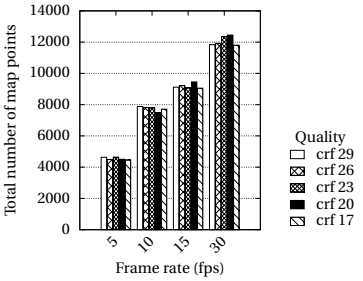
(d) The tracking efficiency at a frame rate of 30 Hz.



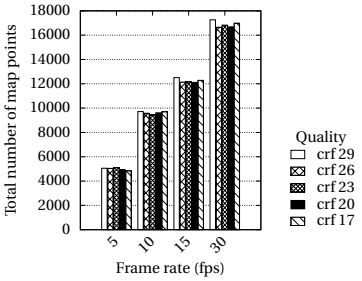
(e) The total number of map points at a resolution of 480 × 270.



(f) The total number of map points at a resolution of 640 × 360.

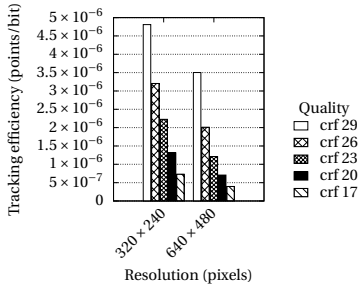


(g) The total number of map points at a resolution of 960 × 540.

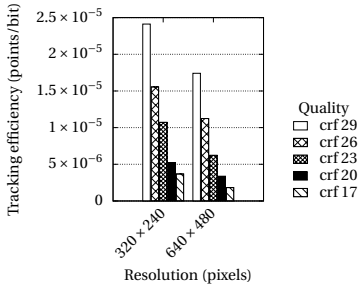


(h) The total number of map points at a resolution of 1920 × 1080.

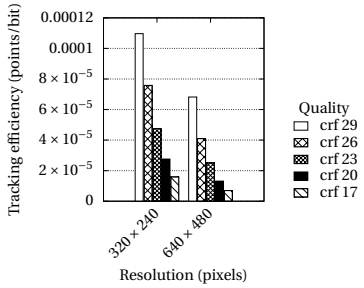
Figure A.25: The total number of map points for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the total number of map points for all CRF factors. (a) – (d) show the total number of map points for each resolution at a fixed frame rate configuration. (e) – (h) show how the total number of map points varies for a constant resolution.



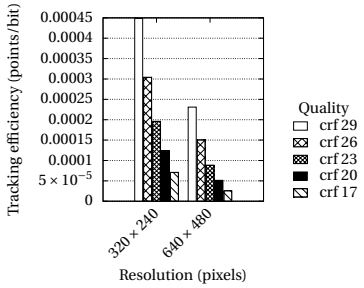
(a) The tracking efficiency at a frame rate of 5 Hz.



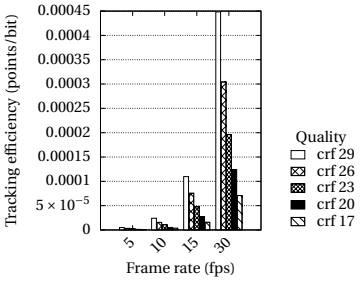
(b) The tracking efficiency at a frame rate of 10 Hz.



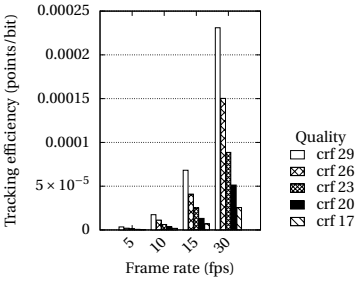
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

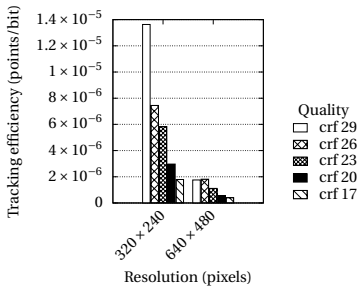


(e) The tracking efficiency at a resolution of 320 × 240.

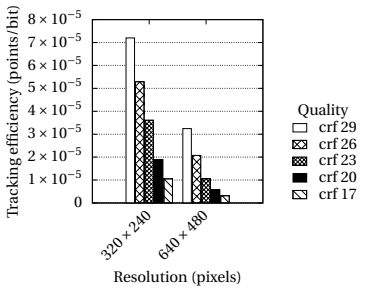


(f) The tracking efficiency at a resolution of 640 × 480.

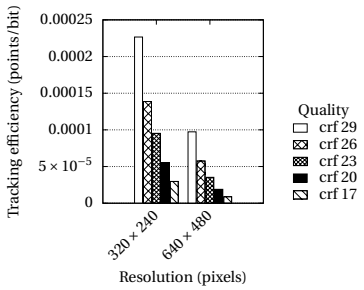
Figure A.26: The tracking efficiency for the RGB-D robot pioneer sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the tracking efficiency for all CRF factors. (a) – (d) show the tracking efficiency for each resolution at a fixed frame rate configuration. (e) – (f) show how the tracking efficiency varies for a constant resolution.



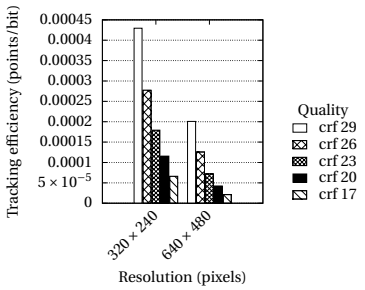
(a) The tracking efficiency at a frame rate of 5 Hz.



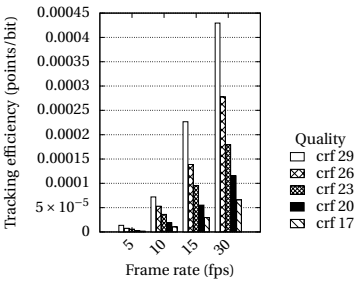
(b) The tracking efficiency at a frame rate of 10 Hz.



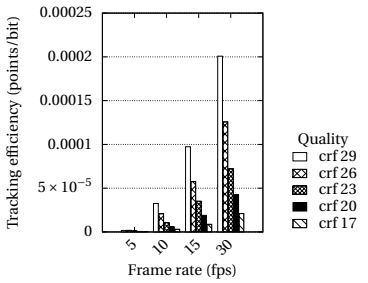
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

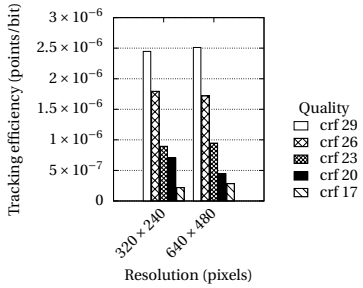


(e) The tracking efficiency at a resolution of 320 × 240.

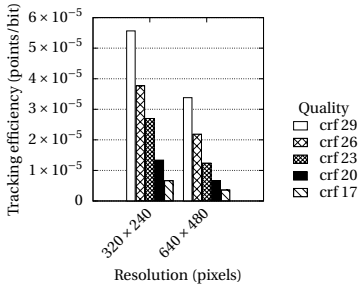


(f) The tracking efficiency at a resolution of 640 × 480.

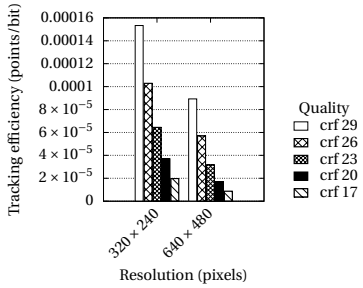
Figure A.27: The tracking efficiency for the RGB-D robot pioneer sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the tracking efficiency for all CRF factors. (a) – (d) show the tracking efficiency for each resolution at a fixed frame rate configuration. (e) – (f) show how the tracking efficiency varies for a constant resolution.



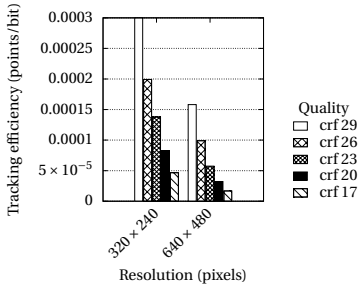
(a) The tracking efficiency at a frame rate of 5 Hz.



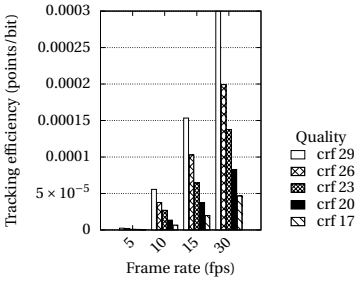
(b) The tracking efficiency at a frame rate of 10 Hz.



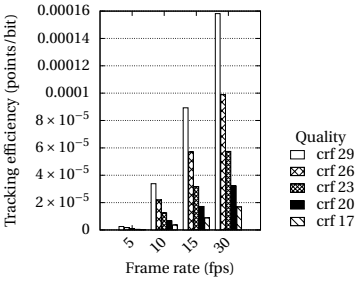
(c) The tracking efficiency at a frame rate of 15 Hz.



(d) The tracking efficiency at a frame rate of 30 Hz.

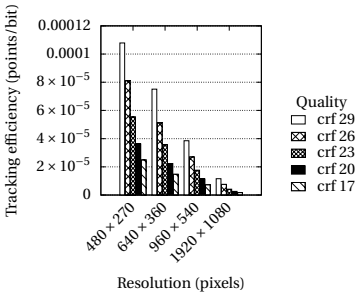


(e) The tracking efficiency at a resolution of 320 × 240.

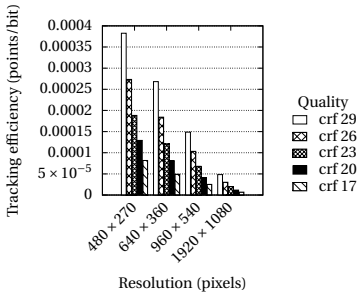


(f) The tracking efficiency at a resolution of 640 × 480.

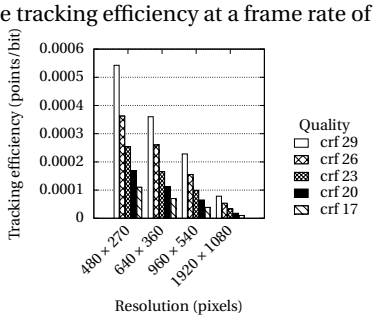
Figure A.28: The tracking efficiency for the RGB-D robot pioneer sequence 3 after encoding the image sequence with a variety of configurations. Each plot shows the tracking efficiency for all CRF factors. (a) – (d) show the tracking efficiency for each resolution at a fixed frame rate configuration. (e) – (f) show how the tracking efficiency varies for a constant resolution.



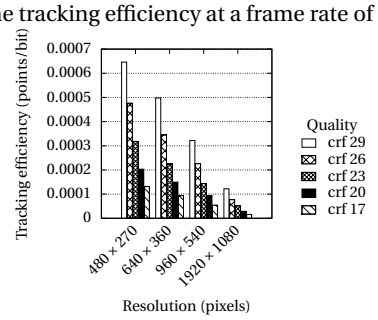
(a) The tracking efficiency at a frame rate of 5 Hz.



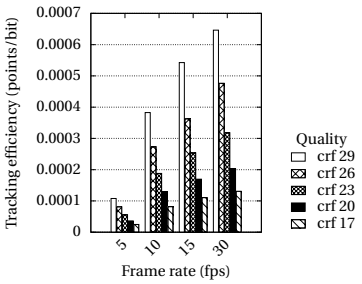
(b) The tracking efficiency at a frame rate of 10 Hz.



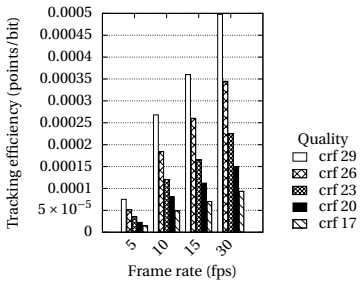
(c) The tracking efficiency at a frame rate of 15 Hz.



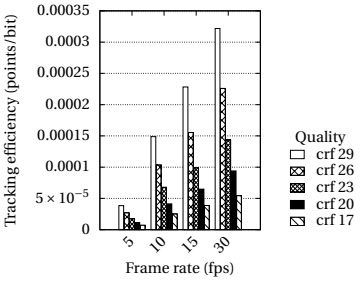
(d) The tracking efficiency at a frame rate of 30 Hz.



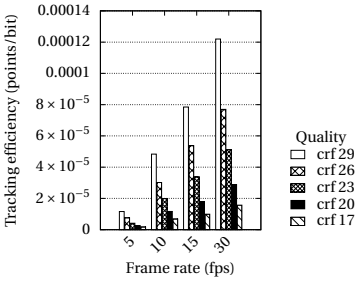
(e) The tracking efficiency at a resolution of 480 × 270.



(f) The tracking efficiency at a resolution of 640 × 360.

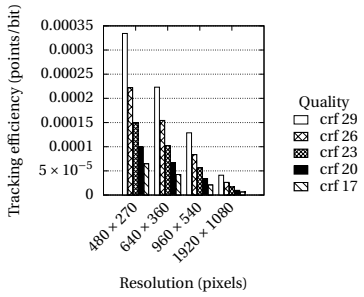


(g) The tracking efficiency at a resolution of 960 × 540.

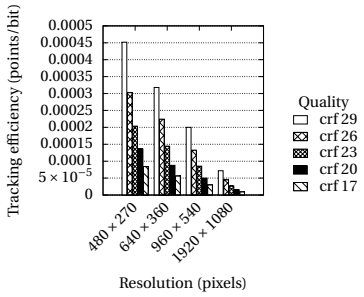


(h) The tracking efficiency at a resolution of 1920 × 1080.

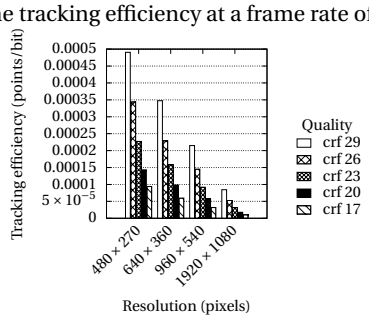
Figure A.29: The tracking efficiency for the CoRBS sequence 1 after encoding the image sequence with a variety of configurations. Each plot shows the tracking efficiency for all CRF factors. (a) – (d) show the tracking efficiency for each resolution at a fixed frame rate configuration. (e) – (h) show how the tracking efficiency varies for a constant resolution.



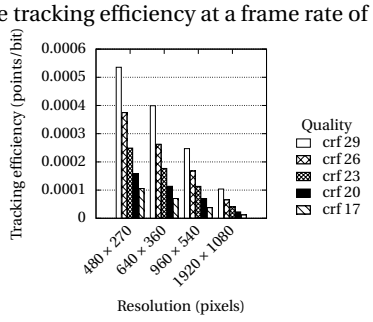
(a) The tracking efficiency at a frame rate of 5 Hz.



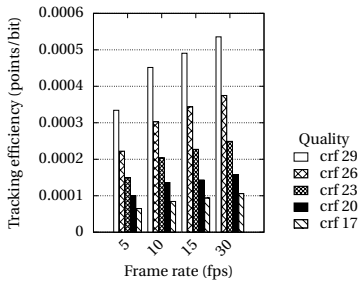
(b) The tracking efficiency at a frame rate of 10 Hz.



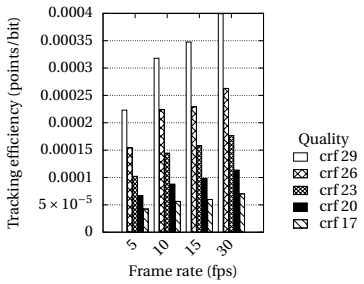
(c) The tracking efficiency at a frame rate of 15 Hz.



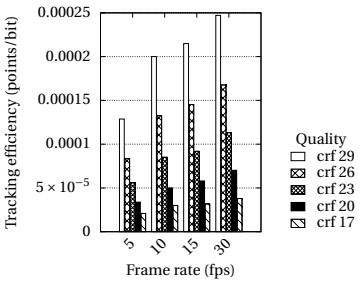
(d) The tracking efficiency at a frame rate of 30 Hz.



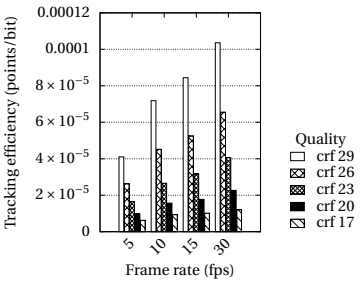
(e) The tracking efficiency at a resolution of 480 × 270.



(f) The tracking efficiency at a resolution of 640 × 360.



(g) The tracking efficiency at a resolution of 960 × 540.



(h) The tracking efficiency at a resolution of 1920 × 1080.

Figure A.30: The tracking efficiency for the CoRBS sequence 2 after encoding the image sequence with a variety of configurations. Each plot shows the tracking efficiency for all CRF factors. (a) – (d) show the tracking efficiency for each resolution at a fixed frame rate configuration. (e) – (h) show how the tracking efficiency varies for a constant resolution.

B Scripts used for the experiments

In this appendix, a copy of the `README` file from the source code of the scripts used during the experiments is provided. It is explained how the scripts can be used to derive the results from this thesis. The scripts themselves are available on the GitLab of the Robotics and Mechatronics Group.

B.1 Scripts used during the thesis *Optimizing wireless video streams for computer vision*

These scripts were used by, Frank van der Hoek (`frank.vanderhoek@gmail.com`), during experiments for the thesis *Optimizing wireless video streams for computer vision*. Using the scripts, it should be possible to recreate the results of my thesis.

B.1.1 Table of contents

B.1.2 Prerequisites

Node

Node version v8.14.0 was used during the experiments. However, the scripts should work for most other versions as well. Node can be downloaded from `https://nodejs.org/en/download/` and installed. For Mac, it is also possible to install it via homebrew by running `brew install node`.

Opencv

The node scripts use OpenCV version 4.1.0. OpenCV is built from source automatically when the npm package `opencv4nodejs` is installed. However, manual install is also possible.

Installation instructions can be found on the opencv website. Pre-built libraries are available for Windows. On linux it is possible to install opencv by running `sudo apt-get install python3-opencv`. For Mac, it is also possible to install it via homebrew by running `brew install opencv@4`. OpenCV can also be installed via python using `pip install opencv-python`.

Datasets

CoRBS dataset The CoRBS dataset can be downloaded from the website of the German Research Center for Artificial intelligence. In the experiments the Desk 1 and Desk 2 sequences were used. Besides the raw images, the camera trajectories for the Desk 1 and the Desk 2 are needed as well. The camera parameters are provided here. To use the camera matrix in a JSON file, copy the following in a file and save it with a `.json` extension:

```
[[1054.35, 0, 956.12], [0, 1054.51, 548.99], [0, 0, 1]]
```

RGB-D dataset The RGB-D dataset can be downloaded from the website of the Technical University of Munich. The pioneer slam 1, 2 and 3 were used for the experiments. The camera

parameters can be found here. To use the camera matrix in a JSON file, copy the following in a file and save it with a `.json` extension:

```
[[525, 0, 319.5], [0, 525, 239.5], [0, 0, 1]]
```

B.1.3 Installation

After cloning this repository, `cd` into it.

If OpenCV is already installed, re-installation can be prevented by running the following command before installing any required node dependencies:

```
# linux and osx:
export OPENCV4NODEJS_DISABLE_AUTOBUILD=1
# on windows:
set OPENCV4NODEJS_DISABLE_AUTOBUILD=1
```

Next, install the required node dependencies by running:

```
npm install
```

B.1.4 Usage of the scripts

There are several scripts that can be used during experiments. After downloading a sequence from one of the datasets, the following process can be used to analyse the performance of the tracking algorithm for videos with different scaling parameters for a sequence of images:

1. Convert the list of images from rgb.txt to a JSON file using the `bag-to-json.js` script.
2. Convert the groundtruth.txt file to a JSON file using the `bag-to-json.js` script.
3. Make sure there is a JSON file containing the camera matrix.
4. Generate spatially scaled versions of the camera matrix using the `generate-camera-matrix.js` script repeatedly.
5. Generate temporally scaled versions of the groundtruth JSON file using the `generate-groundtruths.js` script.
6. Generate videos for all scaling parameters using the `generate-videos.js` script.
7. Count the maximum number of trackable points for each video using the `count-max-trackable-points.js` script.
8. Track points for each video using the `track-points.js` script.

Convert a ROS bag to a json file

This script can be used to convert a ROS bag file to a json file. It assumes that values are space separated and that the 3rd line contains a comment with the headers.

The script reads data from `stdin` and outputs it to `stdout`. An example of a command is:

```
node ./scripts/bag-to-json.js < path/to/bag-file > desired-filename.json
```

Generate a spatially scaled camera matrix

This script can be used to apply spatial scaling to a camera matrix in a JSON file.

It accepts three arguments:

1. The path to the camera matrix JSON file.

2. The source resolution of the camera matrix in `wxh` format.
3. The target resolution of the camera matrix in `wxh` format.

Output is written to `stdout`.

An example command is:

```
node ./scripts/generate-camera-matrix.js \
  /path/to/cam.json 1920x1080 480x270 > cam_480x270.json
```

Generate temporally scaled ground-truth files

This script generates temporally scaled ground-truth files for frame rates of 30Hz, 15Hz, 10Hz, and 5 Hz.

The script accepts 3 arguments:

1. The path to the JSON file containing a list of images and timestamps.
2. The path to the groundtruth.json file with the raw camera poses from the dataset.
3. The output directory (optional).

Output files are written using the filename:

`groundtruth_[frame rate].json`.

An example command is:

```
node ./scripts/generate-groundtruths.js \
  /path/to/rgb-file.json \
  /path/to/groundtruth-file.json \
  /path/to/output-folder
```

Generate all scaled videos for an image sequence

This script generates videos with spatial, temporal, and quality scaling for an image sequence. Spatial scaling parameters are automatically determined based on the image resolution.

It accepts two arguments:

1. The path to the image folder
2. The desired output filename base.

Video files are saved using the filename `[base]_[resolution]_[frame rate]_[quality].mp4`.

An example command is:

```
node ./scripts/generate-videos.js \
  /path/to/image-sequence/rgb \
  /desired/output.mp4
```

This generates videos such as `/desired/output_1920x1080_30_23.mp4`

Count the maximum number of trackable points throughout a video

This script counts the maximum number of trackable points throughout a video.

It accepts one argument, which is the filename of the video.

Output is written in CSV format to `stdout` and contains 1 entry for each second.

An example command is:

```
node ./scripts/count-max-trackable-points.js \
  /path/to/video.mp4 > number-of-points.csv
```

Track points in a video file

This script tracks points in a video.

It accepts 3 arguments:

1. The path to the video file.
2. The path to the spatially scaled camera matrix JSON file.
3. The path to the temporally scaled groundtruth JSON file.

The script may take very long to run and outputs a progress bar to show its progress. For each frame it stores the frameNumber, total number of seeds, number of observable seeds, number of seeds that failed to converge, the average variance of the seeds, and the number of map points in a log file in CSV format. The map points are written to a map file in CSV format, containing the x,y,z and intensity of a 3D point.

An example command is:

```
node ./scripts/track-points.js \
  /path/to/video.mp4 \
  /path/to/cam_1920x1080.json \
  /path/to/groundtruth_30.json
```

Perform tracking for all video files

This is a script that performs tracking for all the videos from both datasets. It limits the number of videos that are processed in parallel to 16, such that the scripts run efficiently on a system with 16 cores.

It assumes that videos are stored in the folder

`$HOME/videos/[dataset name]/[sequence name]/[sequence]_[resolution]_[frame rate]_[crf].mp4`

Where `dataset name` is either `corbs` or `rgbd` and `sequence name` one of `corbs`, `corbs2`, `rgbd`, `rgbd2`, `rgbd3`.

The camera matrices are assumed to be in the folder

`$HOME/camera-matrices/[sequence]/cam_[resolution].json`.

The ground truth files are assuemd to be in the folder

`$HOME/camera-poses/[sequence]/groundtruth_[frame rate].json`.

An example command is:

```
node ./scripts/track-all.js
```

Generate a csv containing all results

This script generates a CSV file containing the columns `dataset`, `sequence`, `resolution`, `framerate`, `quality`, `bitrate`, `duration`, `avgTrackablePoints`, `totalMapPoints`.

The first 5 columns define the video filename.

The bit rate and duration are obtained by the script by executing `ffprobe`. The average maximum number of trackable points is obtained by computing the average of entries in the file

`$HOME/videos/[dataset name]/[sequence name]/[sequence]_[resolution]_[frame rate]_[crf].mp4.csv`

This file is assumed to exist with entries generated by the `count-max-trackable-points.js` script. `dataset-name` is either `corbs` or `rgbd` and `sequence-name` one of `corbs`, `corbs2`, `rgbd`, `rgbd2`, `rgbd3`.

The total number of map points is obtained from the final entry of the file

`$HOME/js/tracker-js/[sequence]_[resolution]_[frame rate]_[quality].mp4_log`.

This file is assumed to have been generated using the `track-points.js` script.

The output of the script is written to `stdout`

An example command is:

```
node ./scripts/gen-final-csv.js > final.csv
```

Bibliography

Van der Auwera, G., P. T. David, M. Reisslein and L. J. Karam (2008), Traffic and quality characterization of the H. 264/AVC scalable video coding extension, *Advances in Multimedia*, **2008**, 2, p. 1.

Capirchio, A. (2017), *Low Latency Video Streaming over Data Distribution Service*, Master's thesis, University of Pisa.

Chiang, J.-C., H.-F. Lo and W.-T. Lee (2008), Scalable video coding of H. 264/AVC video streaming with QoS-based active dropping in 802.16 e networks, in *22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008)*, IEEE, pp. 1450–1455.

Chikkerur, S., V. Sundaram, M. Reisslein and L. J. Karam (2011), Objective video quality assessment methods: A classification, review, and performance comparison, *IEEE transactions on broadcasting*, **57**, 2, pp. 165–182.

Civera, J., A. J. Davison and J. M. Montiel (2008), Inverse depth parametrization for monocular SLAM, *IEEE transactions on robotics*, **24**, 5, pp. 932–945.

Claypool, M. and J. Tanner (1999), The effects of jitter on the peceptual quality of video, in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, ACM, pp. 115–118.

Davison, A. J., I. D. Reid, N. D. Molton and O. Stasse (2007), MonoSLAM: Real-time single camera SLAM, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , 6, pp. 1052–1067.

Eade, E. and T. Drummond (2006), Scalable monocular SLAM, in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, IEEE Computer Society, pp. 469–476.

Eckert, M. P. and A. P. Bradley (1998), Perceptual quality metrics applied to still image compression, *Signal processing*, **70**, 3, pp. 177–200.

Engel, J., V. Koltun and D. Cremers (2017), Direct sparse odometry, *IEEE transactions on pattern analysis and machine intelligence*, **40**, 3, pp. 611–625.

Engel, J., T. Schöps and D. Cremers (2014), LSD-SLAM: Large-scale direct monocular SLAM, in *European conference on computer vision*, Springer, pp. 834–849.

Engel, J., J. Sturm and D. Cremers (2013), Semi-dense visual odometry for a monocular camera, in *Proceedings of the IEEE international conference on computer vision*, pp. 1449–1456.

Forster, C., M. Pizzoli and D. Scaramuzza (2014), SVO: Fast semi-direct monocular visual odometry, in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, pp. 15–22.

Frnda, J., M. Voznak and L. Sevcik (2016), Impact of packet loss and delay variation on the quality of real-time video streaming, *Telecommunication Systems*, **62**, 2, pp. 265–275.

Gardikis, G., G. Xilouris, E. Pallis and A. Kourtis (2012), Joint assessment of Network-and Perceived-QoS in video delivery networks, *Telecommunication Systems*, **49**, 1, pp. 75–84.

Ghinea, G. and J. Thomas (1998), QoS impact on user perception and understanding of multimedia video clips, in *Proceedings of ACM Multimedia*, ACM, pp. 49–54.

Klein, G. and D. Murray (2009), Parallel tracking and mapping on a camera phone, in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, pp. 83–86.

Lin, C.-H., C.-H. Ke, C.-K. Shieh and N. K. Chilamkurti (2006), The packet loss effect on MPEG video transmission in wireless networks, in *20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06)*, volume 1, IEEE, volume 1, pp. 565–572.

Mannos, J. and D. Sakrison (1974), The effects of a visual fidelity criterion of the encoding of images, *IEEE transactions on Information Theory*, **20**, 4, pp. 525–536.

Merritt, L. and R. Vanam (2007), Improved rate control and motion estimation for H. 264 encoder, in *2007 IEEE International Conference on Image Processing*, volume 5, IEEE, volume 5, pp. V–309.

Monteiro, J. M., C. T. Calafate and M. S. Nunes (2008), Evaluation of the H. 264 scalable video coding in error prone IP networks, *IEEE Transactions on Broadcasting*, **54**, 3, pp. 652–659.

Mouragnon, E., M. Lhuillier, M. Dhome, F. Dekeyser and P. Sayd (2006), Real time localization and 3d reconstruction, in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, IEEE, volume 1, pp. 363–370.

Mur-Artal, R., J. M. M. Montiel and J. D. Tardos (2015), ORB-SLAM: a versatile and accurate monocular SLAM system, *IEEE transactions on robotics*, **31**, 5, pp. 1147–1163.

Newcombe, R. A., S. J. Lovegrove and A. J. Davison (2011), DTAM: Dense tracking and mapping in real-time, in *2011 international conference on computer vision*, IEEE, pp. 2320–2327.

Ostermann, J., J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer and T. Wedi (2004), Video coding with H. 264/AVC: tools, performance, and complexity, *IEEE Circuits and Systems magazine*, **4**, 1, pp. 7–28.

Perahia, E. and R. Stacey (2013), *Next generation wireless LANs: 802.11 n and 802.11 ac*, Cambridge university press.

Pinson, M. H. and S. Wolf (2004), A new standardized method for objectively measuring video quality, *IEEE Transactions on broadcasting*, **50**, 3, pp. 312–322.

Robitza, W. (2017a), CRF Guide (Constant Rate Factor in x264, x265 and libvpx), https://slhck.info/video/2017/02/24/crf-guide.html, [Online; accessed 15-July-2019].

Robitza, W. (2017b), Understanding Rate Control Modes (x264, x265, vpx)), https://slhck.info/video/2017/03/01/rate-control.html, [Online; accessed 15-July-2019].

Rui, H.-x., C.-r. Li and S.-k. Qiu (2006), Evaluation of packet loss impairment on streaming video, *Journal of Zhejiang University-SCIENCE A*, **7**, 1, pp. 131–136.

Schierl, T., C. Hellge, S. Mirta, K. Gruneberg and T. Wiegand (2007), Using H. 264/AVC-based scalable video coding (SVC) for real time streaming in wireless IP networks, in *2007 IEEE International Symposium on Circuits and Systems*, IEEE, pp. 3455–3458.

Schwarz, H., D. Marpe and T. Wiegand (2007), Overview of the scalable video coding extension of the H. 264/AVC standard, *To appear in IEEE Transactions on Circuits and Systems for Video Technology*, p. 1.

Segall, C. A. and G. J. Sullivan (2007), Spatial scalability within the H. 264/AVC scalable video coding extension, *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 9, pp. 1121–1135.

Sturm, J., N. Engelhard, F. Endres, W. Burgard and D. Cremers (2012), A Benchmark for the Evaluation of RGB-D SLAM Systems, in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

Teo, P. C. and D. J. Heeger (1994), Perceptual image distortion, in *Human Vision, Visual Processing, and Digital Display V*, volume 2179, International Society for Optics and Photonics, volume 2179, pp. 127–141.

Vogiatzis, G. and C. Hernández (2011), Video-based, real-time multi-view stereo, *Image and Vision Computing*, **29**, 7, pp. 434–441.

Wang, Z. (2001), *Rate scalable foveated image and video communications*, Ph.D. thesis, The University of Texas at Austin.

Wang, Z. and A. C. Bovik (2002), A universal image quality index, *IEEE signal processing letters*, **9**, 3, pp. 81–84.

Wang, Z., A. C. Bovik, L. Lu et al. (2002), Why is image quality assessment so difficult?, in *ICASSP*, volume 4, volume 4, pp. 3313–3316.

Wang, Z., A. C. Bovik, H. R. Sheikh, E. P. Simoncelli et al. (2004), Image quality assessment: from error visibility to structural similarity, *IEEE transactions on image processing*, **13**, 4, pp. 600–612.

Wasenmüller, O., M. Meyer and D. Stricker (2016), CoRBS: Comprehensive RGB-D Benchmark for SLAM using Kinect v2, in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, volume ., IEEE, volume ., p. ., ISBN ., URL http://corbs.dfki.uni-kl.de/.

Wiegand, T., G. J. Sullivan, G. Bjontegaard and A. Luthra (2003), Overview of the H. 264/AVC video coding standard, *IEEE Transactions on circuits and systems for video technology*, **13**, 7, pp. 560–576.

Wikipedia (2019), A sequence of Intra-coded, Predicted and Bi-predicted frames in a compressed video sequence., https://en.wikipedia.org/wiki/Video_compression_picture_types#/media/File:I_P_and_B_frames.svg, [Online; accessed 11-August-2019].

Winkler, S. (1999), Perceptual distortion metric for digital color video, in *Human Vision and Electronic Imaging IV*, volume 3644, International Society for Optics and Photonics, volume 3644, pp. 175–184.