Optimized dynamic multi-view video streaming using view interpolation

Master’s Thesis

submitted by

Pascal Straub
2525320

Supervisor: Prof. Dr.-Ing. Thorsten Herfet
Adviser: M. Sc. Tobias Lange
Reviewers: Prof. Dr.-Ing. Thorsten Herfet
Dr. Christian Rossow

Saarbrücken, 03.02.2016
Eidesstattliche Erklärung
Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath
I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Saarbrücken, den 03.02.2016
Pascal Straub

Einverständniserklärung
Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent
I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, den 03.02.2016
Pascal Straub
Eidesstattliche Erklärung
Ich erkläre hiermit an Eides Statt, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

Statement in Lieu of an Oath
I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Saarbrücken, den 03.02.2016

______________________________
Pascal Straub
Contents

1 Introduction 1

2 Problem statement 3

3 Depth encoding 4

4 Coordinate systems 7

5 Reconstruction algorithm 10

6 Theoretical calculations 16
   6.1 Highest available quality 16
   6.2 Camera coverage 20

7 Optimization algorithm 23
   7.1 Generating the set of views 23
   7.2 Calculation of encoding parameters 27

8 Results 32

9 Conclusion and future work 38

Bibliography 40
Chapter 1

Introduction

The amount and quality of video data which are streamed is increasing constantly. Therefore the required data rate also increases. The ability to have more than one view of a scene available and to switch between them while viewing, furthermore increases the data rate needed. To cope with these needs, extensions for the H.264/MPEG-4 standard are being developed to efficiently encode video data of multi-view video streams. A key functionality of the MVC extension is inter-view prediction which allows us to use prediction not only from other frames in the same view. Also frames of adjacent views from the same point in time can be used for prediction. Therefore the needed data rate is decreased [1–3].

Furthermore new methods which use view interpolation to reduce the data rate while still having an acceptable video quality are analyzed. View interpolation describes the generation of a view between existing ones. The pixels of existing views are transformed to the interpolated image via specific camera matrices. Since not all information of the interpolated view is available in the existing ones, for example caused by occlusions, the interpolated views have less quality than encoded ones. For these new methods good reconstruction algorithms are needed to achieve interpolated views with higher quality and an optimization algorithm which calculates the optimal encoding parameters for the available data rate. With this optimization the highest possible quality for the available data rate is achieved for the multi-view video stream.

In this thesis, we develop an optimization algorithm which calculates the optimal set of views and encoding parameters for a given data rate and a given reconstruction algorithm. After the analyzed problem is stated in chapter 2, the concept of view interpolation is shown in chapter 4 and the reconstruction algorithm used is explained in chapter 5. A calculation of the theoretical maximal achievable quality is done and the influence of the camera placement to the reconstruction quality is analyzed in chapter 6. The results of those calculations are adapted to the situation in which the data rate is limited and an optimal parameter set has to be found. In chapter 7, the two steps of the optimization algorithm are explained.
First, the set of potential views which can be used for reconstruction is calculated and afterward the encoding parameters and the encoded views are optimized for the given available data rate. The results for different data rates, different input scenes and reconstruction qualities are explained in chapter 8.

An additional topic is the encoding of the depth images which are needed for a good reconstruction quality. In chapter 3 different depth encoding methods are analyzed and compared to get a method which reduces the amount of data rate needed by the depth image while having the lowest quality degradation of the reconstructed images.

A conclusion about the algorithm developed and its use for quality improvement is presented in chapter 9. This chapter also shows different topics which can improve the developed optimization algorithm via a reduction of the computation time or an increasing of the resulting quality and methods to verify the given results as well as topics of a future study.
Chapter 2

Problem statement

Based on the results of the master seminar [4], the following topics will be discussed in this master’s thesis.

We want to create an optimizer which calculates an encoding parameter set which maximizes the image quality while using less than a given data rate. In the master seminar, it is shown that an adjustment of the GOP (group of picture) size or the image resolution is no effective method to reduce the data rate while keeping the image quality as high as possible. The GOP size and image resolution should always be chosen to be as large as possible. For other parameters (quantization parameter and number of views) an optimization has to be performed which results in the best quality and a data rate which is as close as possible to the target data rate, but still below it.

Furthermore, the data rate required by depth images has not yet been considered. These depth images are required to reconstruct views from other ones which is a technique to reduce the needed data rate. In publications by Maitre, Merkle and Liu [5–7], it is stated that coded depth images need approximately 20 to 25% of the coded color images. This additionally required data rate has to be considered in the overall calculation.

After these topics are analyzed, we want to have an algorithm which configures the different parameters to achieve the best quality for a given target data rate.
Chapter 3

Depth encoding

For the reconstruction algorithm it is necessary to have depth information from the reference cameras. These depth images have to be transmitted in addition to the color images and therefore consume data rate. To reduce this additional needed data rate as much as possible in this chapter special coding techniques for depth images are analyzed. For depth encoding there are several different methods available. These methods differ in format and encoding scheme of the depth information. Some use existing standards and some other use new principles to code the information more efficiently.

As Merkle et al. stated in "The effects of multiview depth video compression on multiview rendering" [6], in H.264/MVC depth images can be encoded and decoded as conventional color images if the depth images are converted into YUV 4:0:0 format. These images only have information in the luminance channel. This yields a data rate for depth images of 34% of the data rate of the color images. Furthermore nearly the same coding pipeline as for the color images can be used for the depth images.

Pece et al. describe in "Adapting Standard Video Codecs for Depth Streaming" [8] a special compression algorithm for depth images and a different ordering of the depth image information. They encode the 16-bit depth values into 3x8 bits since video codecs compress 24 bits of data per pixel. After this encoding the first six bits are stored in the most significant bits of the first channel, the next five bits in the most significant bits of second channel and the last 5 bits in the most significant bits of third channel. The remaining bits of the three channels will be padded with zeros. The advantage of this technique is the use of the same codec for color and depth images. The data rate of the depth images is 67% of the data rate of the color images.

In 'A Novel Rate Control Technique for Multiview Video Plus Depth Based 3D Video Coding' [7], Liu et al. describe a rate control algorithm for 3D video in which they use different calculation levels and a rate quantization model. After the calculations on the different levels the model is updated and the target rate
is checked. In experiments the data rates for depth images are between 21 and 53\% of the color image data rates, depending on the sequence used. This technique is based on a classifier which has to be trained first with reasonable data. Furthermore the model for this classifier has to be adjusted for different scenes.

Another technique is used by Maitre et al. in 'Joint encoding of the depth image based representation using shape adaptive wavelets' [5]. The encoder proposed in this paper uses additional edge information and encodes the depth and color images by wavelet transformation. With this encoding 20\% of the data rate is needed for the depth images.

Müller et al. store the depth information as inverted real-world depth data in an 8 bit representation such that nearby objects achieve a higher depth resolution than farther away objects. In the paper '3D video formats and coding methods' [9] they describe the advantage of this depth storage method as the independence of the depth values from neighboring cameras. But new depth-enhanced formats and additional scene geometries are needed to get alias-free view synthesis.

'Depth coding using a boundary reconstruction filter for 3D video systems' [10] from Oh et al. describe another method for depth encoding. The proposed algorithm uses a cost function with cost values for occurrence frequency, similarity and closeness. With this algorithm they achieve a data rate saving of about 40\%.

A data rate reduction of up to 29\% is achieved from Shen et al. in the paper 'Edge-adaptive transforms for efficient depth map coding' [11]. They use a mode selection algorithm which chooses between discrete cosine transformation and edge adaptive transformation to reduce the data rate of the depth images.

For the stereo case Pająk et al. show in 'Perceptual depth compression for stereo applications' [12] a color and depth encoding algorithm which is fast and easy to integrate and yield a higher quality for the same data rate than the standard H.264 encoder.

All these techniques need additional computational effort at the client side for decoding of the depth information before being able to start the reconstruction of views. With a depth compression technique like the platelet depth coding a data rate reduction of 40\% is possible without this higher computational effort at the client side. Since the depth images are compressed a lower quality is achieved. The compressed depth image have still a SSIM value of 0.9938 compared to the non-compressed depth image. The quality difference of the reconstructed color images for the compressed and non-compressed depth image is very low: 0.0016 (SSIM) and 0.0577 db (PSNR).

Another option is to not transmit the depth information and construct it from the color images. But this technique yields a lower quality since several errors can occur in the constructed depth images. However the saved data rate (approximately 25\% of the data rate for color images) can be used to get a better quality for the color images. If the better quality of the color images is sufficient to cope with the errors in the depth images this technique can be useful.

All in all it is possible to save 40\% of the data rate of the depth images with a depth encoding technique without a significant quality drop. Since the depth images need approximately 25\% of the data rate for color images (20\% of the overall data rate) a reduction of the data rate for depth images by 40\% yield an overall data rate reduction of up to 8\% with nearly the same quality.

The available data rate has to be reduced by this margin and the resulting data rate can be used to calculate the best quality with the in \textit{chapter 7} described optimization algorithm.
Chapter 4

Coordinate systems

View synthesis describes the generation of virtual views between existing camera positions. With this method the compression of multiple views of a scene can be more efficient than with independent coding of all views [13, 14]. In order to be able to use existing cameras, a set of calibration parameters has to be provided. These parameters describe the position and direction of the camera and the orientation of the image plane inside the camera. Due to occlusions, more than one camera is needed to create a virtual view of decent quality. The more real cameras are used, the more information of the scene can be used.

To perform the transformation from the real cameras to the virtual camera, the intrinsic and extrinsic parameters of the cameras are needed. We transform the coordinates to homogeneous coordinates to describe the nonlinear projective camera geometry by linear mappings in the form of matrices. Therefore we have to add a third coordinate \( w \neq 0 \) as shown in Equation 4.1.

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} \mapsto \begin{bmatrix}
xw \\
yw \\
w
\end{bmatrix}
\]

(4.1)

We set the additional coordinate to 1 such that the world coordinates are described by \((X_w, Y_w, Z_w, 1)\) and the image coordinates by \((x_i, y_i, 1)\).

The intrinsic camera parameters describe the geometry of the image plane inside the camera. They are used to project points from the image coordinate system to the pixel coordinate system, which means from the film plane to the pixel array. The required parameters are usually given in the form of a matrix as shown in Equation 4.2. \( f \) denotes the focal length of the camera, \( m_x \) and \( m_y \) are the scaling factors which describe the number of pixel per mm in x- and y-direction and \( p_x \)
and $p_y$ describe the principal point. The origin of the pixel coordinate system is positioned in the upper left corner.

\[
M_{\text{intr}} = \begin{bmatrix}
fm_x & 0 & px & 0 \\
0 & fm_y & py & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.2)

The extrinsic camera parameters describe the position of the camera using world coordinates as well as the rotation of the camera with respect to the origin of the world coordinates. The camera rotation can be represented as a $3 \times 3$-matrix containing the rotations around every axis. The camera position is given as a three-dimensional vector. To simplify the calculations, the camera rotation and translation are combined in a $4 \times 4$-matrix as shown in Equation 4.3.

\[
M_{\text{extr}} = TR = \begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (4.3)

If we assume that the cameras do not produce a significant amount of lens distortion, the intrinsic and extrinsic matrices can be combined into a projection matrix, see Equation 4.4. This matrix can be used to convert directly between the pixel coordinate system of each camera and the world coordinate system.

\[
M_{\text{proj}} = M_{\text{intr}}M_{\text{extr}} = \begin{bmatrix}
p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\
p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\
p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4}
\end{bmatrix}
\]  \hspace{1cm} (4.4)

The relation between the different coordinate systems is shown in Figure 4.1. The world coordinate system is described by the coordinates $(X_w, Y_w, Z_w)$, the camera coordinate system by $(X_i, Y_i, Z_i)$ and the image coordinate system by $(x_i, y_i)$. $C_i$ is the focal point, which describes the optical center of the camera and $c_i$ is the principal point, which is the intersection between the image plane and the optical axis. The index $i$ describes the camera number. The point $M$ in camera coordinates is projected to the point $m$ in image coordinates.
Figure 4.1: The different coordinate systems and the relations between them.
Chapter 5

Reconstruction algorithm

We use an adapted version of the reconstruction algorithm from Zitnick et al. \[16\] for view synthesis. This algorithm is freely available and used by several researchers for view synthesis such that our results can be compared with experiments in publications of this topic. The pseudo-code of the algorithm is shown in \[\text{Figure 5.1}\]. After reading the given images, the algorithm generates the full projection matrix for all available cameras.

In the next step the two cameras with the smallest distance to the virtual view are determined. The distance calculation, \[\text{Equation 5.1}\], takes the view direction and camera rotation into account and weights the influence of direction and rotation by a factor $\alpha$.

$$\text{distance} = \alpha \ast \text{direction} + (1 - \alpha) \ast \text{rotation} \quad (5.1)$$

read given_images;
for given_camera do
  generate_projection_matrix;
end

calculate_nearby_cameras;
calculate_virtual_depth;
depth_smoothing;
for given_camera do
  project_pixel_to_virtual_view;
  if (given_camera == nearby_camera[0]) do
    dilation_filter;
  end
end
image_inpainting;

\[\text{Figure 5.1}: \text{Pseudo-code of the reconstruction algorithm.}\]
5 Reconstruction algorithm

Via the method of Slabaugh [17] the Euler angles are calculated from the 3-by-3 rotation matrix $R$ of the different views. The pseudo code of this calculation is shown in Figure 5.2.

The views chosen are the nearest ones and mostly share a large portion of their view with the virtual view and therefore provide the main information for the view synthesis. From these two nearby camera images, the algorithm constructs the virtual depth image. It projects each pixel from the original camera image from the pixel coordinate system to world coordinates. Those coordinates are then projected back to pixel coordinates of the virtual view. Because of the different position and angle of the virtual camera, not all pixels in the virtual depth image can be filled with the first depth image. Therefore a depth image from a different camera is used to fill in the missing pixels. Since the available depth images have slight inaccuracies the transformation yields small errors in the transformed depth image. The greater the distance of the reference camera to the transformed depth image the more errors are produced. Therefore only the two nearest depth images are used to generate the transformed depth image. Due to rounding the transformation might result in wrong mappings for a view. To handle these errors the algorithm uses the pixel which has the lowest depth value since this pixel is in the foreground. The different steps of the depth reconstruction are shown in

```plaintext
if($R_{31} \neq \pm 1$)
| $\theta_1 = -\arcsin(R_{31})$
| $\theta_2 = \pi - \theta_1$
| $\psi_1 = \text{atan2}(\frac{R_{12}}{\cos(\theta_1)}, \frac{R_{13}}{\cos(\theta_1)})$
| $\psi_2 = \text{atan2}(\frac{R_{22}}{\cos(\theta_2)}, \frac{R_{23}}{\cos(\theta_2)})$
| $\phi_1 = \text{atan2}(\frac{R_{32}}{\cos(\theta_1)}, \frac{R_{33}}{\cos(\theta_1)})$
| $\phi_2 = \text{atan2}(\frac{R_{32}}{\cos(\theta_2)}, \frac{R_{33}}{\cos(\theta_2)})$
else
| $\phi = \text{anything}; \text{can set to } 0$
| $\delta = \text{atan2}(R_{12}, R_{13})$
| if($R_{31} = -1$)
| $\theta = \frac{\pi}{2}$
| $\psi = \phi + \delta$
| else
| $\theta = -\frac{\pi}{2}$
| $\psi = -\phi + \delta$
| end
else
```

Figure 5.2: Pseudo-code of the calculation of Euler angles from rotation matrices.
Figure 5.3: The green arrow shows a correct mapping of the foreground from the left view and the red arrow shows the same pixel but from the right view. In the right view the pixel is a part of the background, such that this background pixel is not visible in the virtual view. The reconstruction algorithm fills the pixel of the virtual view with the foremost pixel of the given depth images.

Due to the different angle of the virtual view, there are still missing pixels. To fill those missing pixels, a median filter is applied to the reconstructed depth image, see Figure 5.4. We use a median filter because the errors normally consist of single pixel lines inside uniformly colored areas. In such a case median filters are very effective for removing this kind of errors while preserving image details. The disadvantage of this filtering is a higher computational complexity in comparison to a non-filtered image since the median filter has to adjust all pixels of the image.

Figure 5.5 shows the reconstruction steps of the virtual view color image. With the filtered depth image of the virtual view, a projection from pixel coordinates to world coordinates is performed for all pixels in the virtual view. The world coordinates are projected back to pixel coordinates for all given camera images. If the pixel coordinate is inside the image and the depth value of the pixel in the virtual view and in the captured image are in the same range, the pixel color is used to fill the corresponding pixel in the virtual view.

Figure 5.3: Steps of the depth reconstruction.
At the edge between fore- and background, the calculated depth image can contain errors which lead to wrong pixel mappings. To prevent this from happening, a dilation filter with a fixed size of a three-by-three and a rectangular shape is applied to the left view color image as described by Kwan-Jung et al. [18]. The dilation filter increases each area of missing pixels by one pixel in each direction.
This results in less information and more pixels have to be filled by the second depth image. The size is small enough not to erase too much information and big enough to avoid the errors between fore- and background. The rectangular shape is used since the pixels are also a rectangular. Afterwards the deleted pixels are filled using the remaining views. In case pixels from multiple images fulfill those criteria, the pixel in the virtual view gets filled with the pixel value of the first view that fits. An averaging of all views which fulfill the mentioned criteria would result in a blurring of the image, because slight errors in the projection lead to different color values being mapped to the same pixel in the virtual view.

There might still be empty pixels after all given images are included in the new view. For these missing pixels an image inpainting algorithm based on the Navier-Stokes method is used [19]. An extract of the dilated, non dilated and in-painted version of the reconstructed image are shown in Figure 5.6. Both dilation and inpainting result in a better quality of the reconstructed image.

The comparison of the reconstructed and the original image results in a PSNR (peak-signal-to-noise-ratio) value of 27 dB and a SSIM (structural-similarity-index) value of 0.83 for a reconstruction of view 1 from the remaining 7 views. The quality of the reconstruction depends on which view is reconstructed as well as

![Figure 5.6: Extract of the virtual view after a) projection of the left view, b) dilation of this projection, c) filling the missing pixels from different views, d) inpainting. In e) only the dilation step is missing.](image-url)
the position and the number of the remaining camera views. Views at the borders of the camera array are harder to reconstruct than views in the middle. For best results, the different parameters, like the size of the dilation element, the allowed depth difference and the weighting factor for the calculation of the nearest cameras, have to be optimized manually for each scene.
Chapter 6

Theoretical calculations

In this chapter, we calculate the influence of the camera placement on the achievable quality. As a result, the scene coverage of the different cameras and the maximally achievable quality are analyzed. In the next step this is adapted to a real situation in which different parameters are not ideal. Furthermore the influence of a limited data rate is included.

6.1 Highest available quality

In general, the highest available quality $Q$ can be calculated by:

$$Q = \frac{Q_E \cdot m + Q_R \cdot (n - m)}{n},$$  \hspace{1cm} (6.1)

where $Q_E$ is the encoding quality, $Q_R$ denotes the reconstruction quality, $n$ is the number of all views and $m$ denotes the number of encoded views. The quantization value is the most significant parameter which influences the encoding quality $Q_E$. With the assumption that the reconstruction algorithm, described in chapter 5, works perfectly, it holds that $Q_R \approx Q_E$. Since not all pixels in the virtual view will be covered by the capturing cameras, there is a degradation in quality in comparison to the encoded views depending on the pixel coverage of the cameras, see section 6.2. For our work, we analyze combined coverages of at least two cameras since a reconstruction with less cameras yields in a poor quality. In our analysis the reconstruction quality with respect to the coverage is described as $Q_R = (k_1 \cdot cov^2 + k_2 \cdot cov + k_3) \cdot Q_E$, where $cov \in [0,1]$ describes the overall coverage of pixels in the reconstructed view by the reference cameras and $k_1, k_2, k_3$ are scene dependent parameters. Figure 6.1 shows the influence of these combined coverage values to the reconstructed quality. For better readability we define a coverage value $c = (k_1 \cdot cov^2 + k_2 \cdot cov + k_3)$. Therefore the reconstruction quality
6.1 Highest available quality

is described by \( Q_R = c \cdot Q_E \). The overall quality can then be calculated by:

\[
Q = \frac{Q_E \cdot m + Q_E \cdot c \cdot (n - m)}{n}.
\]  

(6.2)

This overall quality can be further decreased by a not perfectly working reconstruction algorithm. Then the reconstruction quality is reduced to \( Q_R < c \cdot Q_E \). The Quality \( Q_R \) depends on the quality of the reconstruction algorithm. To measure this quality, a standardized measurement is required for comparing different reconstruction algorithms. The different reconstruction algorithms have advantages and disadvantages for different scenes or angles, such that a standardized configuration helps to compare them. In this configuration, the distance and angle between the reference cameras, the distance and angle to the virtual view and the used scene have to be specified. If these parameters are configured, the quality of the reconstruction algorithm \( Q_R \) can be estimated by calculating the SSIM value of the reconstructed and the reference image.

The different views are sorted by calculating the Euclidean distance (Equation 6.3) of the cameras, such that view 0 is the view with the highest negative distance to the virtual view and view n-1 is the view with the highest positive distance to the
virtual view of the scene, see Table 6.1 and Figure 6.2.

\[ d(a,b) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \]  

(6.3)

<table>
<thead>
<tr>
<th>camera position</th>
<th>distance</th>
<th>angle</th>
<th>resulting camera number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-15.09, 0.19, 1.38)</td>
<td>-15.16</td>
<td>-0.37</td>
<td>0</td>
</tr>
<tr>
<td>(-11.59, -0.36, 1.05)</td>
<td>-11.64</td>
<td>-0.26</td>
<td>1</td>
</tr>
<tr>
<td>(-7.78, -0.43, 1.39)</td>
<td>-7.92</td>
<td>-0.15</td>
<td>2</td>
</tr>
<tr>
<td>(-3.90, -0.04, 0.17)</td>
<td>-3.91</td>
<td>-0.12</td>
<td>3</td>
</tr>
<tr>
<td>(3.85, 0.04, 0.43)</td>
<td>3.87</td>
<td>0.09</td>
<td>4</td>
</tr>
<tr>
<td>(7.60, -0.05, -0.04)</td>
<td>7.60</td>
<td>0.19</td>
<td>5</td>
</tr>
<tr>
<td>(11.14, 0.20, -0.23)</td>
<td>11.15</td>
<td>0.27</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 6.1: Distance and angle of different cameras to the virtual view at position (0.0, 0.0, 0.0) and resulting camera numbers.

Since the ballet scene is freely available and widely used this scene can be used to measure the quality of the reconstruction algorithm. The reconstruction algorithm has to reconstruct view 4 from three different reference camera combinations:

- small distance (camera 3 and camera 5)
- medium distance (camera 2 and camera 6)
- wide distance (camera 1 and camera 7)

For each distance the SSIM value of the reconstructed image compared to the input image is calculated. For normalization the value is divided by the coverage values of the representing distance. This final score denotes the reconstruction quality value \( r_q \) of the reconstruction algorithm and can be represented as the average value of the three distances or as a curve to specify the behavior of the algorithm for different distances. The reconstruction quality can be estimated by

\[ Q \geq Q_E \cdot m + r_q \cdot c \cdot (n - m) / n. \]  

(6.4)

Furthermore, there are several parameters, which influence the encoding quality \( Q_E \). In the master seminar, we have shown that for best quality results the highest possible GOP size and image resolution have to be used. In addition, a lower quantization value yields a higher quality. But for higher quantization values the
6.1 Highest available quality

quality difference between the encoded and the reconstructed image is reduced such that a user will notice fewer differences between them, see Figure 6.3.

Since the available data rate for the transmission of a MVC stream might be limited, the different parameters have to be adjusted for the given data rate. Therefore the highest quality is not in all cases achieved. In chapter 7 we propose an optimization algorithm which adjusts the encoding parameters for the best quality under a given data rate limit. An optimization of the encoding parameters and the different views is needed to achieve the maximal quality for the limited data rate.
6.2 Camera coverage

The visibility of pixels from the virtual view in the capturing cameras is described as camera coverage. The more pixels are visible the higher is the coverage value $c$, since more information of the virtual view are available. This coverage value is needed to calculate the views to encode and to reconstruct. Cameras with higher coverage are used for encoding.

In [20], Zhang et al. use the camera coverage to build "A Self-Reconfigurable Camera Array", which adjusts the cameras for the best available pixel coverage. Mavrinac et al. computed a three-dimensional coverage model for cameras based on fuzzy sets in [21]. For this model, they calculate the overall coverage $C$ as the intersection of the different subsets visibility $C_V$, resolution $C_R$, focus $C_F$ and direction $C_D$:

$$C = C_V \cap C_R \cap C_F \cap C_D \quad (6.5)$$

\[1\] Fuzzy sets are defined as a pair $(U,m)$, where $U$ describes a set and $m : U \rightarrow [0,1]$ describes a membership function. For $x \in U$; $x$ is fully included in the fuzzy set $(U,m)$, if $m(x) = 1$; not included, if $m(x) = 0$ and a fuzzy member, which means partially included, if $0 < m(x) < 1$. 

Figure 6.3: SSIM value of encoded and reconstructed images of the ballet scene for different quantization values.
For our approach, we can reduce these subsets. The resolution is inversely proportional to the z-coordinate of the pixels. We use cameras which have the same resolution and the cameras are in the same distance range to the scene. Therefore the resolution in the used images is for all pixels high enough not to influence the coverage model. It holds, that the membership function for resolution \( m_R \) equals one for all pixels. Focus is also a function of the z-values. In our images the z-values of all pixels are between the minimum value \( z_{\text{near}} \) and the maximum value \( z_{\text{far}} \) which is the range of the acceptable depth values to be in focus. Therefore the membership function for focus \( m_F \) equals also one for all pixels. All pixels are in focus such that there is no influence on the coverage model from this subset. In our approach, the overall coverage therefore depends on the subsets visibility and direction. For the subset direction there exist three cases. First case, the camera is pointing towards the scene and captures the scene completely \((m_D = 1)\). Second, the opposite case, the camera is pointing away from the scene \((m_D = 0)\). And the last case, a part of the scene is captured by the camera \((0 < m_D < 1)\). In our images the cameras always point towards the scene and capture the scene partially or fully. There exist also three cases for visibility. A pixel is fully visible \((m_V = 1)\), partially visible \((0 < m_V < 1)\) or not visible \((m_V = 0)\). For our work, we only use fully covered pixels, because we address only full pixels. Therefore we can combine the two cases partially visible and not visible, if we treat a partially visible pixel as a not visible pixel. If the camera points away from the scene, the pixels cannot be visible and therefore it holds that \( m_D = 0 \Rightarrow m_V = 0 \). If the direction is towards the scene, partial or full, a number of pixels is visible. Therefore we can calculate the visibility of all pixels in the scene, to calculate the overall coverage of a specified camera. With this, a camera direction away from the scene implies no pixel is visible and for a direction towards the scene, the calculation of visibility \( C_V \) is sufficient to calculate the overall coverage.

This holds true for a single image or frame. But for different frames the coverage value can change since objects will move from background to foreground or from one side of the image to another. This movement yields a change of the pixel visibilities and coverage values. For a valid coverage value of a stream more frames have to be evaluated. Since the number and location of the encoded views cannot change between different frames, the average coverage over all frames of the video stream has to be as high as possible.

Another possibility is to divide the whole video in small parts and evaluate the coverage over the different parts of the video and adjust the encoding parameters for the different parts separately. This results in a higher computational effort but also a better individual adjustment of the parameters.

The coverage value is used by the optimization algorithm to calculate the optimal
set of encoded and reconstructed views, see [chapter 7]. With this set it is possible to maximize the overall quality of the MVC stream for a given target data rate.
Chapter 7

Optimization algorithm

In this chapter, an optimization algorithm is provided which calculates the optimal number of views which should be transmitted and the encoding parameters for a given data rate. To achieve the highest quality of an MVC stream for a given data rate, the optimal encoding parameters and the number and location of the encoded views have to be calculated. To choose the best views for encoding, the camera coverage value, see section 6.2, is used. After this step the encoding parameters and the number of views are optimized for a given data rate.

7.1 Generating the set of views

The optimal number of encoded views and their location depend on the camera parameters and the resulting coverage of the different cameras. Also the computing power of the client system can influence the number of views if only a limited number of views can be reconstructed. For a given camera placement the algorithm calculates the camera coverage of all possible combinations of encoded and reconstructed views as follows. First, the visibility of all pixels is calculated by projecting them from the virtual view to the different existing cameras. Afterward the projected position of the pixels is checked to verify that the pixels are in the range of the reference images. After this step all pixels covered by the specified camera are known. Furthermore the depth values of both images are compared to check if a pixel is visible or occluded by a foreground object, see Figure 5.3. The coverage value for all available cameras and information about which pixels are covered is stored, such that the combined coverage of any camera combination can be computed. This is required to calculate which pixels are covered while avoiding unnecessary recalculations. For example, view 4 of the ballet scene is reconstructed by camera 3 and camera 5. Camera 3 has a coverage of 0.86 and camera 5 a coverage of 0.89. The combined coverage is only 0.98 and not the addition of the two separate camera coverages, see Figure 7.1. The value of the combined coverage is calculated by generation of a matrix with the dimensions of
7.1 Generating the set of views

Figure 7.1: Coverage image of a) camera 3, b) camera 5 and c) combination of both cameras.

the image. For all pixels which are covered by at least one of the reference cameras the matrix entry is set to one, see Figure 7.2. Afterward these entries are counted and divided by the number of all entries to get the combined coverage.

The combined coverage is used to calculate the set of encoded and reconstructed views which results in the highest quality for a specified configuration. The distance and rotation of the different cameras to each other such as the distance of the camera to the scene is considered by the coverage calculation. In the sequences used, the cameras have the same spacing in between and only slight differences in the distance to the scene. These differences have only low influence to the coverage value. For sequences where the cameras have a wider spacing the resulting coverage is lower than for camera placements with lower distance between the cameras.

The pseudo code of the used algorithm is shown in Figure 7.3.

The outer views (view 0 and view n-1), see Figure 6.2, have to be encoded to achieve the highest quality, since at least one view on the left and on the right side of the reconstructed view is needed to achieve a good reconstruction quality. For the other views the camera coverage with respect to the next potentially reconstructed view is calculated. Therefore the above described calculation of the combined coverage is used. If this coverage value is above a predefined threshold, the index of the second reference camera is increased by one and the coverage of this

\[
\begin{bmatrix}
0 & 0 & 1 & \ldots \\
0 & 1 & 0 & \ldots \\
: & : & : & \ldots \\
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 1 & \ldots \\
1 & 0 & 0 & \ldots \\
: & : & : & \ldots \\
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 1 & \ldots \\
1 & 1 & 0 & \ldots \\
: & : & : & \ldots \\
\end{bmatrix}
\]

coverage camera 3  coverage camera 5  combined coverage

Figure 7.2: Example of the calculation of the combined coverage matrix with the coverage matrices of two reference cameras camera 3 and camera 5.
7.1 Generating the set of views

```c
first_view = 0;
last_view = n-1;
i = 2;
encode(0);
encode(n-1);
do{
    virtual_view = first_view+1;
    cov = calculate_coverage(first_view, first_view+i, virtual_view);
    if (cov >= threshold && first_view+i < last_view)
        i++;
    else if (cov < threshold && first_view+i < last_view){
        encoded_view = first_view+i-1;
        encode(encoded_view);
        for(j = 0; j < encoded_view; j++)
            if(encoded_view > virtual_view+j)
                reconstruct(virtual_view+j);
    }
    first_view = encoded_view;
i = 2;
}
else if (cov >= threshold && first_view+i >= last_view){
    encoded_view = first_view+i;
    for(j = 0; j < encoded_view; j++)
        if(encoded_view > virtual_view+j)
            reconstruct(virtual_view+j);
}
break;
}
else{
    encoded_view = first_view+i;
    encode(encoded_view);
    for(j = 0; j < encoded_view; j++)
        if(encoded_view > virtual_view+j)
            reconstruct(virtual_view+j);
}
break;
}while(first_view+i <= last_view);
```

Figure 7.3: Pseudo code of the algorithm to calculate encoded and reconstructed views.
7.1 Generating the set of views

combination is calculated. If the coverage is also above the threshold, the index is further increased until the last view is reached. If the coverage value is lower than the threshold, the index of the second reference camera is decreased by one and this view is used for encoding. All views between the first reference camera and the decreased index are used for reconstruction because the combined coverage value of the reference cameras is high enough. An example of this decision for encoding and reconstruction is shown in Table 7.1. For a threshold of 0.98 the views 3 and 5 are reconstructed and the remaining views are encoded. Since adjacent cameras have the same distance to each other a foreground object blocks the same amount of pixels in the background. Therefore only a low difference between the coverage values of different view combinations is noticeable in this scene.

The threshold for a view to be added to the reconstruction set has to be adjusted to the quality of the reconstruction algorithm. If the algorithm has no significant drop in quality for medium and wide distances of the reconstructed view to the reference cameras the required coverage value for reconstruction can be decreased. For our algorithm we choose a threshold coverage value of 0.98 since our reconstruction quality value is only 0.86 for wide angles. For this low reconstruction quality value the algorithm needs a high coverage to produce acceptable quality results. The reconstruction quality is even higher for smaller angles between the reference cameras such that a maximal reconstruction quality value of 0.94 is achieved for direct neighbors as reference cameras. The resulting coverage for the different angles is shown in Table 7.2.

If the client side allows only a limited number of reconstructed views since the client cannot handle the additional computational effort of the reconstruction, our algorithm calculates the coverage values between pairs of views which have one view in between. The algorithm starts with view 0 and view 2 and increases the view number after the coverage calculation. After all views are analyzed the algorithm chooses the views with the highest coverage. The coverage values are

<table>
<thead>
<tr>
<th>reference cameras</th>
<th>potential view to reconstruct</th>
<th>coverage</th>
<th>reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 and 2</td>
<td>1</td>
<td>0.97</td>
<td>no</td>
</tr>
<tr>
<td>1 and 3</td>
<td>2</td>
<td>0.97</td>
<td>no</td>
</tr>
<tr>
<td>2 and 4</td>
<td>3</td>
<td>0.98</td>
<td>yes</td>
</tr>
<tr>
<td>2 and 5</td>
<td>4</td>
<td>0.97</td>
<td>no</td>
</tr>
<tr>
<td>4 and 6</td>
<td>5</td>
<td>0.98</td>
<td>yes</td>
</tr>
<tr>
<td>4 and 7</td>
<td>6</td>
<td>0.97</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 7.1: Coverage of different camera combinations and decision to encode or reconstruct for a threshold of 0.98.
7.2 Calculation of encoding parameters

<table>
<thead>
<tr>
<th>angle of reference cameras</th>
<th>coverage value</th>
<th>reconstruction quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>small (camera 3 and camera 5)</td>
<td>0.97</td>
<td>0.94</td>
</tr>
<tr>
<td>medium (camera 2 and camera 6)</td>
<td>0.95</td>
<td>0.89</td>
</tr>
<tr>
<td>wide (camera 1 and camera 7)</td>
<td>0.86</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 7.2: Coverage of different camera combinations and resulting reconstruction quality value for a reconstruction of view 4.

stored as a set of key-value pairs to allow fast sorting later on. If the coverage value is higher than the predefined threshold the view is added to the potential reconstruction set. The other views are added to the encoding set. Afterward the coverage values of the potential reconstruction set are sorted and the highest \( k \) values which are above the threshold are chosen for the reconstruction set. All remaining views are set to encoding such there is the possibility that less than \( k \) views are reconstructed if the coverage of the cameras is low. The pseudo code of this modification is shown in Figure 7.4.

7.2 Calculation of encoding parameters

After the sets of encoded and reconstructed views have been determined, the encoding parameters and the number of reconstructed views are adjusted for the given data rate. The optimization algorithm performs the following steps:

- calculate the best set of views for encoding and reconstruction for the given configuration using coverage analysis
- calculate quality and data rate for all combinations from the above setting
- choose the parameters which result in highest quality while the resulting data rate is below the target data rate.

For the first step the coverage value of the different views is calculated to generate the set of views to encode and to reconstruct as described in section 6.2. Afterward the quantization parameter and the number of views with the highest quality and target data rate are calculated from the given sets via a search over the given parameter spaces. For a client system which is only able to reconstruct \( k \) views, the algorithm calculates the best \( k \) views for reconstruction. In Figure 7.5 the pseudo code of this search algorithm is shown. To speed up the algorithm the search space can be decreased, for example by skipping extreme quantization parameters. This can be done by adjusting the parameter \( min_{qp} \) and \( max_{qp} \). As output, the optimization algorithm returns the quantization value, the views which should be
7.2 Calculation of encoding parameters

```c
j = 0;
encode(0);
    encode(n-1);
for(left = 0; left < n-2; left++){
    virtual_view = left+1;
    right = left+2;
    c.make_pair(virtual_view, calculate_coverage(left, right, virtual_view));
}
sort(c, by_value);
for (i = 0; i < c.size(); i++){
    if(c.value > threshold && j < number_of_recon){
        rec(c(i));
        j++;
    }
    else
        enc(c(i));
}
sort(rec, by_key);
for(i = 0; i < rec.size(); i++)
    reconstruct(rec(i).key);
sort(enc, by_key);
for(i = 0; i < enc.size(); i++)
    encode(enc(i).key);
```

Figure 7.4: Pseudo code of the algorithm to calculate encoded and reconstructed views for a limited number of reconstructed views.

encoded and reconstructed, the data rate and the resulting quality that the used configuration will achieve according to our calculation.

The data rate is calculated as the number of encoded views multiplied by the data rate of the base view. This results in a higher calculated data rate since inter-view coding reduces the data rate needed for additional views. Since similarities in the different views are used to efficiently code the data of the MVC stream, the data rate of additional views is reduced in comparison to the data rate of the base view. For this an additional parameter for inter-view coding $i_c$ can be used to adjust the data rate. In the analyzed sequences this value is between 0.75 and 0.95 and depends on the quantization value. This means that each additional view needs only $i_c$ times the data rate of the base view. Using better inter-view coding means that the overall data rate is reduced. Higher quantization values result in lower values for inter-view coding parameter $i_c$. The transmitted coefficients are divided
result_Q = -1;
for(m = number_of_views - max_number_of_reconstructed_views; m <=
number_of_views; m++){}
for(qp = min_qp; qp < max_qp; qp++){
  d = calculate_datarate(qp) + (m-1) * 
calculate_datarate(qp) * inter_view_parameter;
  if(d_target >= d && d > 0){
    Q = calculate_quality(qp, r_q, c, number_of_views,
    m);
    if (Q > result_Q){
      result_Q = Q;
      result_m = m;
      result_qp = qp;
      result_d = d;
    }
  }
}

Figure 7.5: Pseudo code of the optimization algorithm.

by the higher quantization value. Afterward all coefficients which are below zero
are set to zero. Due to the higher quantization values more coefficients are set to
zero after division and rounding. The resulting zeroed block of coefficients can be
efficiently coded. Therefore a lower data rate of the additional views in comparison
to the base view is needed. A comparison of different calculated data rates with
and without the use of the inter-view coding parameter $i_c$ is shown in Table 7.3.
For quantization values which are lower than 20 an inter-view coding parameter
of 0.9 results in a calculated data rate which is lower than the real data rate since
the inter-view coding is less efficient and the resulting inter-view coding parameter
has to be higher for better results. But in this quantization range the data rate is

<table>
<thead>
<tr>
<th>qp</th>
<th>measured data rate</th>
<th>with $i_c$</th>
<th>without $i_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2711</td>
<td>3000</td>
<td>3288</td>
</tr>
<tr>
<td>32</td>
<td>955</td>
<td>1132</td>
<td>1240</td>
</tr>
<tr>
<td>40</td>
<td>430</td>
<td>504</td>
<td>552</td>
</tr>
<tr>
<td>48</td>
<td>241</td>
<td>256</td>
<td>280</td>
</tr>
</tbody>
</table>

Table 7.3: Measured and calculated data rate (in kBit/s) with and without use of inter-
view coding parameter $i_c = 0.9$ for different quantization values $qp$ and a
number of eight views.
very high and a reconstruction is not efficient.

Furthermore if the given data rate is low, the algorithm will choose more views to reconstruct than for a higher data rate since the quality of the encoded views is already low, such that the reconstruction causes lower degradation than for the encoded views with higher quality. This quality difference can be measured by comparing the reconstructed image with the encoded image and not with the original image. The quality compared with the encoded image is much higher than with the original image and the client can only compare these two images since the original image is only available at the server side. For a quantization value between 24 and 32 the highest quality is achieved since the quantization is high enough to reduce the camera noise in the image but not high enough to produce additional artifacts. The SSIM values of the reconstructed image for different quantization values and compared to the original image and the encoded image are shown in Figure 7.6.

For the client the encoded image denotes the reference image because this is the input image. A comparison of the reconstructed image with this reference image for the calculation of the encoding parameter and for the quality of the views after reconstruction is reasonable. But to get the quality of the MVC stream the images,
encoded and reconstructed, have to be compared to the real original image. The averaged quality of all views compared to the original images denotes the quality of the MVC stream.
Chapter 8

Results

To test our optimization algorithm we used the ballet and the breakdancer scene and different target data rates. Since our reconstruction algorithm is far from being optimal our optimization algorithm decides that a reconstruction is only reasonable for very low data rates. For high data rates there is enough data rate available to encode all views with a low quantization parameter and achieve a high quality. Since our reconstruction algorithm produces streams with lower quality the highest quality is achieved with only encoded views. For low data rates an encoding of all views is only possible with very high quantization. If the quantization yields a lower quality than the reconstruction algorithm the highest quality is achieved with a combination of encoded and reconstructed views. The optimization algorithm chooses which views have to be encoded and with which quantization parameter to achieve the highest quality for the given data rate. We had no other algorithm as reference available such that for a better reconstruction algorithm our results are only simulated. For our reconstruction quality the optimization algorithm chooses the encoding parameters shown in Table 8.1. These results can be verified by calculating the encoding quality for quantization parameters which result in a data rate below the target data rate. These values are shown in the last column in the table. For a target data rate of 1000 kBit/s a quantization parameter of 34 has to be used if all eight views are encoded. This results in a quality of 0.933 which is slightly below the achieved quality via reconstruction of one view and encoding of the remaining seven views with a quantization value of 32 which results in a quality of 0.934. In Figure 8.1 the encoded and reconstructed image for a quantization value of 32 and the encoded image for a quantization value of 34 are shown to verify the results above. The quality difference is so low that it is hard to notice differences visually.

For a reconstruction quality value of 0.96 instead of 0.94 more views can be reconstructed and a higher overall quality is achieved. But still only for relatively low data rates, see Table 8.2. For data rates above 2000 kBit/s an encoding of all views results in the highest quality.
Table 8.1: Results of the optimization algorithm for the ballet scene: needed data rate $d$, number of encoded views $m$, Quantization value $QP$, resulting quality $Q$ and encoding quality $Q_E$ for different target data rates $d_{target}$.

<table>
<thead>
<tr>
<th>$d_{target}$</th>
<th>$d$</th>
<th>$m$</th>
<th>$QP$</th>
<th>$Q$</th>
<th>$Q_E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>468</td>
<td>5</td>
<td>36</td>
<td>0.92</td>
<td>0.91</td>
</tr>
<tr>
<td>800</td>
<td>767</td>
<td>6</td>
<td>33</td>
<td>0.929</td>
<td>0.928</td>
</tr>
<tr>
<td>1000</td>
<td>990</td>
<td>7</td>
<td>32</td>
<td>0.934</td>
<td>0.933</td>
</tr>
<tr>
<td>2000</td>
<td>1998</td>
<td>8</td>
<td>27</td>
<td>0.946</td>
<td>0.946</td>
</tr>
<tr>
<td>5000</td>
<td>4276</td>
<td>8</td>
<td>22</td>
<td>0.952</td>
<td>0.952</td>
</tr>
</tbody>
</table>

Table 8.2: Results of the optimization algorithm for the ballet scene for a reconstruction algorithm with reconstruction quality value of 0.96: needed data rate $d$, number of encoded views $m$, quantization value $QP$, resulting quality $Q$ and encoding quality $Q_E$ for different target data rates $d_{target}$.

<table>
<thead>
<tr>
<th>$d_{target}$</th>
<th>$d$</th>
<th>$m$</th>
<th>$QP$</th>
<th>$Q$</th>
<th>$Q_E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>469</td>
<td>4</td>
<td>34</td>
<td>0.935</td>
<td>0.91</td>
</tr>
<tr>
<td>1000</td>
<td>891</td>
<td>4</td>
<td>28</td>
<td>0.941</td>
<td>0.933</td>
</tr>
<tr>
<td>2000</td>
<td>1973</td>
<td>7</td>
<td>24</td>
<td>0.947</td>
<td>0.946</td>
</tr>
</tbody>
</table>

If the reconstruction quality value is above 0.97 a reconstruction increases the quality for data rates up to 10 000 kBit/s, see Table 8.3. Since we have no reconstruction algorithm with these quality values available the described results for these configurations have to be validated with different reconstruction algorithms in a future study.

Another question is whether the overall quality changes for different frames of the video stream. For our reconstruction algorithm there is no significant influence on
the quality by the different frames of the video sequence. In Figure 8.2 the SSIM value of the different frames is plotted. Since the frames have different types (I-, P-, and B-Frames) the SSIM values slightly differ. The SSIM values are in a range of 0.967 to 0.971 which can be treated as relatively constant over the frames. For low quantization values the quality changes over different frames in the range between 0.953 to 0.962, see Figure 8.3.

Since other sequences have more motion in background than the ballet scene, we checked whether our reconstruction quality is comparable for other scenes. We used the breakdance scene which is also freely available and which has much more

### Table 8.3: Results of the optimization algorithm for the ballet scene for a reconstruction algorithm with reconstruction quality value of 0.97: needed data rate $d$, number of encoded views $m$, quantization value $QP$, resulting quality $Q$ and encoding quality $QE$ for different target data rates $d_{\text{target}}$.

<table>
<thead>
<tr>
<th>$d_{\text{target}}$</th>
<th>$d$</th>
<th>$m$</th>
<th>$QP$</th>
<th>$Q$</th>
<th>$QE$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>4743</td>
<td>6</td>
<td>20</td>
<td>0.953</td>
<td>0.952</td>
</tr>
<tr>
<td>8000</td>
<td>7268</td>
<td>7</td>
<td>19</td>
<td>0.954</td>
<td>0.953</td>
</tr>
<tr>
<td>10 000</td>
<td>9668</td>
<td>6</td>
<td>17</td>
<td>0.956</td>
<td>0.955</td>
</tr>
</tbody>
</table>

Figure 8.2: SSIM values of different frames of the ballet sequence.
motion in background areas. For a quantization value of 24 and a GOP size of 8, we achieve an SSIM value of 0.889 for the ballet scene and an SSIM value of 0.875 for the breakdance scene, see Figure 8.4. The quality difference appears because there is more variation in the background area in the breakdance scene which makes it more difficult to separate between foreground and background. This results in more errors by the reconstruction and therefore a lower quality. The highest quality in both scenes is achieved for a quantization value of 24 since the camera noise is mostly reduced for this quantization value. Furthermore there are less additional artifacts from the quantization. The curves for ballet and breakdance scene have a similar behavior and for quantization values above 48 the quality for both scenes is worse.

For the breakdance scene the optimization algorithm returns different configurations for the above used data rates. In Table 8.4 the results for a reconstruction quality value of 0.94 are shown. Since the breakdance scene has more inter-view coding this parameter has to be adjusted to 0.75 in order to calculate a correct data rate. Since the breakdance scene needs much more data rate the reconstruction results in a higher quality for data rates up to 1000 kBit/s. The highest difference in quality between the optimized and the encoded configuration is still for low data rates. The values for the encoding quality $Q_E$ are calculated for the lowest
Figure 8.4: SSIM values of the ballet and breakdance scene for different quantization values.

<table>
<thead>
<tr>
<th>$d_{target}$</th>
<th>$d$</th>
<th>$m$</th>
<th>$QP$</th>
<th>$Q$</th>
<th>$Q_E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>478</td>
<td>4</td>
<td>36</td>
<td>0.895</td>
<td>0.883</td>
</tr>
<tr>
<td>800</td>
<td>784</td>
<td>6</td>
<td>35</td>
<td>0.902</td>
<td>0.898</td>
</tr>
<tr>
<td>1000</td>
<td>979</td>
<td>6</td>
<td>33</td>
<td>0.907</td>
<td>0.905</td>
</tr>
<tr>
<td>2000</td>
<td>1869</td>
<td>8</td>
<td>30</td>
<td>0.921</td>
<td>0.921</td>
</tr>
</tbody>
</table>

Table 8.4: Results of the optimization algorithm for the breakdance scene for a reconstruction algorithm with reconstruction quality value of $0.94$: needed data rate $d$, number of encoded views $m$, quantization value $QP$, resulting quality $Q$ and encoding quality $Q_E$ for different target data rates $d_{target}$.

Quantization values for which the data rate of eight views is below the target data rate.

Another improvement of the quality and the usage of the data rate is to divide the MVC stream into parts which can be adjusted separately. A video stream which is divided into parts has several advantages. Data rate variations can be adjusted for the following parts by the optimization algorithm. If the data rate significantly decreases, the quantization value or the number of views which are reconstructed are increased to stay below the target data rate. On the other side
a higher available data rate results in lower quantization values for the following parts of the MVC stream. Since this adjustment is done on the actual behavior of the available data rate the highest quality can be achieved with the variation of the encoding parameters for the different parts.
Chapter 9

Conclusion and future work

We developed an optimization algorithm to calculate the maximal quality of an MVC stream for a limited available data rate. The optimization algorithm uses view interpolation to reduce the number of transmitted views. The saved data rate is used to encode the remaining views with less quantization. Since the reconstruction of a view at the client side has a lower quality than the encoded one, the optimization algorithm calculates the optimal set of encoded and reconstructed views and the corresponding encoding parameters to achieve the maximal quality for the available data rate. Also the case of limited computation power at the client side and a resulting maximal number of possible view reconstructions are included in the optimization algorithm. To calculate the optimal parameter set, the optimization algorithm needs the reconstruction quality of the reconstruction algorithm. For our reconstruction algorithm this value is far from being optimal at least according to widespread metrics like PSNR or SSIM. Therefore a high coverage of the reconstructed view by the encoded views is needed and the reconstruction is only efficient for very low data rates. Since we have no better reconstruction algorithms available, we can only simulate the values for higher reconstruction qualities. With these simulations we show that the better the reconstruction quality of the algorithm is the more efficient is the use of less encoded views to maximize the overall quality of the MVC stream for a limited data rate. For higher data rates the use of encoded views is more efficient than using reconstructed ones.

In a future work different reconstruction algorithms have to be used to verify the optimization algorithm and to improve the usage of reconstruction for higher data rates as well. Since the available data rates are constantly increasing the reconstruction algorithm also has to be improved to achieve a higher quality. Furthermore the simulated results for high quality reconstruction algorithms can be verified by real measurements if algorithms with higher quality values are available.
Furthermore the reconstruction algorithm needs much computation time at the client side to generate the missing views and the calculation of the parameter set by the optimization algorithm needs prior computation time. After the calculation is finished, the encoding of the different views is started.

The additional time needed for reconstruction can be decreased by paralleling parts of the reconstruction pipeline. Parts of the code where only single pixels and not a combination of these or the whole image is needed can run in parallel. The projection of pixels to the different cameras is such a part. Also, the optimization algorithm can be optimized by paralleling independent parts to reduce the time needed.

Another problem is the validity of the used metrics. In our results we show that there is a very small quality difference measured in SSIM between our optimization and an encoding of all views. The encoding of all views has to use a higher quantization value since the limited data rate is consumed by all views. In our optimization we reconstruct one view and encode the remaining views with a lower quantization value since the data rate is consumed by one view less. If we compare the images visually we can see slight differences but it is hard to identify the image which has the highest quality according to the metrics used. In a future work, subjective quality tests in which users evaluate the different images are needed to prove or disprove the validity of the quality metrics used.
Bibliography


[18] Kwan-Jung Oh, Sehoon Yea, and Yo-Sung Ho, “Hole-filling method using depth based in-painting for view synthesis in free viewpoint television (ftv) and 3D video” (2009).

