3.4 Multiple-view depth estimation

3.4.1 Depth estimation for multi-view video coding

The previous section has presented a method that estimates disparity images using two rectified views. Similar approaches [43, 107] were recently investigated to create depth maps for 3D-TV systems. However, in those applications, a pair of images was used to compute the disparity image. In multi-view video, multiple images are available so that an efficient depth estimation algorithm should employ all views available. More specifically, the disadvantages of a pairwise depth estimation algorithm are threefold. First, while multiple views of the scene are available, only two rectified views are employed simultaneously, which degrades performance. Second, because depth images are estimated pairwise, consistency of depth estimates across the views is not enforced. Third, multiple pre-processing (image rectification) and post-processing (disparity-to-depth conversion and image de-rectification) procedures are necessary.

In order to circumvent these three disadvantages, we propose an approach that employs multiple views and estimates the depth using a correlation table similar to the Disparity Space Image (DSI). As in the previous section, the depth is calculated along each scanline using the dynamic programming algorithm from the previous section. Two novel constraints are introduced which are enforcing smooth variations of depth across scanlines and across the views, thereby positively addressing the aforementioned disadvantages. Additionally, no unnecessary conversion from disparity pixels to depth values is required.

Let us now explain the multi-view depth-estimation algorithm that was introduced previously. The presented algorithm calculates a depth image corresponding to a selected reference view k_r in two steps. In a first step, the 3D world coordinate $(X,Y,Z_c)^T$ of a pixel $p_{k_r} = (x_{k_r},y_{k_r},1)^T$ (from the reference view) is calculated using a candidate depth value, e.g., Z_c. The 3D world coordinate can be calculated using the back-projection relation (see Equation (2.20)) and be written as

$$\begin{pmatrix} X \\ Y \\ Z_c \end{pmatrix} = \boldsymbol{C}_{\boldsymbol{k_r}} + \lambda_{k_r} \boldsymbol{R}_{\boldsymbol{k_r}}^{-1} \boldsymbol{K}_{\boldsymbol{k_r}}^{-1} \boldsymbol{p}_{\boldsymbol{k_r}}, \quad (3.15)$$

where C_{k_r}, R_{k_r}, K_{k_r} correspond to the parameters of the selected reference camera k_r. It should be emphasized that the 3D point $(X,Y,Z_c)^T$ is described in the world coordinate system that is shared by all cameras. In a second step, the 3D point is projected onto each neighboring view k, providing a corresponding pixel $(x_k,y_k,1)^T$ for each view for which:

$$\lambda_k \begin{pmatrix} x_k \\ y_k \\ 1 \end{pmatrix} = \boldsymbol{P}_{\boldsymbol{k}} \begin{pmatrix} X \\ Y \\ Z_c \end{pmatrix}, \quad (3.16)$$

where P_k represents the camera parameters of the neighboring camera k. These two steps are repeated for several depth candidates Z_c, so that a corresponding similarity can be measured and stored in a correlation table $T(x,y,Z_c)$. The algorithm is illustrated by Figure 3.13. It can be noted that this correlation table is similar to the DSI data structure. In practice, an entry in the correlation table $T(x,y,Z_c)$ is calculated using the sum of the correlation measures over all image views I, specified by

$$T(x,y,Z_c) = \sum_{k:k\neq k_r} \sum_{(i,j)\in W} |I_{k_r}(x-i,y-j) - I_k(x_k-i,y_k-j)|. \quad (3.17)$$

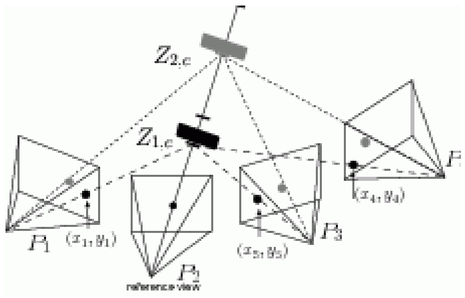


Figure 3.13 Given a reference pixel (black dot), the algorithm tests multiple depth candidates. In this example, the depth candidate $Z_{2,c}$ yields a grey color in the neighboring views and is therefore not consistent with the black color of the reference pixel. The candidate $Z_{1,c}$ yields a consistent black color and for this reason, it should be selected as the final depth value.

Using a WTA optimization strategy, the depth value that yields the most consistent (or similar) color across the views is selected as the final estimated depth value. The pseudocode of the algorithm using a WTA optimization is summarized in Algorithm 2.

Algorithm 2 Multiple view depth estimation using the WTA optimization

```
Require: All camera projection matrices $P_k$
Require: Select view $k_r$ for which the depth image should be computed.
for $(y = 1; y \le height; y++)$ do
  for $(x = 1; x \le width; x++)$ do
  for $Z_c \in [Z_{min}; Z_{max}]$ do
  $(X,Y,Z_c) = C_{k_r} + \lambda_{k_r} R_{k_r}^{-1} K_{k_r}^{-1} (x,y,1)^T$
  $T(x,y,Z_c) = 0$
  for each neighboring camera k do
  $\lambda_k \cdot (x_k,y_k,1)^T = P_k \cdot (X,Y,Z_c,1)^T$
  $T(x,y,Z_c) += \sum_{(i,j)\in W} |I(x-i,y-j) - I_k(x_k-i,y_k-j)|$
  end for
  end for
  $Z(x,y) = \arg\min_{Z_c \in \{Z_{min},\ldots,Z_{max}\}} T(x,y,Z_c)$
  end for
  end for
```

In the previous section, we have seen that the WTA optimization yields inaccurate depth estimates and noisy depth images. Therefore, we have experimented earlier with a dynamic programming optimization strategy to enforce smooth variations of the depth, which has led to streaking artifacts. In the next section, we impose two additional constraints to obtain more accurate depth estimates.

3.4.2 Two-pass optimization strategy

To enforce smooth depth variations within the image and between the images, we present two novel constraints: (1) smooth variations of depth values across image scanlines and (2) consistent depth values across the views. To combine these constraints with the dynamic programming algorithm, we employ two penalty costs, namely an *inter-line penalty cost* and an *inter-view penalty cost* that both extend the cost function E(D) previously defined by Equation (3.11). The proposed algorithm estimates a depth image by minimizing two different objective cost functions in two passes. In a first pass, a depth image is estimated for each view with the constraint that consecutive image scanlines should show smooth depth variations across the lines. The purpose of this pass is to generate one initialization depth image for each view without streaking artifacts. We denote the objective function for one view at the first pass as

$$E_{pass\ 1}(D) = E(D) + E_{smooth\ 1}(D), \quad (3.18)$$

where $E_{smooth\ 1}(D)$ is defined in the following paragraph. This computation is repeated for each view. In a second pass, a depth image is re-estimated for each view with the constraint that an object should have consistent world depth values in all depth images estimated during the first pass. The aim of this second pass is to refine the initialization depth images from the first pass such that the final depth images are consistent across the views. We denote the objective function at the second pass as

$$E_{pass\ 2}(D) = E(D) + E_{smooth\ 2}(D), \quad (3.19)$$

where $E_{smooth\ 2}(D)$ is also defined in the sequel.

Pass 1. Consistent depth values across scanlines.

To enforce smooth variations of depth values across scanlines, we introduce an inter-line penalty cost. Practically, the objective cost function integrating the inter-line penalty cost can be written as

$$E_{smooth\ 1}(D) = \sum_x \underbrace{\kappa_l \cdot |Z_{k_r}(x,y) - Z_{k_r}(x,y-1)|}_{\text{inter-line penalty cost}}, \quad (3.20)$$

where D is the vector of depth values $Z_{k_r}(x,y)$ along a scanline, $Z_{k_r}(x,y)$ the estimated depth at position (x,y) in the reference image with index k_r and κ_l the inter-line penalty parameter (typically a scalar). The index of the reference view k_r varies between $1 \leq k_r \leq N$, for an N-view data set.

Pass 2. Consistent depth values across the views.

In a second pass, the depth is refined for each view by incorporating a cost that penalizes inconsistent depth values across the views into the objective cost function $E_{smooth\ 2}(D)$ (see Figure 3.14). Similar to the first constraint, the objective function that integrates the inter-line and inter-view penalty costs is written as

$$\begin{aligned} E_{smooth\ 2}(D) \quad &= \quad \sum_x \Bigg(\kappa_l \cdot |Z_{k_r}(x,y) - Z_{k_r}(x,y-1)| + \qquad (3.21) \\ & \underbrace{\sum_{k,k\neq k_r} \kappa_v \cdot |Z_{k_r}(x,y) - Z_k(x_k,y_k)| \Bigg)}_{\text{inter-view penalty cost}}, \qquad (3.22) \end{aligned}$$

where κ_v corresponds to the inter-view penalty parameter. The depth $Z_{k_r}(x,y)$ and $Z_k(x_k,y_k)$ correspond to the depth in the reference view with index k_r and neighboring views k, respectively. It should be noted that both depth values $Z_{k_r}(x,y)$ and $Z_k(x_k,y_k)$ are described in the world coordinate system, thereby enabling a direct comparison. Finally, to determine the value of the depth $Z_k(x_k,y_k)$ in the neighboring views, the pixel (x,y) is projected onto the neighboring image planes k, which provides the pixel position (x_k,y_k) and the corresponding depth $Z_k(x_k,y_k)$ that was calculated during the first pass.

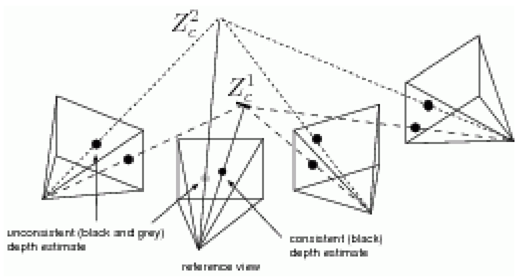

Z_c^2
Z_c^1
unconsistent (black and grey) depth estimate
consistent (black) depth estimate
reference view

Figure 3.14 In this example, a correct depth value (black dot in the reference view) is consistent across the views. Alternatively, an incorrect depth value (grey dot in the reference view) leads to inconsistent depth values across the views.

Admissible path in the DSI.

To optimize the previous objective functions $E_{pass\ 1}(D)$ and $E_{pass\ 2}(D)$, the dynamic programming algorithm is employed. In Section 3.3.3, we have seen that the optimal path in the DSI is derived from the ordering constraint. However, the ordering constraint cannot be employed in the extended case of multiple-view depth estimation because no left or right camera position is assumed. We therefore employ an admissible edge that enables an arbitrary increase or decrease of depth values. The admissible edges in the DSI are portrayed by Figure 3.15.

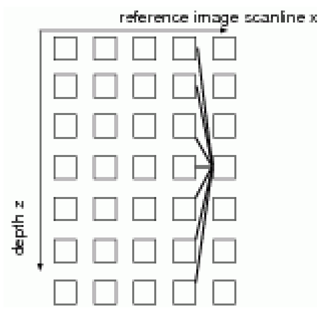


Figure 3.15 Because the ordering constraint cannot be employed in the extended case of depth estimation with multiple views, the admissible edges illustrated above are used, allowing all possible transitions.

3.4.3 Depth sampling of the 3D space

For 3D sampling, it should be noted that there exists a dependency between the depth of an object and its corresponding resolution in the image. For example, far and near objects are seen as small and large in the image, respectively. A near object can therefore be observed with higher resolution than a far object so that the depth resolution of a near object should be higher than that of a far object. In this context, we employ the plenoptic sampling framework [12]. The idea of the plenoptic sampling framework is to employ a non-linear quantization of the depth between a minimum and maximum depth Z_{min} and Z_{max} (see Figure 3.16).

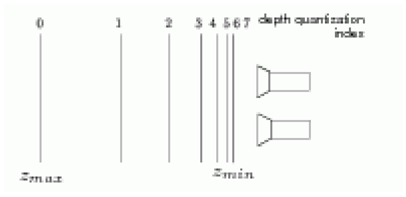


Figure 3.16 Non-linear depth sampling of the 3D space.

More specifically, the depth Z_i for a quantization index i can be written as

$$\frac{1}{Z_i} = \eta_i \frac{1}{Z_{min}} + (1-\eta_i)\frac{1}{Z_{max}}, \quad (3.23)$$

with

$$\eta_i = \frac{i-0.5}{N_d} \text{ and } i = 1, ..., N_d, \quad (3.24)$$

where N_d corresponds to the maximum number of depth values/layers. Alternatively, the quantization index i of a depth value Z_i can be written as

$$i = \lfloor N_d \frac{Z_i^{-1} - Z_{max}^{-1}}{Z_{min}^{-1} - Z_{max}^{-1}} + 0.5 \rfloor. \quad (3.25)$$

The depth of a pixel can be therefore efficiently stored and represented using the corresponding quantization index i. In the case that 256 possible depth values are employed, the depth quantization index can be represented using an 8-bit pixel format and stored using a standard 8 bits per pixel image.

3.4.4 Depth image post-processing

Having calculated a depth image for each view, an optional task is to apply a noise-reduction algorithm as a last post-processing step. Typically, this noise-reduction stage noticeably improves the quality of depth images at a limited computational cost. For example, based on the assumption that depth pixels vary smoothly at the object surfaces, a simple noise-reduction algorithm consists of filtering noisy pixels that have low spatial correlation with neighboring pixels. However, the assumption of smoothly varying depth pixels is invalid at object borders (edges). Therefore, the challenge is to filter noisy pixels (outliers) while preserving the accuracy of depth pixels along object borders. To implement this concept, we propose in this section a depth-image denoising algorithm that actively supports smooth regions in the depth image while preserving sharp depth discontinuities along the object borders.

The proposed noise-reduction algorithm proceeds in two stages. At the first stage, a color segmentation is applied to the texture image that corresponds to the considered depth image. The intuitive idea underlying this first stage is that the resulting segments, which delineate the object contours, correspond to a piece/portion of the object surfaces. Because object surfaces show smooth regions in the depth image, the corresponding depth image segments can be approximated by a piecewise-linear function [97]. Figure 3.17 shows an example of a color-segmented image obtained using the watershed segmentation algorithm [84].

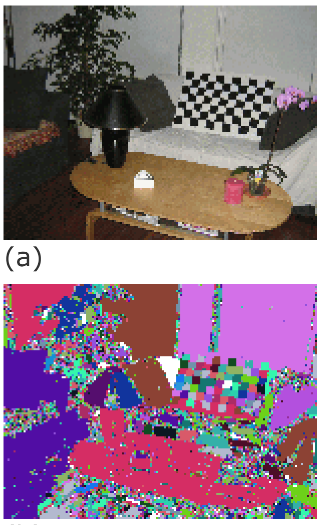
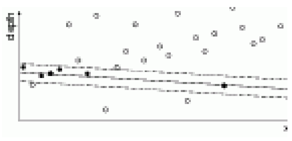


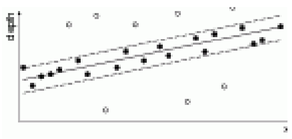
Figure 3.17 (a) Original texture image and (b) corresponding segmented image using a

watershed segmentation algorithm. Note that the depth of each segment, e.g., the background wall, can be modeled by a piecewise-linear function.

At the second stage, a piecewise-linear function which models the object surfaces is estimated for each image segment. To calculate the linear function parameters of each segment, we apply a robust estimation method, i.e., the RANdom SAmple Consensus (RANSAC) algorithm [30]. The advantage of this algorithm is that it can estimate parameters even if the data is disturbed by a large number of outliers, i.e., noisy depth pixels. Because we assume that the data fits to a piecewise-linear function, three pixels are sufficient to calculate a plane and thus the corresponding parameters. The RANSAC algorithm first randomly selects three pixels, which are used to compute candidate parameters of the linear (planar) function. The algorithm then counts the number of pixels that fit to this model (inliers). This procedure is repeated several times and finally, the parameters with the largest support are selected as the parameters for the considered image segment (see a one-dimensional example in Figure 3.18). The robust RANSAC estimation of piecewise-linear function parameters is then performed for all image segments. The advantages of the depth image post-processing step are twofold. First, because the depth is accurately estimated especially at object borders, a high-quality image can be rendered (this property of image rendering will be discussed in Chapter 4). Second, the resulting depth image presents smoothly varying properties and thus can be encoded efficiently. Note that this property will be exploited in Chapter 6 for the coding of depth images.



(a)



(b)

Figure 3.18 Two samples are randomly selected to compute a candidate model indicated by the solid line. Input data close to the line candidate is considered as inliers (black dots between dotted lines). Fig. 3.18(a) and Fig. 3.18(b) show two candidate models with a small and a high number of inliers, respectively. The model with the largest number of inliers is selected.

3.4.5 Experimental results

In this section, we present experimental results evaluating the accuracy of the depth estimation algorithm from the previous section. We especially focus on the challenging problem of estimating depth images using texture-less and colorless multi-view images. For this reason, we employ the multi-view images “Breakdancers”, “Ballet” and “Livingroom”, featuring these properties. Note that because the calibration parameters are necessary for multi-view depth estimation, the depth images “Teddy” and “Cones” are not estimated (no parameters available). For each presented depth image, the parameters are empirically determined and not specifically adapted to each data set. We first provide an objective evaluation and, second, proceed by subjectively assessing the depth image quality. Finally, we compare the quality of the calculated depth images with two alternative methods, based on a two-dimensional optimization strategy.

3.4.5.0 A. Objective evaluation

For the objective evaluation of the depth image quality, we employ an image rendering method that synthesizes virtual views using the calculated depth image. This technique is based on the notion that the quality of the rendered image depends on the accuracy of the estimated depth image. Specifically, the more accurate a depth image is, the higher the rendering quality becomes. In our experiments, the synthesis of virtual images is carried out using the relief texture rendering algorithm, which will be presented in Chapter 4. In the presented experiments, we use a post-processing step as defined in Section 3.4.4. To evaluate the quality of rendered images, a synthetic image is rendered at the same position and orientation of a selected reference camera [4]. By comparing the synthetic image I_s with the reference image I_r, the Peak Signal-Noise Ratio ($PSNR_{rs}$) can be calculated by

$$PSNR_{rs} = 10 \cdot \log_{10}\left(\frac{255^2}{MSE_{rs}}\right), \quad (3.26)$$

where the Mean Squared Error (MSE) is computed by the following equation:

$$MSE_{rs} = \frac{1}{w \cdot h}\sum_{i=1}^{w}\sum_{j=1}^{h}||I_r(i,j) - I_s(i,j)||^2, \quad (3.27)$$

with w and h corresponding to the width and the height of the images, respectively. Note that the $PSNR_{rs}$ metric has shown some limitations for evaluating the image quality because it does not match the human subjective perception. However, this is the only simple alternative image quality metric that has been widely accepted and used by the research community (also adopted in key literature [50] on rendering). Besides computing this metric, we will also comment on the subjective depth image quality. The differences between the objective ($PSNR_{rs}$) and subjective rendering quality will be illustrated by Figure 6.11 in Chapter 6.

The discussed results are compared in a relative sense to the conventional algorithm addressed earlier in this chapter. Up till now, results on multi-view depth estimation in literature are still being presented in a subjective discussion. To avoid the ambiguity of such a discussion, we have presented here a well-defined measurement for estimating the quality of depth images. This indirect method, based on rendering quality evaluation, gives a less precise evaluation of the depth quality than a direct method based on a comparison with, e.g., a ground-truth depth image. However, calibrated ground-truth multi-view depth images are presently not available to the research community. Additionally, it may be argued that the image rendering quality is more important than the quality of the depth images because depth images are not visualized by the user (as opposed to the rendered images).

	Two-view estimation	Multi-view estimation
“Ballet”	25.4 dB	28.3 dB
“Breakdancers”	29.2 dB	32.4 dB
“Livingroom”	23.5 dB	23.8 dB

Table 3.1 Image rendering quality resulting of two depth-estimation algorithms: (1) the two-view depth estimation and (2) the multiple-view depth estimation. Both methods are based on dynamic programming optimization.

Let us now discuss the rendering-quality measurements summarized in Table 3.1. First, it can be seen that the proposed algorithm that uses multiple views to estimate depth images consistently and significantly outperforms the two-view depth-estimation algorithm. For example, an image-quality improvement of 2.9 dB, 3.2 dB and 0.3 dB can be observed for the “Breakdancers”, “Ballet” and “Livingroom” sequences, respectively. The lower improvement for the “Livingroom” sequence is mainly due to an inaccurate texture rendering of the scene, and especially, an inaccurate rendering of the background tree. Considering the applications of 3D-TV and free-viewpoint video, the contributed two-pass algorithm clearly enables high-quality rendering, thereby facilitating the reconstruction of a high-quality depth signal.

3.4.5.0 B. Subjective evaluation

Figures 3.19, 3.20 and 3.21 enable a discussion on the subjective quality of the estimated depth images using the proposed two-pass algorithm. We assess the quality of the depth images *prior to the post-processing step*. First, it can be observed that, prior to the post-processing step, the two-pass algorithm leads to more consistent depth values across the lines, so the annoying streaking artifacts are reduced leading to a significant perceptual improvement. For example, observing Figure 3.19(b), it can be noted that the two-pass algorithm leaves over little streaking artifacts in the image. Similar conclusions can be drawn by observing Figure 3.20(b) and Figure 3.21(b), albeit there is still fuzzyness around the objects. Second, it can be noted that the depth of colorless and texture-less regions can be accurately estimated. For example, the depth values of the texture-less walls in Figure 3.19(b) and Figure 3.21(b) are correctly estimated and do not suffer from streaking artifacts. Additionally, the depth of very thin objects such as the flower stalk, can be correctly estimated (see Figure 3.19(b)). Such an accurate result is obtained by using multiple views simultaneously and by enforcing consistent depth values across the views. However, several estimation errors occur at object boundaries which correspond to depth discontinuities (e.g., borders of the dancer in Figure 3.20(b)). Let us now assess the quality of the depth images *after the post-processing step*. As expected, it can be seen in Figures 3.19(c), 3.20(c) and 3.21(c) that the post-processed depth images depict piecewise-linear regions delineated by sharp boundaries. For example, Figure 3.19(c) shows that the depth along the border of the table is accurately delineated. Similar conclusions can be drawn by comparing Figure 3.20(b) with Figure 3.20(c) and Figure 3.21(b) with Figure 3.21(c). Thus, the introduced post-processing step clearly removes noisy pixels while preserving the edges along objects in the depth image.

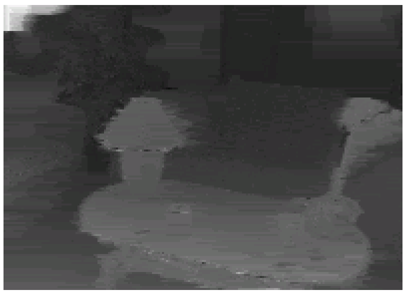
3.4.5.0 C. Subjective comparisons with alternative methods

Next, we have compared the quality of the calculated depth images with two alternative methods based on a two-dimensional optimization strategy (see Section 3.3.3). Note that only a few alternative contributions exist, employing the complex “Breakdancers” and “Ballet” multi-view sequences. We have found only two proposals that employ either both sequences [109], or one sequence only (“Breakdancers”) [16].

The first alternative proposal is based on a modified plane sweeping algorithm [16]. By subjectively comparing Figure 3.20(c) and Figure 3.22(a) it can be seen that the method proposed in this thesis results in a slightly more accurate depth image [5]. For example, the competing method generates some depth artifacts in the occluded region along the arm of the breakdancer. Additionally, the proposed method enables an accurate depth estimation along the delineation between the background wall and the rightmost person (see Figure 3.20(c)). At the opposite, the competing method does not allow the visualization of the discussed depth discontinuity (along the person delineation).

The second alternative proposal [109] performs an over-segmentation of the texture image, and calculates the depth of each segment using a belief propagation algorithm. For the “Breakdancers” sequence, it can be seen that both methods lead to a similar depth image quality. For example, the depth discontinuity between the breakdancer and the background wall is accurately estimated in both cases (see Figure 3.20(c) and Figure 3.22(b)). However, for the “Ballet” sequence, it can be noticed that the proposed method yields a less accurate depth image. For example, the depth discontinuity along the foreground person is better calculated by the alternative proposal (see Figure 3.22(c)).

(a)


(b)

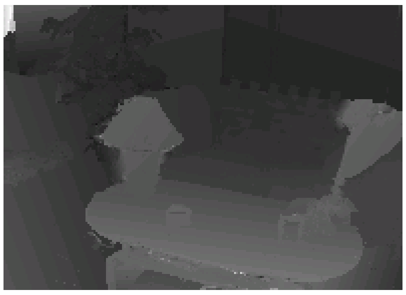
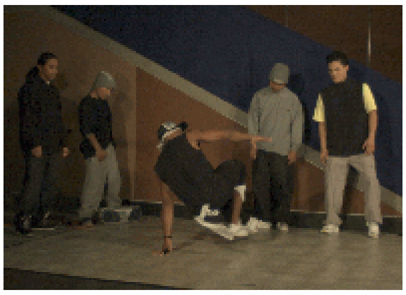
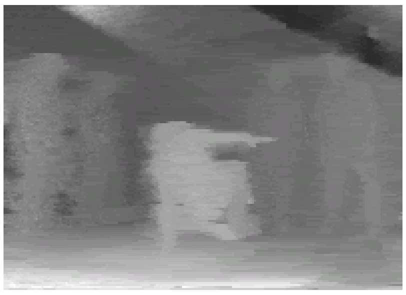

(c)

Figure 3.19 (a) View 3 of the “Livingroom” sequence. Estimated depth images (b) prior to post-processing and (c) after the post-processing step.


(a)


(b)

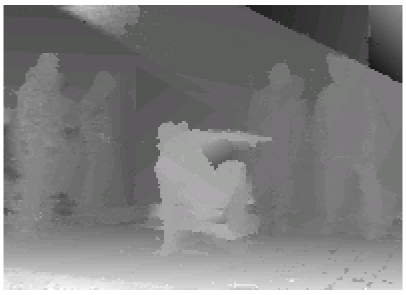
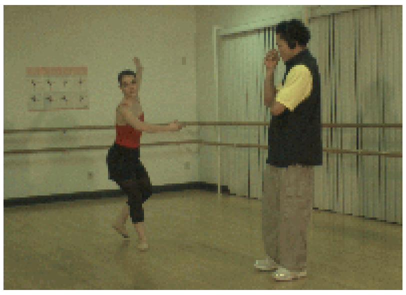
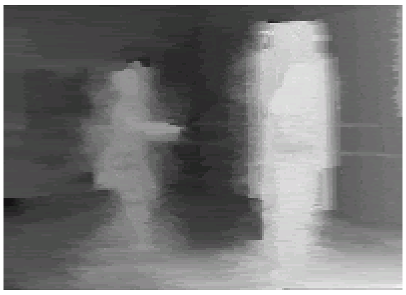

(c)

Figure 3.20 (a) View 3 of the “Breakdancers” sequence. Estimated depth images (b) prior to post-processing and (c) after the post-processing step.


(a)


(b)

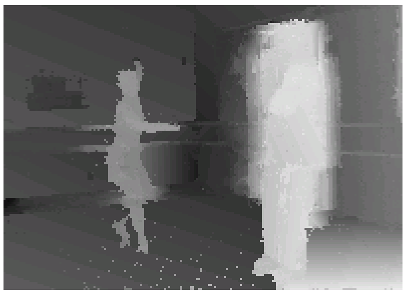
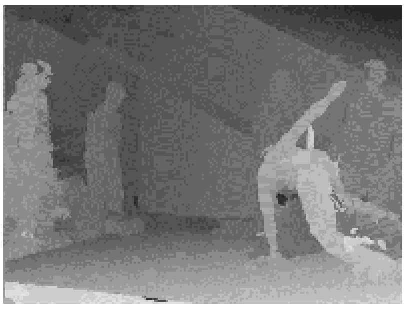
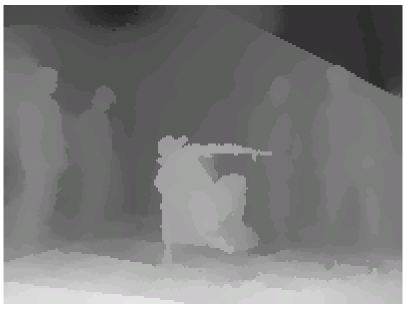

(c)

Figure 3.21 (a) View 3 of the “Ballet” sequence. Estimated depth images (b) prior to post-processing and (c) after the post-processing step.


(a)


(b)

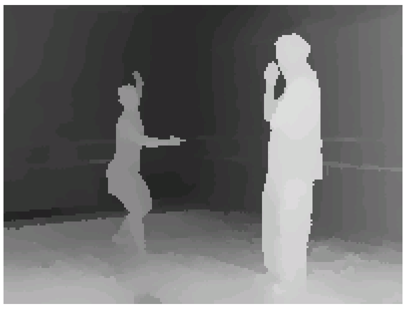

(c)

Figure 3.22 Depth images calculated using the algorithm based on (a) plane-sweeping [16] and (b) (c) over-segmentation [110].

[4]This selected reference view is left out from the data set for rendering.

[5]The discussed alternative method [16] presents the depth image of a different viewpoint and time. However, the properties of the sequence do not vary over time and across the views, so that a subjective comparison is still possible.

[next] [prev] [prev-tail] [front] [up]

yannick.morvan.public@gmail.com