# RATE-DISTORTION OPTIMIZED 3D RECONSTRUCTION FROM NOISE-CORRUPTED MULTIVIEW DEPTH VIDEOS

*Wenxiu Sun*[*], *Gene Cheung*[†], *Philip A. Chou*[§], *Dinei Florencio*[§], *Cha Zhang*[§], *Oscar C. Au*[*]

[*] Dept. of Electronic and Computer Engineering, The Hong Kong University of Science and Technology
[†] National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan
[§] Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA

## ABSTRACT

Transmitting compactly represented geometry of a dynamic scene from a sender can enable a multitude of 3D imaging functionalities at a receiver, such as synthesis of virtual images from freely chosen viewpoints via depth-image-based rendering (DIBR). While depth maps can now be readily captured using inexpensive depth sensors, they are often corrupted by non-negligible acquisition noise. In this paper, we derive 3D surfaces of a dynamic scene from noise-corrupted depth maps in a rate-distortion (RD) optimal manner. Specifically, unlike previous work that finds the most likely (e.g., maximum likelihood) 3D surface from noisy observations regardless of representation size, we judiciously search for the best fitting (i.e., minimum distortion) 3D surface subject to a bitrate constraint. Our RD-optimal solution reduces to the maximum likelihood solution as the rate constraint is loosened. Using the MVC codec for compression of multiview depth video and MPEG free viewpoint test sequences as input, experimental results show that RD-optimized 3D reconstructions computed by our algorithm outperform unprocessed depth maps by up to $2.42$dB in PSNR of synthesized virtual views at the decoder for the same bitrate.

***Index Terms***— 3D reconstruction, depth image, rate-distortion optimization

## 1. INTRODUCTION

With the advent of depth capturing sensors [1] like Microsoft Kinect, depth images (per-pixel distances between objects in the 3D scene and capturing camera) can now be acquired cheaply from multiple viewpoints. Each depth map constitutes a projection of the 3D geometry in the scene to a 2D image of fixed resolution. Thus, having acquired depth maps from multiple viewpoint cameras, one can back-project them to the 3D space to (partially) recover the original 3D geometry. If the multiview depth maps—a representation of the 3D geometry—are compressed and transmitted, then the receiver can perform a range of 3D imaging tasks, such as synthesis of virtual images from freely chosen viewpoints using texture and depth maps of neighboring camera views via depth-image-based rendering (DIBR) [2].

This work was supported by Microsoft Research CORE program.

To enable quality communication of 3D geometry from sender to receiver, however, we are faced with two practical problems. The first is that acquired depth maps from typical depth sensors are corrupted by non-negligible acquisition noise. The second is that if the chosen representation of the 3D geometry requires too many encoding bits (not compact), then the communication cost will be prohibitively high.

In this paper, we address both aforementioned problems simultaneously by deriving a rate-distortion (RD) optimal 3D surface of the dynamic scene given noise-corrupted depth observations, one that is accurate *and* requires few bits for representation in multiview depth maps. Specifically, we search for the "best fitting" 3D surface—one that maximizes the posterior probability of the chosen 3D surface given noise-corrupted observations—subject to a bitrate constraint. Our RD-optimal solution reduces to the maximum likelihood (ML) solution (as commonly done in the computer vision literature [3, 4, 5] with no consideration for representation size) as the rate constraint is loosened. To solve this problem, we propose an optimization scheme that finds a locally optimal solution by iterating between two steps: i) align edges in depth maps of consecutive views to match *scene structure* across views; and ii) smooth texture within depth edges to match *scene texture* across views. Using the MVC codec [6] for compression of multi-view depth video and MPEG free viewpoint (FTV) test sequences as input, experimental results show that RD-optimized 3D surfaces computed by our algorithm outperform unprocessed depth maps by up to $2.42$dB in Peak Signal-to-Noise Ratio (PSNR) of synthesized virtual views at the decoder for the same bitrate.

The outline of the paper is as follows. We first discuss related work in Section 2 and overview our system model in Section 3. We present our problem formulation and corresponding optimization algorithm in Section 4 and 5, respectively. Finally, experimental results and conclusion are presented in Section 6 and 7, respectively.

## 2. RELATED WORK

The problem of finding the optimal 3D surface—one that is the most probable given noise-corrupted depth observations—has been studied extensively in the computer vision literature [3, 4, 5]. The optimally constructed 3D sur-

face, however, may require a large encoding overhead. One naïve approach to find a good rate-constrained 3D surface is to separate the problem into two steps: i) first find the most likely 3D surface (establishing ground truth) from noise-corrupted observation regardless of representation size; and then ii) perform conventional RD optimization as done in standard video codec like H.264 [7] given ground truth surface as input. We argue this is a sub-optimal approach; the problem of finding an RD-optimal 3D surface from noise-corrupted observations is inherently a *probabilistic* one—identifying the most likely 3D surface *within* the search space of surfaces with representation size no larger than a bit budget. By first establishing a ground truth signal and converting the problem to a *deterministic* one during RD optimization, the problem becomes finding the least distorted signal compared to the ground truth signal in the rate-constrained search space, which is *not* equivalent to the originally posed probabilistic problem. We will demonstrate empirically that our computed RD-optimized 3D surfaces outperform surfaces generated from this separation approach in Section 6.

In previous multiview depth map compression work, it has been observed that inconsistency among input depth maps of different views due to acquisition noise incurs expensive coding overhead, but does not lead to better synthesized view quality. Thus, denoising methods to improve inter-view consistency have been proposed [8, 9]. Our work is fundamentally different in that we seek an *optimal 3D surface* in a search space of rate-constrained surfaces, where the chosen surface is then projected to a number of camera viewpoints for compact representation as compressed multiview depth video. Hence, not only that by construction our input depth maps to a multiview codec are always inter-view consistent, the set of generated consistent depth maps represents not just any 3D surface, but one that is RD-optimal.

## 3. SYSTEM OVERVIEW

We first overview our system model. We assume an array of $V$ time-synchronized depth sensors capture depth images of the same dynamic 3D scene periodically from $V$ different viewpoints. The captured depth observations are corrupted by acquisition noise, modeled as multivariate Gaussian. Given observed depth data, the encoder first identifies an RD-optimal 3D surface of the scene, for a given bit budget. The chosen 3D surface is then re-projected back to the camera views, which are subsequently encoded as multiview depth videos as a representation of the chosen 3D surface, using a known multiview video coding scheme like MVC [6]. The challenge is to find the RD-optimal 3D surface for given observed depth data. We discuss the formulation of this problem next.

## 4. PROBLEM FORMULATION

We now present our formulation of the RD-optimized 3D reconstruction problem. As a convention, a matrix and a vector will be denoted respectively by boldface uppercase letter (e.g, $\mathbf{D}$) and lowercase letter (e.g., $\mathbf{d}$), and a scalar will be denoted by an italic upper or lowercase letter (e.g., $n$ or $N$).

Suppose one or more objects move in 3D space and are captured at each time instant $t$ by a set of $V$ depth cameras from different viewpoints, producing at each instant a vector of observed depth maps $\mathbf{y}^t = [\mathbf{y}_1^t, ..., \mathbf{y}_V^t]$. Let $\mathbf{s}^t$ denote the underlying (i.e., not directly observed) surface of the object at instant $t$. One can think of $\mathbf{s}^t$ as a 2D manifold in 3D, which evolves over time. Our objective is to estimate the sequence of surfaces $\mathbf{s} = \{\mathbf{s}^1, \ldots, \mathbf{s}^t\}$ from the corresponding sequence of observed depth maps $\mathbf{y} = \{\mathbf{y}^1, \ldots, \mathbf{y}^t\}$. Let $\hat{\mathbf{s}}$ denote the estimation, or *reconstruction*, of $\mathbf{s}$.

Unlike previous work on 3D reconstruction from multiview depth data [3, 4, 5], we take a *rate-distortion approach* to determine the reconstruction $\hat{\mathbf{s}}$. That is, we formulate our objective as finding the reconstruction $\hat{\mathbf{s}}$ that minimizes a distortion $\mathcal{D}(\mathbf{y}, \hat{\mathbf{s}})$ between the observations $\mathbf{y}$ and the reconstruction $\hat{\mathbf{s}}$ subject to a constraint on the number of bits $\mathcal{R}(\hat{\mathbf{s}})$ used to encode a representation of $\hat{\mathbf{s}}$. It can be shown [10] that this is equivalent to finding the $\hat{\mathbf{s}}$ that minimizes $\mathcal{D}(\mathbf{y}, \hat{\mathbf{s}}) + \lambda\mathcal{R}(\hat{\mathbf{s}})$ for some Lagrange multiplier $\lambda > 0$.

Note that we have chosen to define our distortion measure between $\mathbf{y}$ and $\hat{\mathbf{s}}$ rather than between $\mathbf{s}$ and $\hat{\mathbf{s}}$ (since we have no direct observation of $\mathbf{s}$) or between $\mathbf{y}$ and $\hat{\mathbf{y}}$ (since we wish to reconstruct the underlying surface $\mathbf{s}$ rather than the noisy observations $\mathbf{y}$). We can only hope to find a reconstruction $\hat{\mathbf{s}}$ that is somehow close to (i.e., explains) the observations $\mathbf{y}$. We call this an *observation-surface* distortion measure. In the next sections, we define the distortion term probabilistically, using the maximum likelihood (ML) formulation and an assumed noise model. We then specify the rate term using a proxy that approximates the coding rates of a typical multiview codec like MVC [6].

### 4.1. Distortion Term

We model the physics of the depth sensors and acquisition process using a conditional probability distribution $P(\mathbf{y}|\mathbf{s})$, as follows. At each instant $t$, first the underlying surface $\mathbf{s}^t$ is projected (with hidden surface removal) onto each of the $V$ views producing ideal depth maps $\mathbf{d}_1^t, \ldots, \mathbf{d}_V^t$. From these ideal depth maps, the observed depth maps $\mathbf{y}_1^t, \ldots, \mathbf{y}_V^t$ are generated probabilistically according to a zero-mean Gaussian noise with conditional probability density:

$$f(\mathbf{y}_i^t|\mathbf{d}_i^t) = \frac{1}{(2\pi)^{\frac{MN}{2}}|\mathbf{Q}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}_i^t - \mathbf{d}_i^t)^T\mathbf{Q}^{-1}(\mathbf{y}_i^t - \mathbf{d}_i^t)\right), \quad (1)$$

where $\mathbf{Q}$ is a covariance matrix for the $M \times N$ depth map from camera $i$ at instant $t$, and may depend on $i$, $t$, and even on the signal $\mathbf{d}_i^t$. It is assumed that the sensors are independent from each other and that the measurements are independent across time. This model can reasonably accommodate depth sensors based on stereo, structured light, or time-of-flight by accurately modeling the covariance matrix [11, 1]. However, for the results in this paper, for simplicity we assume that the

noise is uncorrelated and identically distributed from pixel to pixel, i.e., $\mathbf{Q} = \sigma^2\mathbf{I}$ for some variance $\sigma^2$.

We define as our observation-surface distortion measure

$$\mathcal{D}(\mathbf{y}, \hat{\mathbf{s}}) = \sum_{i=1}^{V} (\mathbf{y}_i^t - \mathbf{d}_i^t)^T \mathbf{Q}^{-1} (\mathbf{y}_i^t - \mathbf{d}_i^t), \tag{2}$$

where $\mathbf{d}_1^t, \ldots, \mathbf{d}_V^t$ are the ideal (i.e., noiseless) depth maps obtained by projecting the surface $\hat{\mathbf{s}}$ onto each of the $V$ views. Note that the surface $\mathbf{s}^*$ that minimizes (2) also maximizes the likelihood $P(\mathbf{y}|\mathbf{s}) = \prod f(\mathbf{y}_i^t|\mathbf{d}_i^t)$; hence $\mathbf{s}^*$ is termed the maximum likelihood (ML) surface.

### 4.2. Rate Term

The rate term $\mathcal{R}(\hat{\mathbf{s}})$ is the number of bits needed to signal to the decoder which surface $\hat{\mathbf{s}}$ it should reproduce. In practice, we will use a decoder based on an existing multiview codec such as MVC, combined with a post-processing step to turn its decoded depth maps into a consistent surface. This combination determines the set of all possible valid bit strings and the set of corresponding reproductions surfaces, and therefore also determines $\mathcal{R}(\hat{\mathbf{s}})$. However, for the purpose of efficiently optimizing the encoder, we approximate $\mathcal{R}(\hat{\mathbf{s}})$ based on a simple model of the codec. In this model, a set of depth maps $\mathbf{d}_1^t, \ldots, \mathbf{d}_V^t$ is encoded in blocks using either motion compensated or disparity compensated prediction for each block. As proxies for the bit rate needed to code each block, we use the following cost functionals for each coding mode.

#### *4.2.1. Motion Compensation Proxy*

If a block is predicted from a previous frame of the same view via motion compensation (MC), we write the cost $E_t$ as:

$$\begin{aligned} E_t(\mathbf{d}_i^t(\mathbf{p}), \mathbf{v}_i^t(\mathbf{p})) =& \|\mathbf{d}_i^t(\mathbf{p}) - \mathbf{d}_i^{t-1}(\mathbf{p} + \mathbf{v}_i^t(\mathbf{p}))\|^2 \\ &+ \alpha_t \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} \|\mathbf{v}_i^t(\mathbf{p}) - \mathbf{v}_i^t(\mathbf{q})\|^2 \end{aligned} \tag{3}$$

where $\mathbf{v}_i^t(\mathbf{p})$ is the motion vector (MV) for block $\mathbf{p}$ of view $i$ and instant $t$, and $\mathcal{N}_\mathbf{p}$ is a set of spatial neighboring blocks' positions causal to $\mathbf{p}$ (e.g., left, top, and top-right). In words, (3) computes two terms: i) motion prediction residual, and ii) the difference between MVs for current block $\mathbf{p}$ and causal neighboring blocks—itself a proxy for the cost of encoding MV $\mathbf{v}_i^t(\mathbf{p})$. $\alpha_t$ determines the relative importance between energy of prediction residual and cost of encoding MV $\mathbf{v}_i^t(\mathbf{p})$.

#### *4.2.2. Disparity Compensation Proxy*

If a block is predicted from a frame of a neighboring view of the same instant via disparity compensation (DC), we write the cost $E_v$, similar to $E_t$ in (3), as:

$$\begin{aligned} E_v(\mathbf{d}_i^t(\mathbf{p}), \mathbf{u}_i^t(\mathbf{p})) =& \|\mathbf{d}_i^t(\mathbf{p}) - \mathbf{d}_{i-1}^t(\mathbf{p} + \mathbf{u}_i^t(\mathbf{p}))\|^2 \\ &+ \alpha_v \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} \|\mathbf{u}_i^t(\mathbf{p}) - \mathbf{u}_i^t(\mathbf{q})\|^2 \end{aligned} \tag{4}$$

where $\mathbf{u}_i^t(\mathbf{p})$ is the disparity vector (DV) for block $\mathbf{p}$.

#### *4.2.3. Spatial Smoothness Proxy*

In either mode, a block requires more coding bits if it is not spatially smooth, as its high-frequency components are generally unpredictable and hence carry over to the block's prediction residual. The last proxy $E_s$ accounts for this:

$$E_s(\mathbf{d}_i^t(\mathbf{p})) = \|\mathbf{L}\, \mathbf{d}_i^t(\mathbf{p})\|^2 \tag{5}$$

where $\mathbf{L}$ is a Laplacian matrix with respect to the connectivity between pixels in the block.

#### *4.2.4. Combining Proxies to Rate Term*

We now combine the three defined proxies into a single rate term. For a given block, MVC selects the prediction mode (between MC and DC) with the smaller cost. In either mode, spatial high frequencies can contribute to a higher rate. Thus, we define the rate term $\mathcal{R}(\hat{\mathbf{s}})$ as follows:

$$\begin{aligned} \mathcal{R}(\hat{\mathbf{s}}) = \sum_{i=1}^{V} \sum_{\mathbf{p} \in \mathcal{B}} \big( & E_s(\mathbf{d}_i^t(\mathbf{p})) + \alpha_p \min\{E_t(\mathbf{d}_i^t(\mathbf{p}), \mathbf{v}_i^t(\mathbf{p})), \\ & E_v(\mathbf{d}_i^t(\mathbf{p}), \mathbf{u}_i^t(\mathbf{p}))\}\big) \end{aligned} \tag{6}$$

where $\alpha_p$ denotes the relative importance between spatial smoothness and prediction cost, and $\mathcal{B}$ is the set of coordinates for blocks in each $M \times N$ depth map.

### 4.3. Optimization Problem

To find the rate-distortion optimal surface $\hat{\mathbf{s}}$ for a given $\lambda$, it suffices to find for each instant $t$ a set of depth maps $\mathbf{d}_i^t$ and associated MVs and DVs $\mathbf{v}_i^t(\mathbf{p})$ and $\mathbf{u}_i^t(\mathbf{p})$ that minimize the RD cost subject to a consistency constraint:

$$\underset{\mathbf{d}_i^t, \mathbf{v}_i^t(\mathbf{p}), \mathbf{u}_i^t(\mathbf{p})}{\text{minimize}} \quad \mathcal{D}(\mathbf{y}^\mathbf{t}, \hat{\mathbf{s}}^t) + \lambda \mathcal{R}(\hat{\mathbf{s}}^t) \tag{7}$$

$$\text{s.t.} \quad E_c < \eta. \tag{8}$$

The consistency constraint $E_c < \eta$ ensures that the depth maps are projections of a single 3D surface, and is given as a constraint on the cost $E_c$ of the differences between the depth maps when they are re-projected into other views:

$$E_c = \sum_{i=1}^{V} \sum_{j \in \mathcal{N}(i)} ||\mathbf{d}_i^t - \phi_{j,i}(\mathbf{d}_j^t)||^2, \tag{9}$$

where $\phi_{j,i}(\mathbf{d})$ is a mapping function that maps pixels in $\mathbf{d}$ of view $j$ to pixels in view $i$ *if* they are not occluded by other pixels, and $\mathcal{N}(i)$ is the set of neighboring view indices of view $i$. For simplicity, in this paper we assume the cameras are sequential (left to right) and the neighboring view set is restricted to be neighboring left and right views $i-1$ and $i+1$, if they exist. The consistency constraint is most easily incorporated by adding the cost (9) to the objective function with a large multiplier $\alpha_c$, which leads to the following unconstrained problem:

$$\underset{\mathbf{d}_i^t, \mathbf{v}_i^t(\mathbf{p}), \mathbf{u}_i^t(\mathbf{p})}{\text{minimize}} \quad \mathcal{D}(\mathbf{y}^\mathbf{t}, \hat{\mathbf{s}}^t) + \lambda \mathcal{R}(\hat{\mathbf{s}}^t) + \alpha_c E_c. \tag{10}$$

## 5. OPTIMIZATION ALGORITHM

We now describe our proposed algorithm to solve our formulated optimization problem (10). Since the rate term in (10) is expressed in terms of individual blocks, we can rewrite (10) so that both the distortion and consistency terms are also expressed in terms of blocks. Then, assuming the covariance matrix $\mathbf{Q}$ is diagonal, it suffices to find for each block at location $\mathbf{p}$ the optimal depth map $\mathbf{d}_i^t(\mathbf{p})$, associated MV $\mathbf{v}_i^t(\mathbf{p})$, and DV $\mathbf{u}_i^t(\mathbf{p})$ that minimize objective (10).

Because of the non-convex mapping function $\phi_{i,k}(\cdot)$ in the consistency term (9), finding a globally optimal solution to (10) is difficult. Instead, we propose an alternating two-step optimization scheme that finds a locally optimal solution. The two steps are: i) align edges in depth maps of consecutive views to match *scene structure* across views; and ii) smooth texture within depth edges to match *scene texture* across views. An overview of this algorithm is shown below.

**Algorithm 1** Alternating Two-step Algorithm

1: Initialize $\alpha_c = 1$.
2: **repeat**
3: Step A: Match *scene structure* by edge realignment.
4: Step B: Match *scene texture* by texture smoothing.
5: Increase $\alpha_c$.
6: **until** $\alpha_c$ sufficiently large.

We describe the two steps in order.

### 5.1. Step A: Match scene structure by edge realignment

We observe that typical depth maps are piecewise smooth. Hence large inconsistencies across views usually means a pixel in a closer depth region (foreground) in one view is mapped to a pixel in a further away depth region (background) in a neighboring view (or vice versa), resulting in a large increase in the consistency term (9). Fig. 1 illustrates example blocks with foreground and background regions and distinct edges between them.

To correct for these large consistency errors, we attempt to align the boundaries of regions across views; i.e., we match the scene structure across views. Specifically, we first detect depth edges in a block using a simple thresholding method: we declare an edge between two neighboring pixels if the depth values between them is larger than a threshold $\epsilon$.

Pixels on either side of a declared edge are labeled *candidate pixels*. At each iteration, we test the reassignment of opposite depth values at each candidate pixel (from foreground to background or vice versa), and note the potential decrease in objective (10). The candidate pixel with the largest decrease in objective is chosen for depth value reassignment. The depth value reassignment induces a change in the set of detected edges and candidates pixels, so both are updated correspondingly. To make the reassignment operation robust to noise, the depth value of a candidate pixel will be reassigned only if the resulting consistency $E_c$ decreases by a significant amount. We repeat this process until there are no more depth value reassignment of candidate pixels that can induce a further decrease in objective value.

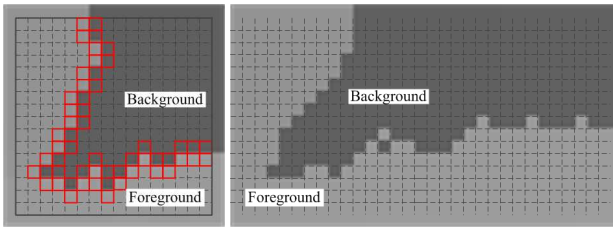


**Fig. 1**: Left: block in current view. Right: corresponding blocks in reference view.

Note that the above edge realignment procedure is performed on a *target* depth map in a single view given a *reference* depth map of a neighboring view is fixed and used for computation of (9). To ensure the role of target and reference can be reversed for each pair of neighboring views, we perform both a forward and a backward pass through the views. The complete algorithm is shown below.

**Algorithm 2** Step A: edge realignment

**for** $i = 2$ to $V$ or $i = V-1$ to $1$ **do**
  Detect edges and identify candidate pixels.
  **repeat**
    **repeat**
      Test opposite depth assignment on candidates.
      Pick winner, update edge and candidate pixels.
    **until** No objective-decreasing candidates.
  **until** All blocks are processed.
**end for**

### 5.2. Step B: Match scene texture by texture smoothing

Given depth edges (structure) of neighboring views are now aligned, we now match the interior regions (texture) of neighboring views. For a single block at position $\mathbf{p}$, assuming $E_t < E_v$ for the rate term (6), we can take the partial derivative of objective (10) to get:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{d}_i^t(\mathbf{p})} &= 2(\mathbf{Q_b}^{-1})(\mathbf{d}_i^t(\mathbf{p}) - \mathbf{y}_i^t(\mathbf{p})) \\ &+ 2\alpha_c \sum_{j\in\mathcal{N}(i)} \left(\mathbf{d}_i^t(\mathbf{p}) - \phi_{j,i}(\mathbf{d}_j^t)(\mathbf{p})\right) + 2\lambda(\mathbf{L}\mathbf{L}^{\mathbf{T}})\mathbf{d}_i^t(\mathbf{p}) \\ &+ 2\lambda\alpha_p \begin{cases} (\mathbf{d}_i^t(\mathbf{p}) - \mathbf{d}_i^{t-1}(\mathbf{p}+\mathbf{v}_i^t(\mathbf{p})) & \text{motion mode} \\ (\mathbf{d}_i^t(\mathbf{p}) - \mathbf{d}_{i-1}^t(\mathbf{p}+\mathbf{u}_i^t(\mathbf{p})) & \text{disparity mode} \end{cases}\end{aligned} \tag{11}$$

where $\mathbf{Q}_b$ is the covariance matrix for a given block. Assuming the mode decision and MV $\mathbf{v}_i^t(\mathbf{p})$ and DV $\mathbf{u}_i^t(\mathbf{p})$ are fixed, we can set (11) to zero and solve for $\mathbf{d}_i^t(\mathbf{p})$ in closed form.

Because the optimal MV $\mathbf{v}_i^t(\mathbf{p})$ and DV $\mathbf{u}_i^t(\mathbf{p})$ depend on $\mathbf{d}_i^t(\mathbf{p})$, they are interdependent. To resolve the interdependency, we alternately optimize either depth block $\mathbf{d}_i^t(\mathbf{p})$

or vectors $\mathbf{v}_i^t(\mathbf{p})$ and $\mathbf{u}_i^t(\mathbf{p})$ at a time, until convergence. The algorithm is summarized below.

**Algorithm 3** Step B: surface smoothing

**for** $i = 1$ to $V-1$ or $i = V$ to 2 **do**
  **repeat**
    **repeat**
      Given $\mathbf{d}_i^t(\mathbf{p})$, find optimal $\mathbf{v}_i^t(\mathbf{p})$ and $\mathbf{u}_i^t(\mathbf{p})$ .
      Given $\mathbf{v}_i^t(\mathbf{p})$ and $\mathbf{u}_i^t(\mathbf{p})$, find optimal $\mathbf{d}_i^t(\mathbf{p})$.
    **until** MV and DV converge.
  **until** All blocks $\mathbf{p}$ are processed.
**end for**

## 6. EXPERIMENTATION

To test the performance of our proposed algorithm, we use texture and depth maps from two $1024 \times 768$ MPEG FTV multiview test sequences, `Lovebird1` and `Balloons`, at camera captured views $4, 6, 8$ and $1, 3, 5$, respectively.

The test sequences are pre-processed with one of three methods before being compressed with the MVC codec [6]. In the first method, `Unprocessed`, the raw acquired depth maps $\mathbf{y}$ are fed into the MVC codec. This is the conventional method. In the second method, `RD-optimized`, our algorithm is used to produce a surface $\hat{\mathbf{s}}_\lambda$ that minimizes $\mathcal{D}(\mathbf{y}, \hat{\mathbf{s}}) + \lambda \mathcal{R}(\hat{\mathbf{s}})$ for a selected value of $\lambda$. The surface $\hat{\mathbf{s}}_\lambda$ is projected onto the $V$ views, and the resulting depth maps are fed into the MVC codec. In the third method, `ML-solution`, our algorithm is used with $\lambda = 0$ to produce the ML surface $\mathbf{s}_0$. The surface $\hat{\mathbf{s}}_0$ is projected onto the $V$ views and the resulting depth maps are fed into the codec.

After encoding and decoding the set of depth maps using the MVC codec, the decoded depth maps may no longer be consistent. To ensure inter-view consistency at the decoder, we apply an averaging procedure, similar to one in [9], which projects all views to the center view, averages the projected depth values for each pixel, and then re-projects the center view back to the other views. These re-projected depth maps represent the decoded surface.

We evaluate the quality of the decoded surface using two metrics. The first metric is our observation-surface distortion measure (2) relative to the minimum possible value of the distortion for the given observations $\mathbf{y}$,

$$\mathcal{D}'(\mathbf{y}, \hat{\mathbf{s}}) = \mathcal{D}(\mathbf{y}, \hat{\mathbf{s}}) - \mathcal{D}(\mathbf{y}, \hat{\mathbf{s}}_0), \qquad (12)$$

where $\hat{\mathbf{s}}_0$ is the ML surface, which minimizes $\mathcal{D}(\mathbf{y}, \hat{\mathbf{s}})$. The second metric is the PSNR of a virtual view (between neighboring camera views) synthesized from decoded texture and depth maps, via depth-image-based-rendering (DIBR) [2]. For this metric, the ground truth is taken to be the virtual view synthesized from the uncompressed texture maps and uncompressed depth maps obtained from the ML surface.

Figure 2 show distortion-rate curves for `Unprocessed` and `RD-optimized` for the two sequences `Lovebird1` and `Balloons`, for the relative observation-surface distortion measure (12). The RD curve for `Unprocessed` was generated by varying the MVC quantization parameters (QPs), while the RD curve for `RD-optimized` is the lower convex hull of all RD pairs generated by varying both the QPs and $\lambda \in \{0, 0.01, 0.1, 1, 10, 100\}$. One can see that `RD-optimized` outperforms `Unprocessed` by a significant amount, demonstrating that pre-processing of the acquired noise-corrupted depth maps $\mathbf{y}$ is essential in improving RD performance.

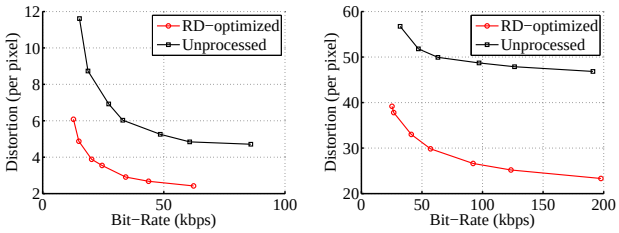


**Fig. 2**: Distortion-rate curve of the RD-optimized surface and the unprocessed surface comparing against ML surface for `Lovebird1` (left) and `Balloons` (right).

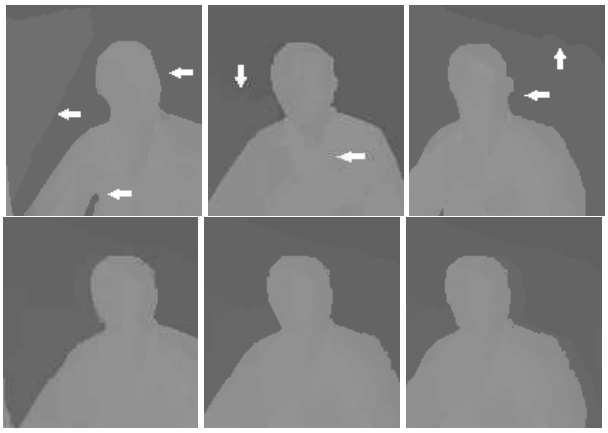

**Fig. 3**: Comparing unprocessed with ML depth maps for `Lovebird1` (for clarity in visual presentation, all pixel values are increased by 80). From top to bottom: unprocessed depth maps at views 4,6,8; ML solution at views 4, 6, 8.

**Fig. 4**: RD-optimized depth map with $\lambda = 1, 100$, respectively. Left two: `Lovebird1` at view 8. Right two: `Balloons` at view 5.

Figures 3 shows unprocessed and ML depth maps from different views for `Lovebird1`. Compared to the unprocessed depth maps, the ML depth maps show a visually significant improvement in inter-view consistency.

Figure 4 shows RD-optimized depth maps in the center

view for $\lambda = 0.01, 1, 100$. As $\lambda$ increases, the rate term becomes a heavier penalty, resulting in a larger distortion. We observe that the depth map for $\lambda = 100$ has smeared edges between foreground and background; without sharp edges, a 3D surface becomes easier to code.

We now compare the performance of the different methods using the second quality metric—PSNR of synthesized virtual views. We use widely adopted DIBR software, VSRS version 3.5, to generate the virtual views. Fig. 5 shows the PSNR-rate curve. Using the Bjontegaard metric to compute PSNR gain, for `Lovebird1`, `RD-optimized` has average gains of $1.74$dB and $2.42$dB over `ML-solution` and `Unprocessed`, respectively. For `Balloons`, `RD-optimized` has average gains of $0.87$dB and $1.28$dB over `ML-solution` and `Unprocessed`, respectively. As expected, as rate increases, 3D surfaces computed by `RD-optimized` approach those computed by `ML-solution`, thus achieving the same RD performance.

Cropped images at virtual views of the two sequences are shown in Fig. 6. Improvements indicated by arrows can be clearly observed.

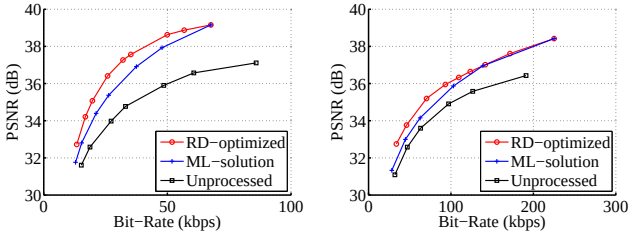


**Fig. 5**: PSNR of synthesized virtual views at decoder versus coding rate for `Lovebird1` (left) and `Balloons` (right).

**Fig. 6**: Top Row (`Lovebird1`): synthesized view 5 using texture & depth maps at view 4, 6. Depth maps are of 48kbps: `Unprocessed` (left), `ML-solution` (center), `RD-optimized` (right). Bottom Row (`Balloons`): synthesized view 2 using texture & depth maps at view 1, 3. Depth maps are of 100kbps.

## 7. CONCLUSION

Given noise-corrupted depth observations from multiple viewpoints, in this paper we propose to derive an RD-optimal 3D surface of a dynamic scene subject to a representation size constraint. Unlike previous work that finds the most likely 3D surface given noisy observations regardless of representation size, our identified 3D surface optimally trades off the posterior probability with representation size. We propose an iterative algorithm that alternately optimizes the scene structure (depth edges) and the scene texture (depth texture) until convergence. Experimental results show that using projections of our RD-optimized 3D reconstruction to multiple depth maps for multiview depth video coding can outperform unprocessed depth maps by up to $2.42$dB in PSNR of synthesized virtual views at the decoder for the same bitrate.

## 8. REFERENCES


[1] S. B. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor system description, issues and solutions," in *CVPR Workshop*, Washington, DC, June 2004.

[2] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *App. of Digital Image Processing XXXII, Proceedings of the SPIE*, 2009, vol. 7443 (2009), pp. 74430T–74430T–11.

[3] Z. Zhang and O. Faugeras, "A 3D world model builder with a mobile robot," in *International Journal of Robotics Research*, August 1992, vol. 11, no.4, pp. 269–285.

[4] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *ACM SIGGRAPH*, New Orleans, LA, August 1996.

[5] R. Newcombe et al., "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE Int. Symp. on Mixed and Augmented Reality*, Basel, Switzerland, October 2011.

[6] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," in *IEEE Trans. on CSVT*, Nov. 2007, vol. 17, no.11, pp. 1461–1473.

[7] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Trans. on CSVT*, July 2003, vol. 13, no.7, pp. 560–576.

[8] H. Helgason, H. Li, and M. Flierl, "Multiscale framework for adaptive and robust enhancement of depth in multi-view imagery," in *IEEE ICIP*, Orlando, FL, October 2012.

[9] R. Li, D. Rusanovskyy, M. Hannuksela, and H. Li, "Joint view filtering for multiview depth map sequences," in *IEEE ICIP*, Orlando, FL, October 2012.

[10] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," in *IEEE Trans. on ASSP*, September 1988, vol. 36, no.9, pp. 1445–1453.

[11] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang, "3D deformable face tracking with commodity depth cameras," in *ECCV*, Crete, Greece, September 2010.