IMAGE INTERPOLATION FOR DIBR VIEW SYNTHESIS USING GRAPH FOURIER TRANSFORM

Yu Mao [*], *Gene Cheung* [#] *and Yusheng Ji* [#]

[*] The Graduate University for Advanced Studies, [#] National Institute of Informatics

ABSTRACT

Given texture and depth maps of one or more reference viewpoint(s), depth-image-based rendering (DIBR) can synthesize a novel viewpoint image by mapping texture pixels from reference to virtual view using geometric information provided by corresponding depth pixels. If the virtual view camera is located closer to the 3D scene than the reference view camera, objects close to the camera will increase in size in the virtual view, and DIBR's simple pixel-to-pixel mapping will result in expansion holes that require proper filling. Leveraging on recent advances in graph signal processing (GSP), in this paper we propose to select appropriate graph Fourier transforms (GFT)—adaptive to unique signal structures of the local pixel patches—for expansion hole filling. Our algorithm consists of two steps. First, using structure tensor we compute an adaptive kernel centered at a target empty pixel to identify suitable neighboring pixels for construction of a sparse graph. Second, given the constructed graph with carefully tuned edge weights, to complete the target pixel we formulate an iterative quadratic programming problem (with a closed form solution in each iteration) using a smoothness prior in the GFT domain. Experimental results show that our algorithm can outperform inpainting procedure employed in VSRS 3.5 by up to 4.57dB.

***Index Terms* —** depth-image-based rendering, image interpolation, graph Fourier transform

1. INTRODUCTION

Free viewpoint video [1] provides users the freedom to choose any vantage point from which to reconstruct a viewpoint image for observation of a 3D scene. To enable free viewpoint, texture maps (conventional color images) and depth maps (per-pixel distance between objects in the 3D scene and the capturing camera) from multiple camera viewpoints are captured and encoded at the sender—a format called *texture-plus-depth*. At the receiver, a new virtual viewpoint image can be synthesized using *depth-image-based rendering* (DIBR) techniques such as 3D warping [2]. In a nutshell, DIBR maps each texture pixel in a reference view to a pixel location in the virtual view, using geometric information provided by the corresponding depth pixel. Due to occlusion in the reference view (spatial areas in virtual view that are occluded by foreground objects in the reference view), missing pixels in the virtual view (called *disocclusion holes*) are subsequently filled in using inpainting algorithms [3]. For small camera movement from reference to virtual view along x- or y-dimension (camera moving left-right or top-down), this DIBR synthesis plus inpainting approach has been shown to work reasonably well [1], and is the conventional approach in the free view synthesis literature.

In immersive applications such as teleconferencing, a viewer in a sitting position observes rendered images on a 2D display, where the image viewpoints are adjusted according to the tracked head locations of the viewer [4]. The resulting *motion parallax* effect can enhance the viewer's depth perception in the 3D scene. Besides x-dimensional head movement (moving one's head left-right), z-dimensional head movement (moving one's head front-back) is also natural for a sitting observer. However, few works in the literature have formally addressed the problem of synthesizing viewpoint images corresponding to large z-dimensional camera movements. We address this problem in our paper.

(a) captured view (b) DIBR-synthesized view

Figure 1. Examples of disocclusion and expansion holes: a) camera captured texture map; b) *diocclusion holes* are larger contiguous empty regions next to foreground object boundaries, and *expansion holes* are smaller empty regions on the surfaces of foreground objects.

When the virtual camera is located closer to the 3D scene than the reference view camera, objects close to the camera will increase in size in the virtual view. This means that the aforementioned pixel-to-pixel mapping during DIBR from reference to virtual view is not sufficient to complete entire surfaces of rendered objects, resulting in *expansion holes* [5]. Note that expansion holes differ from disocclusion holes in that the objects are visible in the reference view(s), but *insufficient pixel samples* in reference view(s) results in holes in the virtual view. See Fig. 1 for an illustration of expansion and disocclusion holes.

In this paper, leveraging on recent advances in *graph signal processing* (GSP) [6], we propose to select appropriate *graph Fourier transforms* (GFT) for expansion hole filling in the virtual view image. Like fixed transforms such as Discrete Cosine Transform (DCT), projecting a signal onto GFT is a simple linear operation, yet unlike DCT, the definition of GFT can adapt to the unique signal structure of each local patch for signal-adaptive processing. Our algorithm consists of two steps. First, using structure tensor we compute an adaptive kernel centered at a target empty pixel to identify suitable neighboring pixels for construction of a sparse graph. Second, given the constructed graph with carefully tuned edge weights, to complete the target pixel we formulate an iterative quadratic programming problem (with a closed form solution in each iteration) using a smoothness prior in the GFT domain. Experimental results show that our algorithm can outperform inpainting procedure employed in VSRS 3.5 by up to 4.57dB.

The outline of the paper is as follows. We first discuss related work in Section 2 and overview our free view synthesis system in Section 3. We then discuss the construction of an appropriate graph centered at a target pixel in Section 4. Formulation of an

978-1-4799-4758-4/14/$31.00 ©2014 IEEE

iterative quadratic programming problem for image completion is discussed in Section 5. Finally, experimentation and conclusions are presented in Section 6 and 7, respectively.

2. RELATED WORK

Increase in object size due to large z-dimensional virtual camera motion is analogous to increasing the resolution (*super-resolution* (SR)) of the whole image. However, during z-dimensional camera motion an object closer to the camera increases in size faster than objects farther away, while in SR resolution is increased uniformly for all spatial regions in the image. Nonetheless, SR techniques [7] can potentially be employed for expansion hole filling. However, SR techniques typically operate on regular 2D pixel grid, while in the DIBR scenario the available pixels mapped from the reference view(s) are initially not on the grid (before rounding to nearest grid positions for rendering), and hence a more general graph formulation is more natural. Further, recent non-local SR techniques such as [7] tend to be computationally expensive, while our interpolation scheme essentially performs only local filtering, and thus is significantly more computation-efficient.

Instead of transmitting texture / depth image pairs of different captured viewpoints to the decoder for DIBR-based virtual view synthesis plus interpolation, an alternative is to represent captured texture / depth pixels as a triangular mesh at the encoder [8]. At the decoder, each pixel on the 2D grid in the virtual image is then linearly interpolated using nodes that define the enclosing triangle. In this paper, though we assume the popular texture-plus-depth image representation of a 3D scene, the focus is on the image interpolation aspect of view synthesis. Thus in the experiments we compare our proposal to a linear interpolation scheme that is representative of the performance of a mesh-based representation.

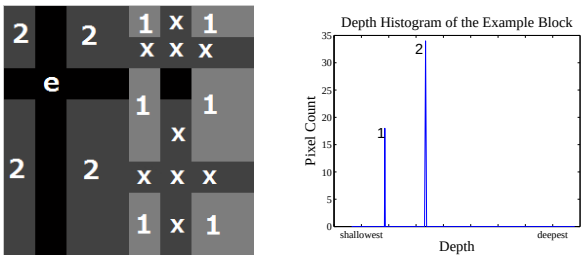
GSP is the study of signals that live on structured data kernels described by graphs [6]. In particular, GFT has been successfully used for depth map compression [9], denoising [10], etc. In this paper, we propose to use GFT for expansion hole filling, or more generally, image interpolation. Compared to our previous work on the topic [5], we introduce three improvements: i) an adaptive kernel based on structure tensor has been deployed to select suitable neighboring pixels around a target pixel for graph construction; ii) a parameter h in the graph-signal smoothness prior $\mathbf{x}^T\mathbf{L}^h\mathbf{x}$ is adjusted according to the shape of the adaptive kernel, so that the amount of smoothness applied can be adapted based on local signal characteristics; and iii) we formulate an iterative unconstrained quadratic program, where each iteration can be solved in closed form efficiently. Our optimization method is an order of magnitude faster than [5] that formulates a linear program, and thus is conceivably implementable in real-time.

3. DIBR SYSTEM OVERVIEW

We first overview our interactive free viewpoint streaming system. A sender transmits a single texture / depth map pair of one camera captured view (*reference view*), so that a receiver can synthesize images of virtual views near the reference view via DIBR. If the client desires to render images of virtual viewpoint farther away from the reference view, a new texture / depth map pair of captured view nearer the desired virtual viewpoint is transmitted. In this paper, we focus only on synthesis of virtual view images near the reference view but with large z-dimensional camera movements.

3.1. Depth Layering for Image Interpolation

As discussed, after DIBR there exists disocclusion and expansion holes in the synthesized image that require filling. We define an expansion hole as follows: a spatial area of an object's surface in the virtual view, whose corresponding area in the reference view is



(a) depth block (b) depth histogram

Figure 2. Examples of depth layers and corresponding histogram: a) pixels in a depth block are classified into depth layers and empty pixels; b) corresponding histogram of depth values for the block.

visible but smaller in size. Unlike disocclusion holes, expansion holes can leverage on information of neighboring pixels of the same object for interpolation.

To identify pixels from the same physical object that are useful for interpolation, we adopt a *depth layering* approach. Specifically, for a given pixel block in the synthesized view, we first construct a histogram containing depth values of pixels in the block. Fig. 2(b) shows an example. Peaks in the histogram are labeled as layers ordered from shallow depth to deep depth. Fig. 2(a) shows the depth pixels in the block with assigned layer numbers.

Interpolation is performed layer-by-layer, starting from the shallowest, so that when interpolation for layer i is performed, each pixel in layer $j > i$ that is inside a convex set spanned by pixels in layer i is treated as an empty pixel. In Fig. 2(a), layer 2 pixels that are marked 'X' are treated as empty pixels during expansion hole filling of layer 1. This is important because when there is insufficient pixel sampling in the reference view, during DIBR background pixels can land in empty pixel locations of foreground objects in the virtual view, resulting in an incoherent mixture of foreground and background pixels. We focus our discussion on empty pixel interpolation for a given layer i next.

4. ADAPTIVE KERNEL TO CONSTRUCT GRAPH

We now discuss how to choose a subset of pixels in the same depth layer around a target empty pixel $\mathbf{p}$ to construct a graph; the constructed graph will be subsequently used for graph-based pixel interpolation. The reason for adaptively choosing only a subset of pixels of the same layer is because the same physical object can have distinct textural patterns that influence how pixels should be interpolated. For example, a red and blue striped shirt implies that an empty pixel inside a blue stripe should be interpolated using only neighboring blue pixels. To detect present textural patterns, we use *adaptive kernel* introduced in [11].

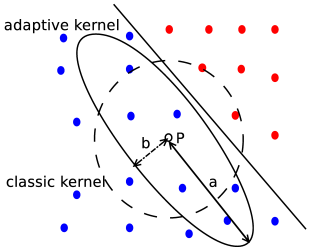


Figure 3. Illustration of adaptive kernel on a pixel patch with stripes of blue and red pixels: an ellipse is elongated along a direction perpendicular to the principal gradient, so that only similar pixels are selected for pixel interpolation.

There are two intuitive steps in adaptive kernel. First, the *principal gradient* in a local patch is derived via computation of the *structure tensor*. The structure tensor $S_w[\mathbf{p}]$ defined on a pixel location $\mathbf{p}$ can be computed as:

$$\begin{bmatrix} \sum_{\mathbf{r}} w[\mathbf{r}](\Delta_x[\mathbf{p}-\mathbf{r}])^2 & \sum_{\mathbf{r}} w[\mathbf{r}]\Delta_x[\mathbf{p}-\mathbf{r}]]\Delta_y[\mathbf{p}-\mathbf{r}] \\ \sum_{\mathbf{r}} w[\mathbf{r}]\Delta_x[\mathbf{p}-\mathbf{r}]]\Delta_y[\mathbf{p}-\mathbf{r}] & \sum_{\mathbf{r}} w[\mathbf{r}](\Delta_y[\mathbf{p}-\mathbf{r}])^2 \end{bmatrix} \quad (1)$$

where $\mathbf{r}$ defines a neighborhood around pixel $\mathbf{p}$, $\Delta_x(\mathbf{p})$ and $\Delta_y(\mathbf{p})$ are the texture image gradients[1] along the x- and y-axis at pixel $\mathbf{p}$ respectively, and $w[\mathbf{r}]$ is a weight assigned to neighbor $\mathbf{r}$. Weights are chosen so that $\sum_{\mathbf{r}} w[\mathbf{r}] = 1$. Having computed $S_w[\mathbf{p}]$, one can perform eigen-decomposition on the matrix, and the eigenvector v_2 that corresponds to the larger eigenvalue λ_2 is the principal gradient of the patch. See Fig. 3 for an illustration.

In the second step, an adaptive kernel ellipse centered at the target pixel $\mathbf{p}$ is defined to identify pixels of the same depth layer for graph construction. The ellipse has major and minor axes aligned with the tensor eigenvectors v_1 and v_2. The idea is to construct an ellipse elongated along a direction perpendicular to the principal gradient of the patch. In particular, let a and b be the major and minor radius, *i.e.* in the eigenvector coordinate system (x', y'),

$$\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 = 1 \quad (2)$$

We compute a and b as:

$$a = \delta\lambda_2, \quad b = \delta\lambda_1 \quad (3)$$

for parameter δ. In other words, if the principal gradient is large (large λ_2), then the ellipse is more elongated along a direction perpendicular to the gradient.

Fig. 3 shows an example ellipse elongated to contain only blue neighboring pixels. In contrast, a classic kernel will be a circle with a fixed radius, containing blue and red pixels.

4.1. Graph Construction

Having identified a subset of pixels in the same depth layer suitable for interpolation of the empty pixel, we construct a graph $\mathcal{G}$ as follows. Each pixel in the kernel ellipse is represented as a node in the graph $\mathcal{G}$. We draw an edge between pixels (nodes) $\mathbf{p}$ and $\mathbf{q}$ if their geometric distance $\|\mathbf{p}-\mathbf{q}\|_2$ is smaller than a threshold ϵ. The edge weight $w_{\mathbf{p},\mathbf{q}}$ between the two pixels is computed as:

$$e_{\mathbf{p},\mathbf{q}} = \exp\left\{-\frac{\|I(\mathbf{p}) - I(\mathbf{q})\|_2^2}{\sigma^2}\right\} \quad (4)$$

where $I(\mathbf{p})$ is the intensity for pixel $\mathbf{p}$, and σ is a chosen parameter. For empty pixels in the kernel without intensity values, the average of neighboring pixel intensities can be used for initialization; given the derived adaptive kernel, neighboring pixels are likely similar, so this initialization is reasonable.

5. GRAPH-BASED PIXEL INTERPOLATION

Given a constructed graph $\mathcal{G}$, we now discuss how we perform graph-based image interpolation. We first define the following terms. *Adjacency matrix* $\mathbf{A}$ has entry $A_{i,j}$ containing edge weight $e_{i,j}$ if an edge connecting nodes i and j exists, and 0 otherwise. *Degree matrix* $\mathbf{D}$ is a diagonal matrix with non-zero entries $D_{i,i} = \sum_j e_{i,j}$. A *graph Laplacian* $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. $\mathbf{L}$ is used in our definition of objective function, as discussed next.

[1]Gradient $\Delta(\mathbf{p})$ at pixel $\mathbf{p}$ is computed as the difference in intensity from a nearest neighbor $\mathbf{q}$ divided by the distance between $\mathbf{p}$ and $\mathbf{q}$.

5.1. Quadratic Programming Formulation

Let the total number of pixels (synthesized and empty pixels) in the kernel ellipse be N. Without loss of generality, let the K synthesized pixels in the kernel be $s_1, \ldots, s_K$, and the to-be-interpolated length-N signal, $N > K$, be $\mathbf{x}$. Let $\mathbf{u}_i$'s be a set of K length-N unit vectors, $[0, \ldots, 0, 1, 0, \ldots, 0]$, where the single non-zero entry is at position i. Our objective is to minimize a weighted sum of: i) the l_2-norm of the difference between interpolated signal $\mathbf{x}$ and K synthesized pixels s_i, and ii) smoothness prior $\mathbf{x}^T\mathbf{L}^h\mathbf{x}$:

$$\min_{\mathbf{x}} \|\sum_{i=1}^{K} \mathbf{u}_i^T\mathbf{x} - s_i\|_2^2 \; + \; \mu\,\mathbf{x}^T\mathbf{L}^h\mathbf{x} \quad (5)$$

where μ is a weighting parameter that balances the distortion and smoothness terms.

Given $\mathbf{u}_i$, s_i, μ and $\mathbf{L}$, (5) is an unconstrained quadratic programming problem in $\mathbf{x}$ and can be solved efficiently in closed form [12]. After solving for $\mathbf{x}$, the edge weights $e_{i,j}$ and consequently Laplacian $\mathbf{L}$ can be updated using (4), and then (5) is solved again. This iterative procedure continues until limited computation resource is exhausted, or solution $\mathbf{x}^*$ converges.

5.2. Graph Fourier Transform Interpretation

The optimization (5) can be alternatively interpreted as follows. Let $\mathbf{\Phi}$ be the eigen-matrix (eigenvectors arranged as rows in matrix) of the Laplacian $\mathbf{L}$. $\mathbf{\Phi}$ is known as the *graph Fourier transform* (GFT) for defined graph $\mathcal{G}$. (5) can now be rewritten as [6]:

$$\min_{\alpha} \|\sum_{i=1}^{K} \mathbf{u}_i^T\mathbf{\Phi}^{-1}\alpha - s_i\|_2^2 \; + \; \mu \sum_i \lambda_i^h \alpha_i^2 \quad (6)$$

where α are the GFT coefficients given signal $\mathbf{x}$, *i.e.* $\alpha = \mathbf{\Phi}\mathbf{x}$, and λ_i is the ith eigenvalue of Laplacian $\mathbf{L}$, or equivalently, the ith graph frequency of GFT $\mathbf{\Phi}$. In words, instead of solving for the signal $\mathbf{x}$ directly in (5), we can equivalently solve for the GFT domain representation of $\mathbf{x}$, *i.e.* coefficients α. From (6), one can see that the smoothness term $\mathbf{x}^T\mathbf{L}^h\mathbf{x}$ is rewritten as a sum of squared coefficients α_i^2 each weighted by the graph frequency λ_i raised to the power h. Hence, an optimal solution α^* with a small objective function value cannot have large high-frequency coefficients—an optimal solution must be smooth.

The amount of smoothness applied for the optimization can be enforced *globally* via parameter μ and *locally* via parameter h. h means a signal should be smooth with respect to its h-hop neighbors. In this paper, we select h to be proportional to the major radius a of the adaptive kernel ellipse. The rationale is that an elongated ellipse means more pixels geometrically farther from the target empty pixel is included in the kernel, and our h selection allows to smooth over more pixels from the same side.

6. EXPERIMENTATION

We used Middlebury datasets `art` and `laundry`[2] as our multi-view image test sequences. We used the same methodology in [5] to first generate a reference view v_r with texture and depth maps of lower resolution than captured images. Using texture and depth maps of v_r, we used DIBR to generate virtual view v_0.

Four different methods were used to construct v_0. In the first method called `VSRS+`, we modified VSRS software version 3.5 to use a single reference view, and then called the default inpainting scheme in VSRS to fill in all holes. For the other three methods, we first identified expansion holes and then used different methods to interpolate the holes. `linear` and `GFT` are the linear and graph-based interpolation methods in [5]. `AGFT` is our proposed scheme in this paper.

[2]http://vision.middlebury.edu/stereo/data/scenes2006/

Table 1. PSNR Comparison

	VSRS+	Linear	GFT	AGFT
art	19.11	22.87	23.36	23.69
laundry	19.17	21.94	22.53	23.04

Table 2. SSIM Comparison

	VSRS+	Linear	GFT	AGFT
art	0.9650	0.9771	0.9792	0.9810
laundry	0.9651	0.9743	0.9768	0.9784

6.1. Experimental Results

We computed the PSNR of the virtual view images interpolated using the four methods against the ground truth v_0. Since our proposal addresses filling of expansion holes only, we only calculated PSNR for identified expansion hole areas. The PSNR comparison is shown in Table 1. For the art sequence, we see that Linear, GFT and AGFT outperformed VSRS+ significantly: by 3.76dB, 4.25dB and 4.57dB respectively. This demonstrates that the correct identification of expansion holes and subsequent interpolation are important for DIBR image synthesis of virtual view with significant z-dimensional camera movement. Further, we see that AGFT outperformed GFT and linear by 0.49dB and 0.81dB, showing that by selecting kernel and smoothness prior adaptively, we can achieve better image quality.

For the laundry sequence, we observe similar trend. In this case, we see that Linear, GFT and AGFT outperformed VSRS+ by 2.77dB, 3.36dB and 3.85dB, respectively.

The SSIM comparison is also given in Table 2, which is a further comfirmation of the trend we observed by PSNR.

(a) expansion holes (b) VSRS+ (c) AGFT

Figure 4. Visual comparison between VSRS+ and AGFT for art.

Next, we examine the constructed image quality visually. In Fig. 4, we show an example region of the synthesized image before expansion hole filling, and after filling using VSRS+ and AGFT. First, we see visually in Fig. 4(a) that the presence of expansion holes is a significant problem. Second, we see in Fig. 4(b) that applying inpainting algorithm naïvely to fill in all missing pixels indiscriminately do not lead to acceptable quality for expansion hole areas. Finally, we see in Fig. 4(c) that using AGFT, expansion holes can be filled in a visually pleasing manner. Similar results for the laundry sequence are shown in Fig. 5.

7. CONCLUSION

In free viewpoint video, when the viewer's chosen virtual view for image rendering involves large z-dimensional camera motion, objects close to the camera will increase in size, and simple pixel-to-pixel mapping in DIBR from reference to virtual view will result in expansion holes that require filling. In this paper, we propose a two-step procedure for expansion hole filling leveraging on recent advances in graph signal processing. Specifically, for each target pixel, we: i) construct an adaptive kernel using structure tensor to identify suitable neighboring pixels to create a graph, and ii) complete target pixel via an iterative quadratic programming formulation with a smoothness prior in the graph transform domain. Experimental results show up to 4.57dB gain in PSNR over inpainting method employed in VSRS 3.5.

(a) expansion holes (b) VSRS+ (c) AGFT

Figure 5. Expansion holes and visual comparison between VSRS+ and AGFT for sequence laundry.

8. REFERENCES

[1] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," in *IEEE Signal Processing Magazine*, January 2011, vol. 28, no.1.

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

[3] S. Reel, G. Cheung, P. Wong, and L. Dooley, "Joint texture-depth pixel inpainting of disocclusion holes in virtual view synthesis," in *APSIPA ASC*, Kaohsiung, Taiwan, October 2013.

[4] C. Zhang, Z. Yin, and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing," in *IEEE International Workshop on Multimedia Signal Processing*, Rio de Jeneiro, Brazil, October 2009.

[5] Y. Mao, G. Cheung, A. Ortega, and Y. Ji, "Expansion hole filling in depth-image-based rendering using graph-based interpolation," in *IEEE International Conference on Acousitics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," in *IEEE Signal Processing Magazine*, May 2013, vol. 30, no.3, pp. 83–98.

[7] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," in *IEEE Transactions on Image Processing*, August 2012, vol. 21, no.8, pp. 3467–3478.

[8] D. Farin, R. Peerlings, and P. H. N. de With, "Depth-image representation employing meshes for intermediate-view rendering and coding," in *3DTV-Con*, Kos Island, Greece, May 2007.

[9] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[10] W. Hu, X. Li, G. Cheung, and O. Au, "Depth map denoising using graph-based transform and group sparsity," in *IEEE International Workshop on Multimedia Signal Processing*, Pula, Italy, October 2013.

[11] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," in *IEEE Trans. on Image Processing*, Feb. 2007, vol. 16, no.3, pp. 349–366.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, 2004.