Seventh Irish Workshop on Computer Graphics (2006)
Eurographics Irish Chapter

# Visual tuning of an image-based rendering algorithm

Yann Morvan and Carol O'Sullivan

Interaction, Simulation and Graphics lab, Computer Science Department, Trinity College Dublin

**Abstract**

*A wide variety of image-based rendering techniques exist but they all present authoring challenges. One such challenge is the setting of appropriate user parameters on a per scene basis. We demonstrate a simple approach to tune such parameters on the unstructured lumigraph rendering algorithm. It is based on a visual comparison of the input views with renderings from the same viewpoint. We further use a reasonable assumption on the unstructured lumigraph's geometric proxy to fine tune the algorithm.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display Algorithms

## 1. Introduction

A wide range of industries have come to make heavy use of computer assisted visual content creation. The human resources invested in related tasks, such as modeling, denoising of captured data, texturing or shader programming, have become considerable. The automating, or even bypassing, of some of these tasks is a significant research area in computer graphics. Image-based techiques have shown great potential in this regard. They allow for the rendering of novel views from sets of available photographs, possibly with the help of coarse geometry, whose recovery is also aided thanks to photogrametry.

Research in Image-Based Rendering (IBR) has mainly focused on achieving interactive frame rates at satisfactory visual quality while exploring the tradeoff between how finely the light field has to be sampled and how much geometric information to use [CW93, MB95, LH96, GGSC96, DBY98, SH99, WAA*00]. From an artist's point of view, all the proposed methods are not trivial to leverage. Be it because of the mechanical devices sometimes needed to acquire numerous regularly spaced views, or the difficulties of recovering registered 3D models.

One authoring drawback that all techniques share is their reliance on the proper setting of user parameters that vary depending on the scene being represented. Some techniques introduce user parameters to control the sampling approximations that they make. Examples of this would be the number of views considered for blending in view-dependent texture mapping [DBY98], or the Jpeg quality level that is used to compress each view. Others take an exact approach, but their huge memory footprint indirectly introduces parameters when it comes to controlling the lossy compression schemes that they require. For instance, Discrete Cosine Transforms require to chose a frequency above which coefficient are ignored, while Vector Quantization expects a bound on the dimensionality of the subspace to use to represent the signal. Both are commonly used for the compression of image-based datasets, the latter for light fields [LH96] and the former for surface light fields [MEG00].

In this work we propose a simple framework to help tune user parameters by maximizing a visual quality score over parameter space. We demonstrate our idea on a modified version of Buehler *et al*'s unstructured lumigraph rendering algorithm, chosen because of the authoring flexibility it provides.

We start by describing the general principle behind our framework. We then provide some background on the original unstructured lumigraph algorithm. In a third section, a discussion of the algorithm is put forward, resulting in modifications. We then describe how we applied our framework to the modified algorithm, in particular to tune parameters with a finer granularity than initially possible.

## 2. Basic principle

Image-based rendering techniques present the advantage that a gold standard of the quality of rendering that is to be achieved is available in the form of the input photographs used to compute novel views. It is thus possible to try and quantify the visual quality of the output of image-based rendering algorithms by comparing it to that gold standard, very much like in image or video compression. Tools have been developed in those fields to automate the comparison process: image fidelity metrics.

Just like those metrics are applied frame after frame to obtain a fidelity score between a compressed video sequence and the original, we can apply them input view after input view, comparing the input photograph with a rendering of the scene seen from the same viewpoint, to obtain an aggregate fidelity score for the image-based representation. By exploring the scores resulting from different user parameter choices for the rendering algorithm, optimal parameters can be determined.

There is only one caveat: image-based rendering algorithms tend to possess the desirable property of *epipolar consistency*. This means that when required to render the scene from the exact point of view of one of the input views, they output the exact photograph corresponding to that input view. It is therefore necessary to take care not to use that photograph when rendering the frame with which it is to be compared.

In the course of our work on perceptually guided view selection [MO06], we experimented with a purely physically based visual fidelity metric (root mean square error, *rmse*) as well as a metric taking into account features of the human visual system (Wang *et al.*'s structural similarity index [WBSS04]). We observed that *rmse* performed better in that context, which is identical to that considered in the following parts of this work. This is in line with the results of a broad assessment of visual fidelity metrics over a wide range of video sequences that the Video Quality Expert Group (VQEG) conducted in 2000 [VQE00]. They showed that none of the metrics that relied on a simulation of the HVS exhibited aggregate performances that were statistically distinguishable from peak signal to noise ratio ($psnr = 20\ log_{10}\left(\frac{2^B-1}{rmse}\right)$). As a result, we limit ourselves to the use of *psnr* in this work.

## 3. Background

Buehler *et al*'s [BBM*01] Unstructured Lumigraph Rendering (ULR) algorithm was introduced as a general purpose image based rendering technique able to bridge both ends of the light field sampling *vs.* geometric information compromise. The principle is that it is equally capable of rendering scenes using many views but little geometry as it is using few views but detailed geometry. The "unstructured" means that, contrary to concentric mosaics or Lumigraphs, input images taken from arbitrary viewpoints can be used without a need for a rigid "structured" parameterization of the light field. Those features give it a flexibility for authoring scenes of varying nature, and under varying resource constraints, that others methods lack. It is therefore the approach that we chose to focus on in our work.

An unstructured lumigraph renderer takes as input a polygon mesh approximating the geometry of the scene, a set of images of it and the camera pose information corresponding to each image. The polygon mesh is dubbed a *geometric proxy*, it can be as simple as a mere focal plane, or a mesh as complex as needed to model the scene faithfully. It is important that the registration between the geometric proxy and the input views be known.

The main principle of the technique is to compute a blending field that depicts how the color of each pixel of the desired view is to be modulated from the colors of the corresponding pixels in the input images.

Buehler *et al* design a continuous function that gives a high weight to views that see a given point of the geometric proxy with good resolution from an angle close to that from which it is seen in the desired view. This function only gives a non-null weight to a small number $k$ of best views (in practice, they choose four). The weights are computed by: (i) assigning a penalty value to each input view, (ii) sorting them by increasing penalty, (iii) taking the $k$ first and finally (iv) using a smooth function that gives a weight of 0 to the last view (*i.e*, the $k^{th}$) and the biggest weight to the first one, while ensuring that the weights sum to 1. To achieve interactive frame rates, the blending weights for each input view are not evaluated at each pixel but linearly interpolated from evenly located sample points: the vertices of a triangulation of the image plane constrained by the geometric proxy's edges.

Rendering is thus achieved in three steps:

1. The image plane is triangulated using the constrained Delaunay triangulation algorithm. The vertices used are those of a regular grid spanning the image plane, plus those of the geometric proxy (as projected in the rendered view), plus the centers of the input views (*idem*). The edges of the geometric proxy (*idem*) are used as constraints.
2. Each input view is assigned a blending weight at each vertex of the triangulation.
3. The triangulation is rendered using hardware accelerated projective texture mapping and blending: the final image is accumulated by projecting the input images onto the triangles for whose vertices they have non-null blending weights.

For a given point $P$ in the represented scene, the penalty function that is used to sort the input views (here index by $i$) is the weighted sum of three terms:

$$\mathrm{penalty}(i) = \alpha\,\mathrm{penalty}_{ang}(i) + \beta\,\mathrm{penalty}_{res}(i) + \gamma\,\mathrm{penalty}_{fov}(i) \qquad (1)$$

$\mathrm{penalty}_{ang}(i)$ accounts for the difference of viewing angle between view $i$ and the desired view. $\mathrm{penalty}_{res}(i)$ accounts for the difference in the resolution with which the scene surface at point $P$ is sampled in view $i$ and the desired view. $\text{penalty}_{fov}(i)$ represents the penalty ascribed to an input view $i$ that does not see point $P$, it is different from the previous two in two respects: it is either 0 or infinite, as it does not make sense to let a view that contributes no information to a point's color influence it, and it does not depend on the parameters of the desired view.

## 4. Discussion of the ULR algorithm

Because of the discretization of the blending field, the $\text{penalty}_{fov}(i)$ term is more complex than suggested by the authors. It is indeed possible that a vertex is visible from an input view, but that this input view does not cover the totality of the triangles that share that vertex. If therefore $\text{penalty}_{fov}(i)$ is computed solely on the visibility of that vertex, leading to a non-infinite penalty and thus a possibly non-null blending weight, then by linear interpolation of the blending weight over a partially covered triangle, it can happen that the view contributes to the color of a pixel for which it has no light information, because its blending weight can be non-null there. This possibility is illustrated in Figure 1.

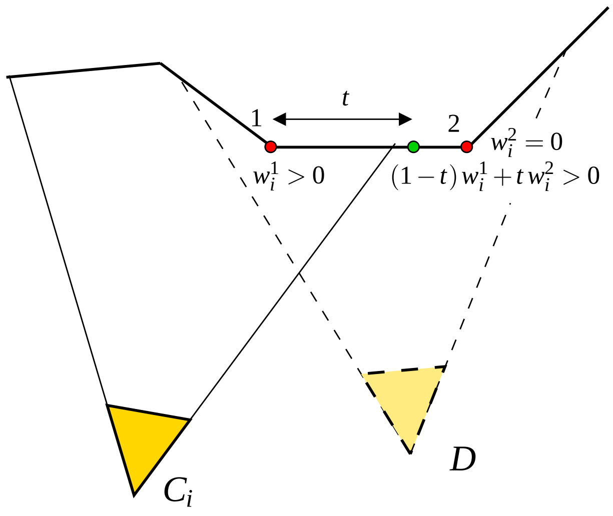

**Figure 1:** *The blending weight for view $i$ at the point in green, visible from the desired view $D$, but not seen by camera $C_i$, is non-null if the blending weight for view $i$ at point 1 ($w_i^1$) is non-null.*

This observation makes it difficult to reconcile the view-dependent triangulation step that Buehler *et al.* propose with an efficient estimation of the $\text{penalty}_{fov}$ term. It would indeed involve determining at each frame whether each input view does cover the whole 1-neighborhood of each vertex of the triangulation, which necessitates a visibility test for each neighboring vertex.

The reason why image plane triangulation was introduced in the first place is to ensure the regularity of the sampling of the blending field. The technique was indeed designed to accommodate a wide range of geometric levels of details: from a mere focal plane to arbitrarily detailed geometric proxies. The triangulation step kicked in at the lower end of that spectrum in order to generate additional points where to sample the blending field given a desired view.

The paper does not discuss another important detail: how to determine the depth of added points. The trivial answer is to project them onto the geometric proxy (however coarse it is). This however leads to special cases that can be hard to resolve when the geometric proxy is non convex, particularly if it has to be done quickly before rendering each frame. The first problem is an occlusion one: when surfaces of the geometric proxy overlap when seen from the desired view, the added point's depth should be that of its projection onto the closest surface (*Cf.* Figure 2). This can be solved, albeit slowly, by reading back from the depth buffer after having draw the geometric proxy in a first pass.

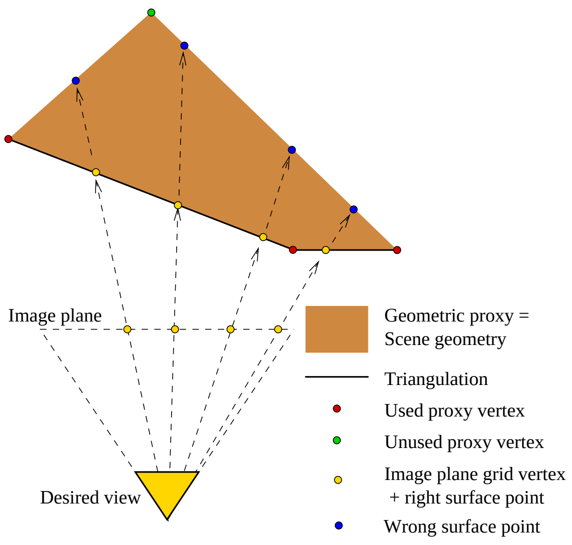

**Figure 2:** *The depth of an image plane grid point has to match that of the closest surface.*

The second problem, illustrated in Figures 3 and 4, arises when the added point projects onto a region of the geometric proxy that is next to an overlap.

There are two possibilities here: either the geometric proxy is accurately describing the scene, or it is not. If it is, simply triangulating the points yields a surface that departs from the actual scene geometry, as illustrated in Figure 3.

The solution to this would be to duplicate the points belonging to the overlapping edge, so that they could be given the proper depth according to whether they belong to a triangle of the overlap or a triangle next to it. However, this

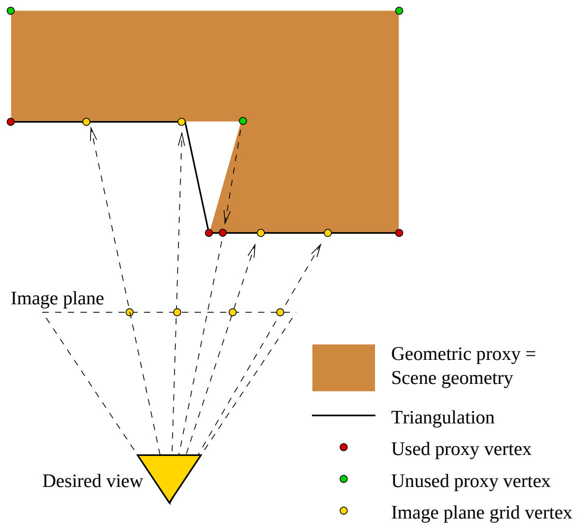

**Figure 3:** *Erroneous triangulation due to an overlap of the geometric proxy in the desired view.*

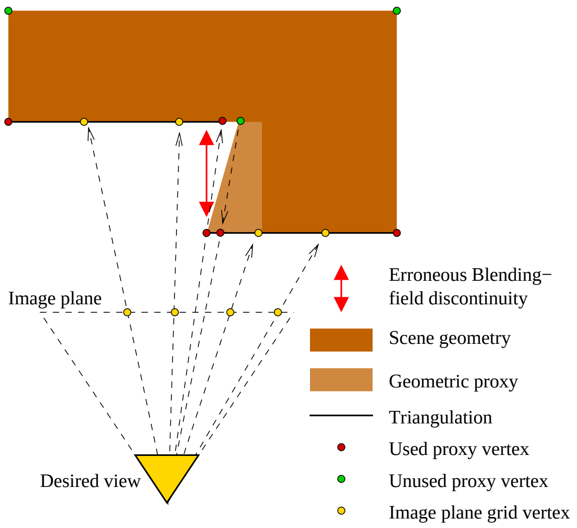

**Figure 4:** *The geometric proxy does not match the scene's geometry. As a consequence, introducing a discontinuity in the triangulation causes a discontinuity of the blending field that does not match the scene, yielding a visual artefact.*

approach breaks the continuity of the blending field, as it leads to different blending weights being attributed to a point depending on the triangle it belongs to (because its 3D position is different from one to the other). If applied to a region where there is a discrepancy between the proxy and actual geometry, the discontinuity introduced by that approach would produce a rendering artefact, instead of reproducing a feature of the scene. Figure 4 illustrates this problem. Thus, barring a way to discriminate between accurate and un-accurate edges, triangulation results in a catch-22.

The paper mentions another drawback of view-dependent triangulation: triangle flipping. Slight interpolation differences can indeed happen from one frame to the other when the change in desired view between them causes the topology of the triangulation to change.

In this context, it appears sensible to simply subdivide the geometric proxy evenly as a pre-process and get rid of the view-dependent triangulation. The only extra cost of this approach is that it will sample the blending field un-necessarily tightly where triangles of the proxy project to a small region of the desired view. In practice, we found that this was counterbalanced by the savings incurred by scrapping the triangulation step.

## 5. User parameter tuning

In our case, since the sets of points over which the blending field is computed remain constant (because there is no image plane triangulation), it is possible to pre-compute the $\text{penalty}_{fov}(i)$ term by projecting the triangle rings (1-neighborhoods) around each vertex of the geometric proxy in each input view (we call those projections *sub-views*, *Cf.* Figure 5). The influence of that term is to give an infinite penalty to input views that do not contribute visual information over the whole 1-neighborhood of the vertex under consideration, in order to avoid using them. Upon rendering, it is thus possible to skip the computation of equation 1 and directly set the overall penalty to infinity when the queried $\text{penalty}_{fov}(i)$ is infinite. In the other case (retrived $\text{penalty}_{fov}(i) = 0$), the $\gamma$ coefficient is made irrelevant.

Consequently, there are 3 user parameters in our implementation of the unstructured lumigraph algorithm. They are: the number $k$ of views that are blended at a given vertex of the geometric proxy, the weight $\alpha$ given to the penalty that accounts for viewing angle discrepancy, and the weight $\beta$ given to the penalty that accounts for viewing resolution discrepancy. $\alpha$ and $\beta$ influence the aggregate penalty as coefficients of a weighted sum. Because the aggregate penalty is used as an ordering, it only matters in relative terms, not in absolute value. Fixing $\alpha$ as 1 and only letting $\beta$ vary therefore yields the same functionality.

Evaluating the aggregate fidelity metric is not fast, particularly if there is a great number of input views. Depending on the dimentionality of the user parameter space, this could make it unpractical to use standard optimization algorithms. This could be addressed by only using a subset of the initial input views for the purpose of evaluating the fidelity metric.

In our case the parameter space is only 2-dimensional, with $k$ varying discreetly over a narrow range of sensible values (from 1 to a reasonable number, i.e. 8). In practice,

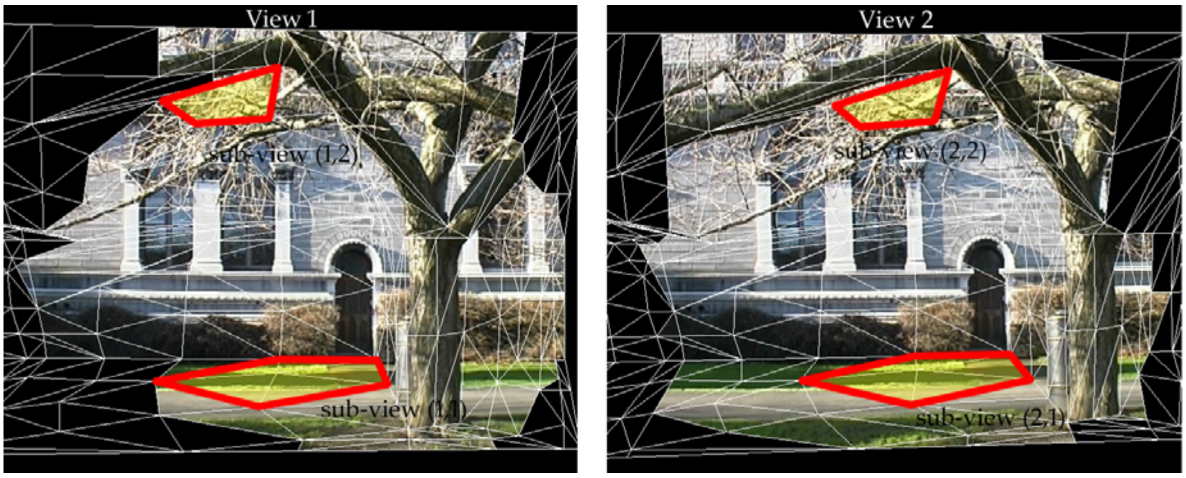

**Figure 5:** *Input views, geometric proxy and* 4 *sub-views, corresponding to the same* 2 *geometric proxy vertices seen in* 2 *different input views. The areas in black correspond to triangles that belong to a sub-view that was not fully covered by that input view.*

we simply evaluate the aggregate metric over a predetermined set of possible parameter pairs and go with the one that gives the highest value (we choose 20 values for β). Figure 6 shows a plot of the aggregate metric as a function of *k* and β for one of our scenes.

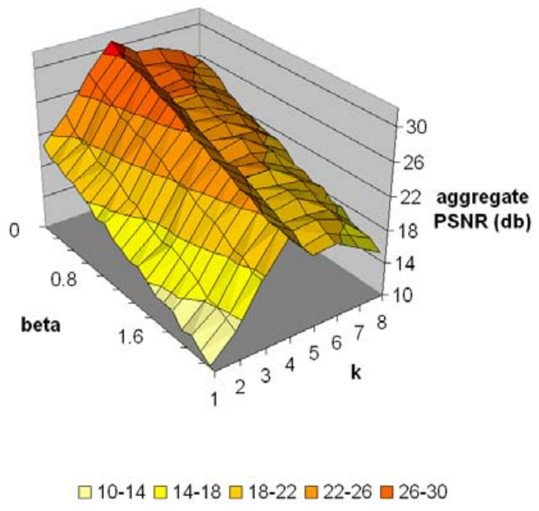

**Figure 6:** *Aggregate metric against k and* β.

## 6. Finer tuning

The functioning of the modified unstructured lumigraph algorithm allows for a finer visual granularity than setting user parameters once for the whole scene. The reason for this is that the blending weight given to a view at a particular vertex has no visual influence outside of the triangles that share that vertex. This, combined with the fact that we do not perform view-dependent triangulation, makes it possible to fine tune the parameters at each vertex of the geometric proxy.

To achieve this, it might make sense to explore the local parameter space at a vertex by evaluating the visual fidelity metric over the corresponding sub-view (see Figure 5). Each image pixel is however influenced by the blending weights at the vertices of the triangle it belongs to, which means that in order to determine the optimal parameters for a given pixel, the parameter spaces of each of the three vertices need to be explored simultaneously. This change of dimensionality from 2 to 6 causes a combinatorial explosion: while it was possible to exhaustively explore 8 values for *k* and 20 values for β for the whole scene, doing so for each vertex requires $(8\times20)^3 \approx 4\,10^6$ evaluations of the fidelity metric over each triangle.

There is another problem: the blending field has to be continuous, therefore each vertex's blending weight is shared by the triangles that share it. This introduces constraints on the parameters that have to be reconciled from one triangle's vertices to that of another. By propagation of these constraints, we can see that the local parameters have to be explored as a whole. Keeping the previous figures, this corresponds to $(8\times20)^n$ evaluations of the metric over the whole scene, where $n$ is the number of vertices of the proxy.

We propose to approximate the triangle to triangle constraints: local optimal parameters are determined independently for each triangle. The final parameters at each vertex are then reconciled by taking the average of the optimal parameters of that vertex in each of the triangles that share it.

This leaves the question of efficiently determining the optimal parameters over each triangle. Fortunately, computing the visual fidelity metric over a triangle is quite fast. It is therefore possible to use a standard optimization technique, such as simulated annealing.

## 7. Results and discussion

So far we have only applied our framework to one test scene (shown in Figure 5) and only to adjust the parameters globaly for the whole scene. Results show that in this particular case, the penalty$_{res}$ term is best ignored alltogether. This is because this scene was captured from a constant distance with a constant focal length, meaning that all input views cover the geometric proxy with similar resolution. We are working on producing results on other test scenes as well as applying our local parameter tuning heuristic.

A major drawback of our approach is that it does not take time into account. Indeed, image-based rendering techniques sometimes exhibit visual artefacts caused by certain changes of viewpoint over time. Detecting them automatically looks like a difficult problem because it appears difficult to explore all possible viewpoint paths if the representation is meant for unconstrained navigation around the scene.

## References

[BBM*01] Buehler C., Bosse M., McMillan L., Gortler S., Cohen M.: Unstructured lumigraph rendering. In *Computer Graphics (SIGGRAPH 01)* (August 2001), pp. 425–432.

[CW93] Chen S., Williams L.: View interpolation for image synthesis. In *Computer Graphics (SIGGRAPH'93)* (1993), pp. 279–288.

[DBY98] Debevec P., Borshukov G., Yu Y.: Efficient view-dependent image-based rendering with projective texture mapping. In *Rendering Techniques 98 Proc. Eurographics Workshop on Rendering* (1998).

[GGSC96] Gortler S., Grzeszczuk R., Sweliski R., Cohen M.: The lumigraph. In *Computer Graphics Proceedings, Annual Conferences Series, SIGGRAPH'96* (1996), pp. 43–54.

[LH96] Levoy M., Hanrahan P.: Light field rendering. In *Computer Graphics Proceedings, Annual Conferences Series, SIGGRAPH'96* (1996), pp. 31–42.

[MB95] McMillan L., Bishop G.: Plenoptic modeling: An image-based rendering system. In *Computer Graphics (SIGGRAPH'95)* (August 1995), pp. 39–46.

[MEG00] Magnor M., Eisert P., Girod B.: Model-aided coding of multi-viewpoint image data. *ICIP'00* (2000), 919–922.

[MO06] Morvan Y., O'Sullivan C.: A peceptual approach to trimming unstructured lumigraphs. *submitted to IEEE Computer Graphics and Applications* (2006).

[SH99] Shum H.-Y., He. L.-W.: Rendering with concentric mosaics. In *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH'99)* (1999), pp. 299–306.

[VQE00] VQEG: *Final Report from the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment*. Tech. rep., VQEG, 2000.

[WAA*00] Wood D. N., Azuma D. I., Aldinger K., Curless B., Duchamp T., Salesin D. H., Stuetzle W.: Surface light fields for 3D photography. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 287–296.

[WBSS04] Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P.: Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing* (April 2004), vol. 13 no. 4.