

A Perceptual Approach to Trimming and Tuning Unstructured Lumigraphs

YANN MORVAN and CAROL O'SULLIVAN

Graphics, Vision and Visualisation Group, Trinity College Dublin

We present a novel perceptual method to reduce the visual redundancy of unstructured lumigraphs, an image based representation designed for interactive rendering. We combine features of the unstructured lumigraph algorithm and image fidelity metrics to efficiently rank the perceptual impact of the removal of subregions of input views (*subviews*). We use a greedy approach to estimate the order in which subviews should be pruned to minimize perceptual degradation at each step. Renderings using varying numbers of subviews can then be easily visualized with confidence that the retained subviews are well chosen, thus facilitating the choice of how many to retain. The regions of the input views that are left are repacked into a texture atlas. Our method takes advantage of any scene geometry information available but only requires a very coarse approximation. We perform a user study to validate its behaviour, as well as investigate the impact of the choice of image fidelity metric as well as that of user parameters. The three metrics considered fall in the physical, statistical and perceptual categories. The overall benefit of our method is the semiautomation of the view selection process, resulting in unstructured lumigraphs that are thrifter in texture memory use and faster to render. Using the same framework, we adjust the parameters of the unstructured lumigraph algorithm to optimise it on a scene by scene basis.

Categories and Subject Descriptors: I.3.0 [Computer Graphics]: General—*Perceptually-based Rendering: Image-based Rendering*

General Terms: Algorithms, experimentation, human factors, performance

Additional Key Words and Phrases: Image-based rendering, perceptual metrics

ACM Reference Format:

Morvan, Y. and O'Sullivan, C. 2009. A perceptual approach to trimming and tuning unstructured lumigraphs. *ACM Trans. Appl. Percpt.* 5, 4, Article 19 (January 2009), 24 pages. DOI = 10.1145/1462048.1462050 <http://doi.acm.org/10.1145/1462048.1462050>

1. INTRODUCTION

The creation of digital visual content has become a major activity in a broad range of industries. Huge numbers of person-hours are invested in related tasks, from modelling to denoising of captured data. There has been significant research in computer graphics aimed at automating or even bypassing some of these tasks. In the field of rendering, one such direction is the image-based approach, which allows new views to be computed from available images and rudimentary geometric information. Perceptually adaptive graphics, albeit so far predominantly explored to achieve machine-hours savings, is another area of great potential.

Author's addresses: email: ymorvan@cs.tcd.ie.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2009 ACM 1544-3558/2009/01-ART19 \$5.00 DOI 10.1145/1462048.1462050 <http://doi.acm.org/10.1145/1462048.1462050>

ACM Transactions on Applied Perception, Vol. 5, No. 4, Article 19, Publication date: January 2009.



Fig. 1. Exterior scene. A: frame of DV footage. B: a novel view rendered after trimming of the unstructured lumigraph.

Research in image-based rendering (IBR) has mainly focused on achieving interactive frame rates at satisfactory visual quality while exploring the trade-off between how finely the light field has to be sampled and how much geometric information to use. From an artist's point of view, all the proposed methods are difficult to leverage: on one hand, depth-map, dense correspondence or optic flow based techniques [Heigl et al. 1999] rely either on the availability of an accurate geometric model or on computer vision methods such as stereo matching or range scanners. These perform poorly on scenes that contain complex occlusion boundaries, large surfaces of uniform irradiance or highly reflective materials. On the other hand, methods that put the emphasis on a dense sampling of the light field are often based on a rigid parametrisation that requires expensive and impractical gantries to constrain camera positions. In Gortler et al. [1996], views can be acquired in an unstructured way, for example using a hand-held camera, but a lossy rebinning preprocess is then needed.

Buehler et al. [2001] propose their unstructured lumigraph rendering algorithm as a flexible method that bridges both ends of the light field sampling versus geometry compromise. It behaves similarly to light field rendering when given many views and a single plane, and like view dependent texture mapping [Debevec et al. 1998] when given fewer views but more detailed geometry. A major drawback of unstructured IBR techniques is that they cannot exploit any parametrisation properties to guide compression, making it difficult to compress interview redundancy. Selecting a set of views that minimises redundancy is therefore crucial.

Perceptually adaptive rendering is particularly well suited to global illumination, see the works of Myszkowski et al. [1999], Yee et al. [2001] and Stokes et al. [2004] among others. One reason is that perceptual metrics, whether based on complex simulations of the human visual system (HVS) or on implicit models of its properties, tend to be computationally expensive so that their own cost can counterbalance the savings they achieve (there exist works that take up this challenge in the context of interactive rendering: Dumont et al. [2003], Walter et al. [2002], Williams et al. [2003]). Another reason is that intermediate results of global illumination algorithms that proceed by progressive refinement can be substituted for the gold standard that most perceptual metrics need.

In the context of unstructured IBR, view selection is a preprocess, which makes the use of perceptual metrics more tractable. Furthermore, the input images themselves can be used as a gold standard. The initial set of views then needs to be dense enough to sample the scene properly. This will typically be the case when working from video sequences.

This article proposes a framework to facilitate the authoring of thrifty unstructured lumigraphs, whose rendering parameters are optimised for the target scenes. We make use of image fidelity metrics

as a criterion to remove redundancy from the original dataset and adjust parameters. Three metrics are considered, respectively of a physical, statistical and perceptual nature. We perform a user study to investigate which metric performs best, as well as validate that our framework yields better results than the alternatives. The scope of this work is limited to static scenes under constant illumination.

2. RELATED WORK

Hlavac et al. [1996] were the first to study the problem of view selection for unstructured IBR, more specifically in the context of view interpolation. They demonstrated their method in the case where camera positions are limited to one degree of freedom. It consists of growing view position intervals until the quality of the interpolation over them falls under a threshold, and then keeping the views at their bounds. They point out that the computational cost of the interval growing algorithm explodes in the general case.

In Fleishman et al. [2000], the geometry of the scene is assumed to be known and its surfaces to be Lambertian. Thus, views can be selected without a priori knowledge of the corresponding images. A heuristic is presented to determine a reduced set of views that ensure coverage of all the scene polygons with quality superior to some user specified threshold. Vazquez et al. [2001] propose a similar technique inspired by information theory, using viewpoint entropy to guide the selection process.

Schirmacher et al. [1999] use a lumigraph representation to interactively render high quality global illumination solutions. Each frame of the solution being expensive to compute, they propose an iterative method to progressively add views to the lumigraph that are predicted to most increase rendering quality. Coombe et al. [2005] introduce a system that lets an author interactively create a surface light field online by giving him feedback on what views to capture next. Views are incorporated on the fly into the light field using an online SVD algorithm. Their method presupposes a reasonably accurate geometric model of the captured scene.

The framework we propose is based on assumptions different from those made by these two last works: a dense set of views, along with their camera pose information, is available, but geometric information can be sparse and/or inaccurate. Those assumptions fit the convenient capture scenario of a user taking a series of shots, or a video sequence, with a hand-held camera from various locations around the scene. In this scenario, structure from motion algorithms will be able to reconstruct a number of feature points but will rarely deliver accurate dense geometry with correct topology.

In this context, our framework aims to prune visual information that contributes less to rendering quality. As such it is related to the various light field compression techniques that have been developed. Further discussion of this topic can be found in the works of Xin and Gray [2003]. Our approach puts more emphasis on facilitating the authoring process. The main advantage of our framework is that it transparently deals with the compromise between geometric accuracy and the light field sampling rate.

3. PROPOSED FRAMEWORK

Given a perceptual measurement tool that evaluates how similar an image is to a reference, it is straightforward to define the perceptual quality score (PQS) of a subset of input views. This is calculated by applying the measurement tool to pairs consisting of each initial input view and its reconstruction by the IBR algorithm using that subset, then taking the sum. From there, we can assess the perceptual degradation caused by the removal of an input view by taking the difference in perceptual quality score of renderings computed with and without it. The higher the degradation, the more view dependent information that input view captures, and the less redundant it is. For clarity, we will call the views used for the purpose of computing perceptual quality scores touchstone views.

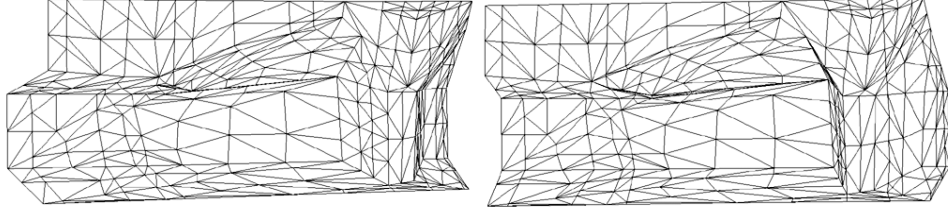


Fig. 2. A geometric proxy corresponding to the scene shown in Figure 1, shown from two viewpoints.

We show how the unstructured lumigraph rendering algorithm makes it easy to exploit spatial coherence to speed up the computation of the perceptual degradation caused by the removal of an input view. We then describe the perceptual measurement tools that we have chosen to use and justify our choice. The details of our greedy pruning process are then provided, followed by a discussion of the choice of touchstone views. Finally, we explain how the use of perceptual quality scores can be used to adjust rendering parameters.

3.1 Context: Unstructured Lumigraph Rendering

Buehler and coworker's [2001] unstructured lumigraph rendering (ULR) algorithm is a general purpose IBR technique that 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* (Figure 2). 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 obtained by blending the colors of the corresponding pixels in the input images.

Buehler et al. [2001] design a continuous function that gives a high blending weight to views that see a given point P 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 nonnull weight to a small number k of best views (in practice, they choose four) and is computed as follows:

Firstly, a penalty value $p(i)$ is assigned to each input view i . It is infinite if P is not visible in i , otherwise it is the weighted sum of an angular penalty $p_{ang}(i)$ and a resolution penalty $p_{res}(i)$:

$$p(i) = \alpha p_{ang}(i) + \beta p_{res}(i) \quad (1)$$

The term $p_{ang}(i)$ accounts for the difference of viewing angle between view i and the desired view. It is computed as the angle between the line going through P and the optical centre of view i (C_i), and the line going through P and the optical centre of the desired view (C_d). The term $p_{res}(i)$ accounts for the difference in the resolution with which the scene surface at point P is sampled in view i and in the desired view. For efficiency, it is approximated as:

$$p_{res}(i) = \max(0, \|P - C_i\| - \|P - C_d\|) \quad (2)$$

Second, the k input views of lowest penalty are selected and the worst penalty in those k , p_{thresh} , is recorded.

Third, preliminary blending weights for the selected views are computed as:

$$w(i) = 1 - \frac{p(i)}{p_{thresh}} \quad (3)$$

Finally, the blending weights are normalised so that they sum to unity.

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 the triangulated geometric proxy.

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 optical centres of the input views (*ditto*). The edges of the geometric proxy (*ditto*) 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 nonnull blending weights.

In the context of our framework, we can see that, for a given touchstone view, the ULR algorithm makes it possible to determine the list of triangles over which visual changes will happen when computing the perceptual degradation caused by the removal of an input view. It is simply the list of triangles that have at least one vertex whose set of k best views contains that input view. If that list is empty for all triangles in a given touchstone view, its contribution to the overall perceptual quality does not need to be updated. Moreover, if we are able to compute the perceptual measure locally, time can be saved by computing it only over triangles where a visual change has occurred.

3.2 Perceptual Measure of View Utility

For our framework, we considered three image fidelity metrics: the traditional root mean square error (RMS), Yee and Newman's [2004] PerceptualDiff (PDIFF), and the structural SIMilarity index (SSIM) of Wang et al. [2002]. We had considered including Daly's visible difference predictor (VDP) [Daly 1993] using the most recent implementation of Mantiuk et al. [2005], but its longer computation times (between one and two orders of magnitude) made our framework intractable in its current iteration. In this work, all the metrics considered were computed only on the luminance channel, thus disregarding chrominance information. This was motivated by computation time concerns, as well as the fact that for each of the metrics considered, color handling is an orthogonal addition that we thought better to investigate at a later point.

3.2.1 RMS. RMS and related peak signal-to-noise ratio (PSNR), quantify the value-wise similarity between two signals \mathbf{x} and \mathbf{y} . They are expressed in Equation 4, assuming the signals are registered and sampled by n corresponding values encoded with B bits.

$$\text{RMS} = \sqrt{\frac{1}{n} \sum_{i=1}^n |\mathbf{x}_i - \mathbf{y}_i|^2} \quad \text{PSNR} = 20 \log_{10} \left(\frac{2^B - 1}{\text{RMS}} \right) \quad (4)$$

These two quantities are commonly used in computer graphics as a conservative estimate of how well techniques perform compared to a gold standard, for instance in compression or global illumination rendering.

3.2.2 SSIM. In 2000, the Video Quality Expert Group (VQEG) conducted a broad assessment of visual fidelity metrics over a wide range of video sequences. Results showed that none exhibited aggregate performances that are statistically distinguishable from PSNR [VQEG 2000].

Wang et al. [2002] investigate possible causes for those disappointing results. They enumerate the assumptions made by traditional metrics based on error sensitivity and perceptual channel decomposition

and discuss their validity. The assumption that interaction between channels is weak is found particularly questionable. They also illustrate the limitations of Minkowski error pooling, which is widely used to combine information over channels in those metrics. Stating that “the main function of the human eyes is to extract structural information from the viewing field, and the human visual system is highly adapted for this purpose. Therefore, a measurement of structural distortion should be a good approximation of perceived image distortion,” they propose a simple metric integrating three structural factors, namely loss of correlation, mean distortion and variance distortion. As such, their approach is a hybrid between the physical and the perceptual.

Given two aligned nonnegative image signals \mathbf{x} and \mathbf{y} , their structural similarity index (SSIM) is defined as:

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (5)$$

where μ_x (μ_y) is the mean of \mathbf{x} (\mathbf{y}), σ_x^2 (σ_y^2) is the variance of \mathbf{x} (\mathbf{y}) and σ_{xy} is the covariance of \mathbf{x} and \mathbf{y} . We refer to Wang et al. [2004] for a detailed derivation of this formula from luminance, contrast and structural similarity terms. C_1 and C_2 are terms linked to the dynamic range of the signal introduced for stability:

$$C_1 = (K_1L)^2 \quad C_2 = (K_2L)^2 \quad (6)$$

Where L is the dynamic range of the signals and K_1, K_2 user defined constants.

Wang et al. evaluate the SSIM locally over the images using a sliding window approach. To avoid blocking artifacts resulting from a square window, they use a 11×11 circular-symmetric Gaussian of 1.5 samples standard deviation normalised to unit sum. This leads to a rewriting of μ_x , σ_x^2 and σ_{xy} over each window as follows (w_i are the Gaussian weights of the window):

$$\mu_x = \sum_{i=1}^N w_i x_i, \quad \sigma_x^2 = \sum_{i=1}^N w_i (x_i - \mu_x)^2, \quad \sigma_{xy} = \sum_{i=1}^N w_i (x_i - \mu_x)(y_i - \mu_y) \quad (7)$$

We justify our inclusion of SSIM as follows:

- As Wang et al. show, when predicting the visual fidelity of a wide range of images, their method can compare favourably to RMS, peak signal to noise ratio, and more importantly, techniques based on models that reproduce the error sensitivity of the HVS.
- Spatial frequency masking is a HVS property accounted for by these latter techniques, as opposed to SSIM, but taking it into account is not clearly desirable for our purpose. This is because changes of the viewing context, like a change of viewpoint or the presence of an occluder between the image based rendered object and the observer, will cause arbitrary modifications of the visual information surrounding a point of the object.
- The SSIM index is less computationally intensive because it takes a statistical approach as opposed to a signal processing one (needed to simulate the response of the HVS as in Daly [1993] or Lubin [1995]), which requires complex transforms to be applied.
- It has a negligible initial setup cost, an important advantage if we want to evaluate it on many small triangles and not just a few whole images, as mentioned in Section 3.1.
- It is straightforward to implement as a multipass fragment shader.

SSIM's evaluation is based on a sliding window mechanism and computes a score for each pixel (we invite the reader to consult Wang et al.'s [2002] for further details). The value of the SSIM index over each triangle is obtained by taking its average over the relevant pixels.

3.2.3 **PDIFF.** Yee and Newman’s [2004] (PDIFF), which they put forward in the context of production testing, is based on the simplified version of the VDP by Ramasubramanian et al.’s [1999]. Like the VDP, it accounts for three features of the HVS: amplitude nonlinearity, sensitivity variation as a function of spatial frequency, and visual masking. For efficiency purposes, the original VDP’s decompositions of the signal into different bands in the frequency domain and different orientations are discarded. This allows for a purely spatial approach at the cost of a more rudimentary modelling of the visual masking phenomenon (because interactions between signal components based on frequency and orientation similarity are not considered). Since no use is made in this work of the ability of PDIFF to handle chrominance, for our purposes, it is identical to Ramasubramanian and colleagues metric, which is computed in this way:

A spatial frequency hierarchy is constructed from the luminance channel (Y) of the reference image. It is efficiently computed using Burt and Adelson’s Laplacian pyramid and used to determine how sensitivity to contrast changes decreases with increasing frequency. The normalised contrast sensitivity function is then computed and multiplied by the masking function given in Daly [1993] to obtain the combined threshold elevation factor, F . These functions depend on the field of view occupied by the image, its resolution and the brightness rating of the target display device. Two corresponding pixels of coordinates (x, y) in the reference (Y_{ref}) and test (Y_{test}) images are marked as different if the following test fails:

$$|Y_{ref}(x, y) - Y_{test}(x, y)| > F \cdot T(Y_{adapt}) \quad (8)$$

where T is the psychophysical function mapping adaptation levels to detection thresholds, and Y_{adapt} is the adaptation luminance, computed as the average of pixels in a one degree radius from the Y channel of the reference image.

3.3 Greedy Subview Selection

In order to take advantage of the locality of view-dependent phenomena—be they inherent to the scene, such as reflective surfaces, or caused by geometric proxy inaccuracies—our framework does not discard whole input views but subregions of them: The triangle ring surrounding each vertex of the geometric proxy is projected into each input view. The resulting image areas are treated as *subviews* (thus a subview is identified by the input view it belongs to, and a vertex of the geometric proxy). This choice is justified by the nature of the ULR algorithm: it operates on vertices, whose list of blending weights only affect the neighbouring triangles.

For subviews to be consistent from one primary view to the other, it is necessary to drop the view dependent triangulation step of the original ULR algorithm. To maintain proper sampling of the blending field (see Section 3.1), we simply subdivide the initial geometric proxy evenly. The only disadvantage of this approach is that it will sample the blending field unnecessarily tightly where triangles of the proxy project to a small region of the desired view. On the other hand, skipping the projection of the image plane grid points onto the geometric proxy and the constrained Delauney triangulation introduces time savings, as this had to be done for each frame.

We then greedily prune the subviews in order of increasing perceptual impact: at each step, the discarded subview is the one whose removal causes the least perceptual quality degradation, as defined in section 3.2. Brief pseudocode is given in Algorithm 3.1.

The output of the algorithm is a list of all initial subviews sorted by the estimated order in which they should be removed to minimise perceptual degradation.

Algorithm 3.1: SUBVIEW ORDERING

```

Compute the initial perceptual qualities with all subviews in use
while there remain subviews
    do {
        for each remaining subview
            do {
                for each triangle in each touchstone view
                    that it affects
                    do {
                        Compute the visual degradation
                        caused by its removal.
                    }
                Compute the average visual degradation
                for this subview.
            }
        Append the subview whose removal causes the
        least degradation to the list of ordered subviews.
        Remove that subview from the list of remaining
        subviews.
    }

```

3.4 Touchstone Views Selection

The use of our framework raises the question of how to choose the touchstone views. It is an authoring choice that depends on the final purpose of the representation. In graphical applications, image based representations are particularly well suited to midrange visual content: content that is not far enough away to be rendered as a static billboard, but yet not close enough for the user to interact with. The ULR algorithm is better suited for this than most IBR techniques because of the ease with which it can be incorporated into a typical hardware accelerated 3D engine.

In this context, parts of the captured objects will end up being occluded by closer range objects when viewed from the area that users can navigate. Apart from the obvious choice of limiting the set of touchstone views to the navigable viewing region, our method lets the author leverage these known occlusions. By rendering the occluders in each touchstone view, areas that are irrelevant to visual quality (because ultimately unseen) can be masked out, as allowed by our implementations of SSIM and PDIFF. Figure 3(b) illustrates this.

If the artist is given information on where users are likely to spend the most time, and which are the most likely viewing angles, he can adjust the density of touchstone views accordingly. He can either discard or enforce high-frequency phenomena, such as a glare in a window, by discarding or choosing to keep the touchstone views that exhibit it Figure 3(c).

The choice of touchstone views is related to the behaviour of the perceptual measurement tool. In our experience, the quality of the registrations obtained with commercial tracking software was good enough for the tools we chose to behave well. However, for a scene where some views are not properly registered, any perceptual metric with a registration requirement acts like a double-edged sword. If a badly registered view is used as a touchstone view, it will upset the behavior of the framework for subviews taken from a neighboring viewpoint. If on the other hand the badly registered view is excluded from the set of touchstone views, its subviews will be automatically discarded by the framework, as their removal will increase rendering quality.

Being part of the authoring process, the choice of touchstone views is very much case dependent. To evaluate the potential of our approach, we chose to place ourselves in the neutral case where all views are equally desirable. Thus, we retain one input view out of every two as touchstone view, in an even distribution. In Section 5.3, we analyze the outcome of picking different touchstone views

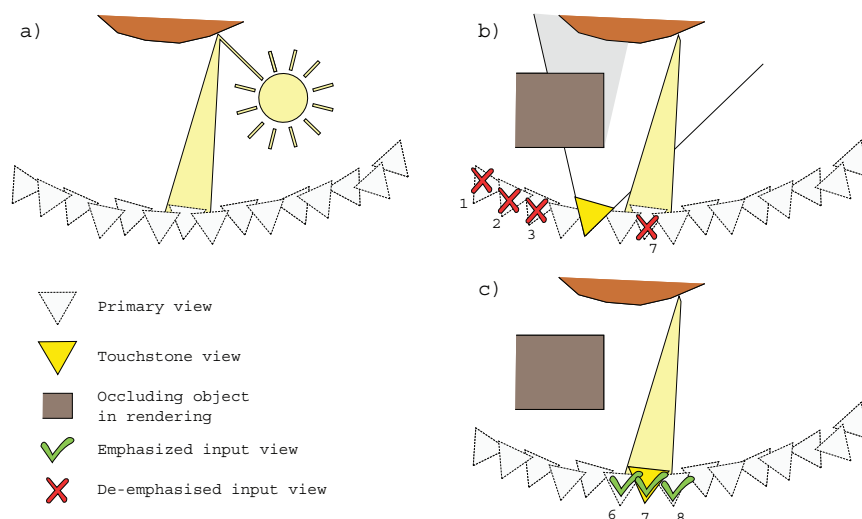


Fig. 3. (a): The captured scene contains a specular highlight. (b): The information captured by input views 1, 2 and 3 is irrelevant because it will be occluded upon rendering in the selected touchstone view; the specular highlight captured by input view 7 is in disagreement with the selected touchstone view, which does not see it: it is de-emphasized. (c): Input views 6, 7 and 8 capture part of the specular highlight that the selected touchstone view sees, while other input views do not: they are emphasized.

in a practical case. This case corresponds to situations where a skilled user is not needed or available.

3.5 Parameter Adjusting

One authoring drawback that image-based rendering 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. 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.

By exploring the metric scores resulting from different user parameter choices for the rendering algorithm, optimal parameters can be determined.

3.5.1 Case of the Unstructured Lumigraph. The rendering parameters of the ULR algorithm, α , β and k , appear in the function that it uses to compute blending weights for each input view (see Section 3.1).

In the same way that we use the aggregate visual fidelity metric to compare rendering quality with different sets of retained views, we can use it to compare rendering quality with different choices of user parameters. This lets us adjust rendering parameters to suit particular scenes.

Because the aggregate penalty $p(i)$ (see Equation 1) is used as an ordering, it only matters in relative terms and not as an absolute value. Fixing α as 1 and only letting β vary therefore yields the same functionality.

The parameter space is thus two-dimensional, with k varying discretely over a narrow range of suitable values: 2 is the minimum value of k that guarantees the smoothness of the blending field. 8 is a reasonable upper bound as it corresponds to trilinear interpolation over the vertices of a cube containing the desired view. Therefore, while it is possible to explore parameters as an optimization problem, in practice we simply evaluate the aggregate metric over a predetermined set of possible parameter pairs

and pick the one that gives the highest value. In practice, we choose values of β so that it doubles in importance with respect to $\alpha = 1$, covering roughly 3 orders of magnitude: $\{\beta_i = 2^i | i \in [-6, 6]\}$, *i.e.* 13 values.

3.5.2 Finer Tuning. Just as the finer visual granularity of the unstructured lumigraph algorithm was used to perform view selection at the subview level, it can be exploited to fine tune the rendering parameters at each vertex of the geometric proxy (which is static, as discussed in Section 3.3).

To achieve this, one solution is to explore the local parameter space at a vertex by evaluating the visual fidelity metric over the corresponding subview. Each frame-buffer 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 7 values for k and 13 values for β for the whole scene, doing so for each vertex requires $(7 \times 13)^3 \approx 7 \cdot 10^5$ 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 $(7 \times 13)^n$ 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. Since computing the visual fidelity metric over a triangle is fast, it is, therefore, possible to use a standard optimisation technique, such as simulated annealing.

4. IMPLEMENTATION

In this section, we detail how the metrics and the subview selection algorithm were implemented. We then describe how to leverage the removal of subviews to effectively trim unstructured lumigraphs for rendering.

4.1 Metrics

4.1.1 SSIM. We implement a modified version of SSIM using OpenGL's fragment shader mechanism. Care needs to be taken to disregard the signal outside of a triangle when computing the measure over it. To achieve this, we use the alpha channel as a flag indicating whether a pixel should be taken into account (1.0) or not (0.0). This allows us to accumulate in a register the proper sum of the window weights that should be used when normalizing, by discarding the weights associated with undesirable pixels. Note that this is different from assuming that the signal is uniformly null outside of the triangle. This method tends to make the SSIM index more sensitive around the corners of triangles, as this is where the support of the weighting window is most narrow. We hypothesize that this sits well with the fact that the corners are precisely where the ULR algorithm samples the blending field with the most accuracy.

The whole reference image is initially stored in texture unit 1. Before the triangle to be compared is rendered using the ULR algorithm, we set the scissors to its bounding rectangle expanded by the radius of the SSIM window (11) and clear that area with black and alpha = 0. After the triangle is rendered, the expanded rectangle containing it is copied into texture unit 0. The SSIM is then computed over the

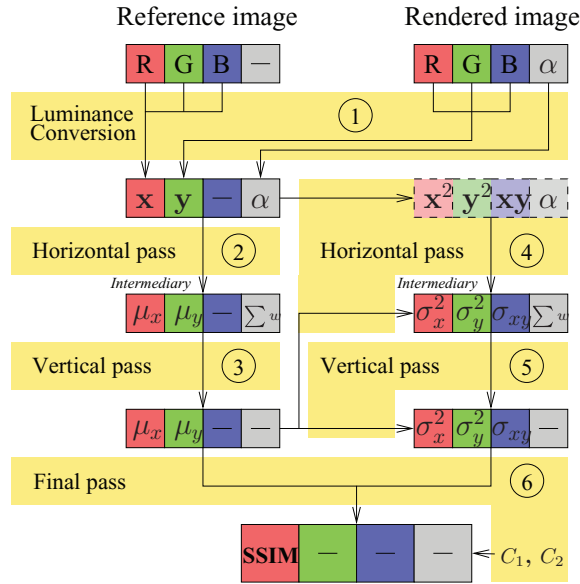


Fig. 4. The six fragment shader passes used to compute SSIM indices.

triangle in six fragment shading passes, taking advantage of the separability of the weighting window. The process is illustrated in Figure 4. (see Section 3.2.2 for clarification of the mathematical symbols.)

- (1) The luminance of each signal is computed and stored in the red and green channels, while the alpha channel is set to that of the test signal (rendered triangle). The result is copied into texture unit 2.
- (2) The horizontal components of μ_x and μ_y are accumulated, reading from texture unit 2, and stored in the red and green channels. The horizontal component of the sum of weights is accumulated in the alpha channel. The result is copied into texture unit 3
- (3) The vertical components of μ_x and μ_y and the sum of weights are accumulated, reading their intermediary result from the previous pass from texture unit 3. The final values of μ_x and μ_y are then obtained by dividing their accumulated values by the sum of weights. They are stored in the red and green channel. The result is kept in texture unit 3.
- (4) Reading from texture unit 2, x^2 , y^2 and xy are computed and stored in the red, green and blue channels. The alpha channel is set to that of the test signal (rendered triangle). The result is copied into texture unit 2.
- (5) The horizontal components of σ_x^2 , σ_y^2 and σ_{xy} are accumulated in the red, green and blue channels, reading x^2 , y^2 and xy from texture unit 2, and μ_x and μ_y from texture unit 3. The weights are accumulated in the alpha channel and the result copied into texture unit 2.
- (6) The vertical components of σ_x^2 , σ_y^2 and σ_{xy} are accumulated in the red, green, and blue channels, reading their intermediary result from the previous pass from texture unit 2, and μ_x and μ_y from texture unit 3. The weights are accumulated in the alpha channel. The final values of σ_x^2 , σ_y^2 and σ_{xy} are then obtained by dividing their accumulated values by the sum of weights. At that point, all the values necessary for the evaluation of the SSIM are available, including C_1 and C_2 , which have been passed as parameters to the last fragment program. The result is stored in the red channel.

Following the experimental findings of Wang et al.'s [2002], we take $K_1 = 0.01$ and $K_2 = 0.03$ in Equation 6. Our standard OpenGL display setup means that the luminance dynamic range is 8 bits, *i.e.* $L = 255$. This yields $C_1 \simeq 6.5$ and $C_2 \simeq 58.5$.

The value of the SSIM index over each triangle is obtained by taking the average over the contributing fragments. This is done in software after reading the red channel from the area of the frame buffer bounding the triangle (this time without the window safety margin) into client memory.

4.1.2 PDIFF. We made use of Yee and Newman's [2004] code for their PDIFF metric, which they have made publicly available.

In the context of our framework, the computation of the Laplacian pyramid proved a manageable overhead in the application of PDIFF to individual triangles. Unlike SSIM, which outputs a normalised score for each pixel, PDIFF's output consists of a number of pixels where the metric predicts viewers will perceive a difference. To obtain a score over a triangle to use within our framework, we took the ratio between the number of pixels predicted indistinguishable and the total number of pixels covered by the triangle.

4.2 Subview selection

The subview selection algorithm can be sped up through the use of caches. We explain the functioning of our implementation then analyse its time complexity.

4.2.1 Caching. Some book-keeping is necessary to avoid recomputing the perceptual quality score over triangles that were not affected by the last subview removal. An array is initialised with the perceptual quality score (PQS) of each triangle for each touchstone view obtained when using every input view for rendering. It is accessed whenever the perceptual degradation over a triangle needs to be recomputed in order to compare the new appearance with the initial (and best) appearance. We update the following data structures after each subview removal:

- A1 A cache containing the PQS degradation over each triangle in each touchstone view for each primary view to be potentially next removed.
- A2 A cache containing the next PQS degradation resulting from the potential removal of each subview.
- L3 The list of remaining subviews.
- L4 The list of remaining input views ordered by ULR blending weight (see Section 3.1) for each vertex in each touchstone view.
- L5 The sublist of remaining input views that will affect the rendering of each triangle in each touchstone view.
- L6 The list of triangles (grouped by the touchstone view they belong to) to whose rendering each remaining subview contributes.

A1 only needs to be updated when the last removed subview contained that triangle. A2 is obtained by averaging the PQS degradations of all triangles of a subview over all relevant touchstone views. It gets updated if the removal of the last subview affected the PQS degradation of a relevant triangle in any touchstone view.

L3 is self explanatory. L4 is included in order to avoid having to go through all the remaining input views each time the blending weights are computed when rendering a triangle. Here, we have to note that the ULR algorithm is designed to approximate epipolar consistency. This means that if we were to evaluate the PQS on a touchstone view, while using that very touchstone view as an input view, it would be selected as first among the k best views and given the highest weight for each vertex. Removing other input views would then cause the PQS to increase all the more if their weight was important, because

Table I. Test scenes statistics (note that we do not count subviews that are not blendable for lack of visibility of the whole triangle ring)

	Views	Vertices	Faces	Subviews
Exterior	143	268	491	18638
Library	91	244	434	15625
Mezzanine	68	444	808	14508
Objects	175	119	214	20674

the relative weight of the touchstone view itself would be increasing. Such behavior is the opposite of what we want to measure. To correct it, for each vertex of a given touchstone view of L4, we initially remove that touchstone view from the list of potential input views. This ensures that each touchstone view is not used by the ULR algorithm when rendering a triangle with the purpose of comparing it with its appearance in that specific touchstone view.

L5 is built from L4 by taking the union of the first $k + 1$ input views over the vertices of each triangle. Indeed, to compute the new PQS over a triangle resulting from the potential removal of each single input view that affects it, we need to render the triangle without that view, and, therefore, need the $(k + 1)^{th}$ input view that will fill the gap for each vertex where the removed view was present. (Taking k input views would leave us with only $k - 1$ views to blend after having removed a candidate for removal).

When a subview is selected for removal, other subviews will make their way onto the list of k best views for each vertex in each touchstone view it affected. This has an impact on which touchstone views to render on the next step to evaluate the perceptual degradation caused by the potential removal of each remaining subview. L6 stores that information and it is, therefore, updated by identifying which input view “filled the gap” left by the removal of the last subview for each affected triangle in each touchstone view.

4.2.2 Time Complexity. The worst case scenarios for the greedy view-selection algorithm are scenes where all the triangles of the geometric proxy are visible in each input view (and, therefore, each touchstone view). This maximises the number of triangles whose PQS has to be updated in the cache when a subview is removed. (Updating L3 to L6, and fetching from and writing to the caches takes negligible time compared to evaluating the PQS). For a given scene, let n be the number of input views, m the number of touchstone views, and v the number of vertices of the geometric proxy. The probability of an input view being part of the list of blended views for a given vertex is $\frac{k}{n}$, thus the probability that the removal of an input view will affect a triangle in a particular touchstone view is: $1 - (1 - \frac{k}{n})^3$ (i.e., the probability that none of its vertices’ blending lists contain that input view). Let us define d as the average number of triangles shared by a vertex of the geometric proxy. By definition, the removal of a given subview can only affect at most d triangles. Therefore, on average, the number of re-evaluations of the PQS (n_{pqs}) to update the cache after a subview is removed is:

$$md \left(1 - \left(1 - \frac{k}{n} \right)^3 \right) \quad (9)$$

If we make the conservative hypothesis that the geometric proxy is an omni-triangulated polyhedron, the cost of initializing the cache is $2vnm$ evaluations of the PQS. Putting it all together, we get:

$$n_{pqs} = 2vnm + vnm d \left(1 - \left(1 - \frac{k}{n} \right)^3 \right) \quad (10)$$

Table II. Approximate time taken to order all subviews by increasing perceptual importance (hours)

	RMS	SSIM	PDIFF
Exterior	≈ 5 h	≈ 6 h	≈ 8 h
Library	≈ 5 h	≈ 5 h	≈ 7 h
Mezzanine	≈ 6 h	≈ 7 h	≈ 9 h
Objects	≈ 5 h	≈ 6 h	≈ 8 h

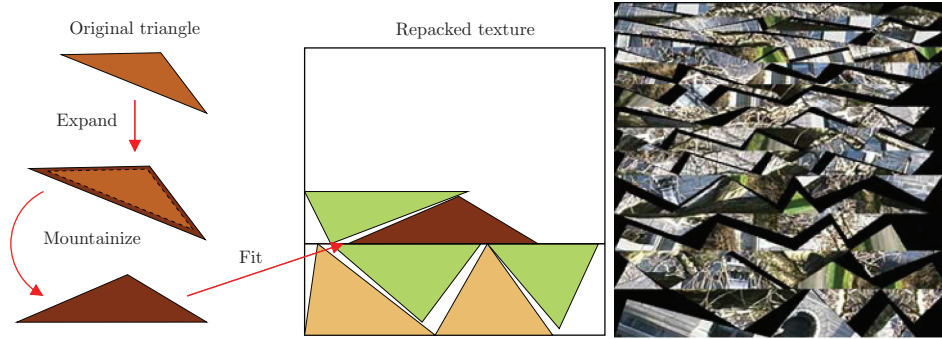


Fig. 5. Triangle packing.

This gives a complexity in $\mathcal{O}(vnm)$. Assuming values of $k = 4$, $d = 6$ and making a first-order approximation on the probability term, the following rule of thumb is obtained: $n_{pqs} = vnm(2 + \frac{72}{n})$. In practice, the amortized cost of evaluating the PQS on a triangle (PQS_{cost}) is lower in the cache initialisation stage than in the updating stage, because of the more frequent and less-structured texture changes in the later. Let us take the example of a scene made of a geometric proxy with 400 vertices and 200 input views, half of which are used as touchstone views. For computation time to remain in the order of hours, PQS_{cost} should be strictly lower than 2 milliseconds. This is in keeping with RMS, PDIFF and SSIM run times on images of the size of a typical triangle (less than 1000 square pixels) on a modern desktop computer. Overall, our implementation of the algorithm is slightly slower than analysed here, as shown in Section 5, Table II.

4.3 Texture Atlas Generation

The ULR algorithm uses hardware accelerated projective texture mapping and colour blending to achieve interactive frame rates. To capitalize on the visual redundancy removal achieved by pruning subviews, input views (now with holes in them) have to be repacked into a texture atlas to optimise the use of texture memory.

The texture coordinates of each triangle in each input view can be recovered from the hardware’s automatic texture generation mechanism. We adapt Hale’s [1998] triangle packing heuristic to our needs. It begins by rotating triangles into “mountain” shapes and then mirrors them horizontally or vertically to tightly fill image rows of decreasing height. OpenGL’s rasterizing functionality is used to produce the transformed textures by rendering each triangle in its new position with its position in the original image as texture coordinates. Because of the interpolation that takes place upon rasterizing, the original triangles have to be expanded with a one-pixel margin, otherwise the background colour of the texture atlas could generate seams between triangles upon rendering. The packing process is illustrated in Figure 5.

Since we are taking triangles from many input views, this method would lead to the big triangles being packed in the first textures and the small ones in the last textures, regardless of the input view they belong to. Thus, to render a view from a certain viewpoint, many more textures than necessary would need to be in texture memory. To avoid this, we apply Hale's heuristic on clusters of triangles. These clusters are determined by applying the k-means algorithm to the triangles, based on the euclidean distance between the optical centres of the input views they belong to. The number of clusters is taken as $1.3 A_{triangles}/A_{textures}$, where $A_{triangles}$ is the sum area of retained triangles, and $A_{textures}$ is the area of the target texture images. The 1.3 constant comes from Hale's empirical findings and is a conservative estimate of the efficiency of the packing. We take advantage of the fact that the heuristic is flexible in terms of packing height to adjust texture heights to specific cluster areas. A limitation of this approach is that it makes the assumption that the clusters are not of widely varying size, which means that input views are assumed to be roughly evenly distant from each other, or that similar numbers of views are taken around distinct locations.

5. RESULTS

We acquired four video sequences of different scenes using a hand-held Canon XL1 PAL DV camcorder. Camera poses were recovered using PFTrack 3.0, a commercial camera tracking solution from The Pixel Farm. Rough geometric proxies were created in a few dozen minutes with a standard polygon editing tool, using six to ten reconstructed features and world-space orientation from PFTrack as construction guides. Since our method bypasses the view-dependent image-plane triangulation of the original ULR algorithm, our proxies need to have unit depth complexity when seen from the viewing region of interest. Statistics for each unstructured lumigraph thus created are summarised in Table I. Views of these scenes are shown in Figures 1 and 6.

5.1 User Study

We designed and ran an experiment to evaluate the results of our framework.

We first wished to measure how much better our view selection technique is to currently available alternatives. As discussed in the related work section, without accurate geometric knowledge of the scene and when cameras can be arbitrarily placed, current alternatives are limited to techniques that ensure a uniform coverage of the geometric proxy by the retained views. We, therefore, picked two test scenes where uniform coverage could easily be enforced for any number of discarded subviews: their initial views were obtained by evenly trucking the camera respectively in front of a building facade (see Figure 1) and in the university library (see Figure 6). In this case, uniform coverage for any number of remaining views can be maintained by picking them evenly from the initial set. Since our technique operates at the subview level, for a given number sv of discarded subviews, we estimated the number v of views to discard by dividing sv by the average number of subviews contained in each view. Once all the subviews contained in the evenly picked v views were discarded, we made up the difference in the following fashion: remaining views were browsed in sequence, removing (or recovering if more subviews than needed had been discarded) a single subview from each. To ensure the even distribution of removed (recovered) subviews, we used a vertex coloring of the geometric proxy and picked subviews of the same color, moving on to the next color when one was exhausted. This heuristic, including the graph coloring, took less than a minute to apply to each test scene. We will call it *regular* in the remainder of the article.

Next we wanted to know how different the results of our framework would be depending on the metric used, PDIFF, SSIM, or RMS.

The goal of the experiment was, therefore, to measure, for each of the four subview discarding strategies (three metrics plus *regular*), at which number of discarded subviews users started to perceive visual degradation compared to the original dataset. We chose a double-random staircase experiment design.

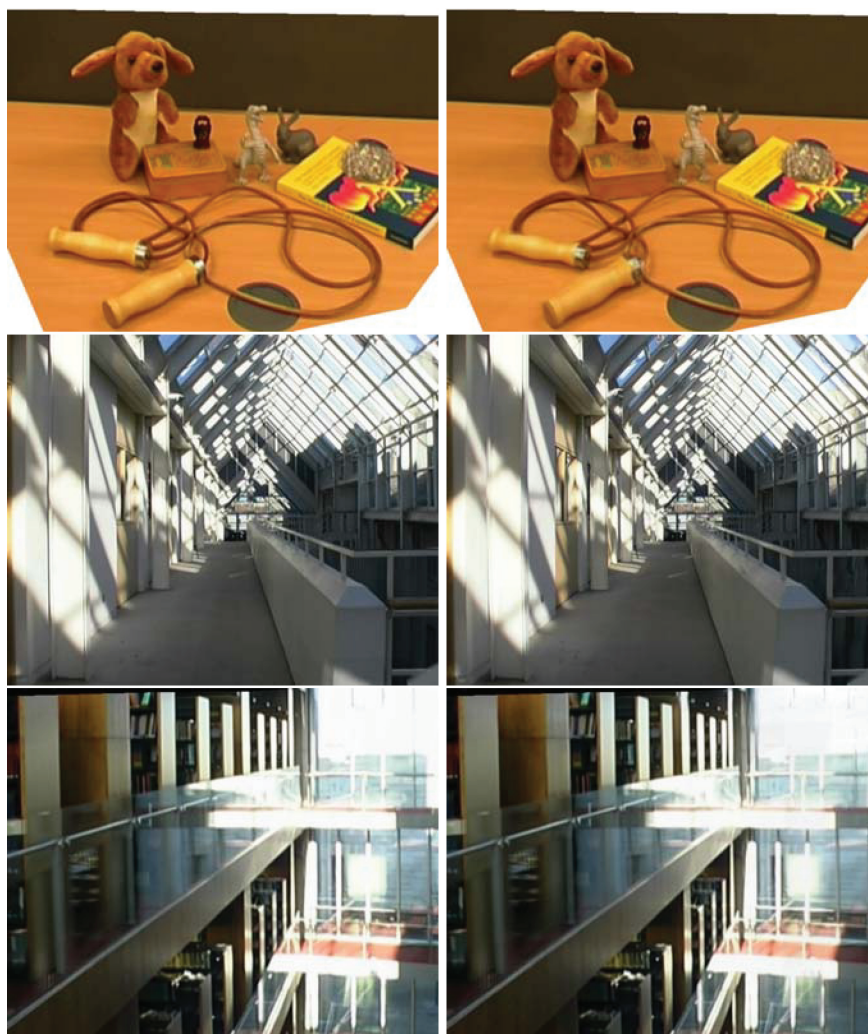


Fig. 6. Novel views of some test scenes, from top to bottom: Objects, Mezzanine and Library. On the left: original dataset. On the right: trimmed dataset.

In a staircase experiment, users are presented with a sequence of stimuli at different levels of intensity. They are asked to perform a two-alternative forced choice at each step. The sequence of stimulus levels is influenced by the behaviour of the user: the level is regularly updated in one direction while the user keeps making the same choice, and direction changes when the current choice contradicts the last one (this is called a reversal). Ascending staircases start with the lowest stimulus level and their initial stimulus update direction is up. It is the reverse for descending staircases. Double-random staircase designs randomly interleave one staircase of each kind.

In our case, the stimuli presented are pairs of short (2 seconds) rendered video sequences of the test scene, one of which uses the original dataset, shown one after the other. The number of subviews discarded before rendering the second video corresponds to the stimulus strength. The forced choice consists of deciding whether the videos are of the same or different quality.

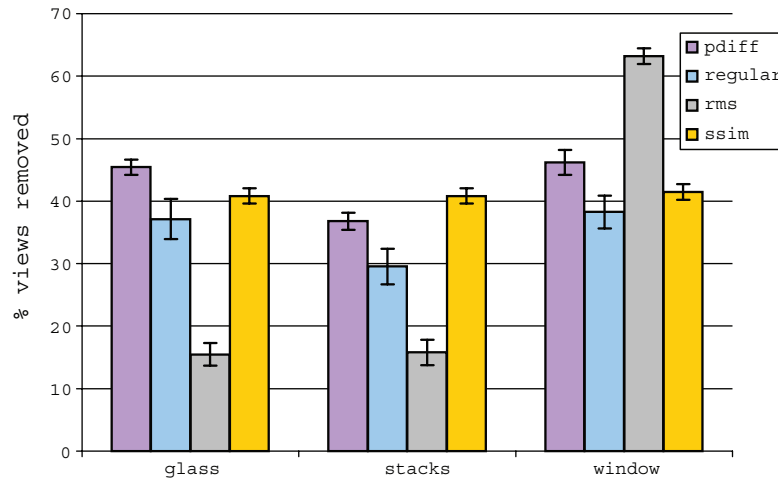


Fig. 7. Proportion of subviews discarded for each strategy for each test subregion at the PSE (standard errors are shown).

Asking participants to compare the quality of two full size PAL videos of short duration is problematic because it is difficult to control where they focus their attention. For this reason, we produced four sets of smaller videos showing two subregions of each test scene. For the *Exterior* scene, the subregions consisted of the windows plus the tree branch (*window*, resolution 184×156), and the door plus the tree trunk (*door*, resolution 172×194). For the *Library* scene, they consisted of a section of the upper glass barrier (*glass*, resolution 176×150), and a section of the upper book stacks (*stacks*, resolution 162×124). The novel camera paths used to produce the videos were created ad-hoc. The same path was used for both subregions of a test scene. The experiment was setup so that the displayed video for each subregion subtended roughly between 4° and 5° . The monitor used had a brightness rating of $250\text{cd}/\text{m}^2$, since it was set to 75% brightness for viewing comfort, we passed a value of $187.5\text{cd}/\text{m}^2$ to the PDIFF metric.

The full experiment thus consisted of 32 staircases: 4 subview discarding strategies \times 4 test subregions \times 2 staircases (one ascending, one descending), all randomly interleaved to counteract learning and expectation effects. Each staircase was limited to 8 reversals. We chose 3,200 subviews as the initial increment (decrement) value of the stimulus level, which was lowered to 1,600 upon the third reversal and 800 upon the sixth. A few pilot runs of the experiments yielded an average duration of over an hour. We judged this too long for the participants to maintain concentration, thus we split the experiments in two sessions of under 40 minutes, asking participants to come back later during the day for the second session (break times varied from 40 minutes to 2 hours). Each session consisted of the staircases corresponding to one subregion of each scene. Which session each participant sat through first was randomised. 16 participants took part within a controlled setup (same computer, display device and viewing conditions). All were from the computer science department (3 staff, 13 students), seven were women, all had normal or corrected to normal eyesight.

By fitting a psychometric function to their responses, we estimated the point of subjective equality (PSE) (i.e., the number of discarded subviews at which participants had an even chance of reporting some visual degradation). Results are summarised in Figure 7. Results for the *door* subregion were discarded because staircases failed to converge in most cases. We attribute that fact to the presence of a slight popping artifact in the video obtained with the original dataset, which could have confused participants.

Table III. Number of retained subviews for each test scene

	Remaining subviews	%
Exterior	11000	59 %
Library	11500	73.6 %
Mezzanine	3000	20.7 %
Objects	10000	48.4 %

We performed a two-factor analysis of variance (ANOVA) with replication on the results. It shows a significant main effect of the view discarding strategy factor ($F(3, 180) = 15.48, p \approx 0$), a significant main effect of the subregion factor ($F(3, 180) = 71.24, p \approx 0$), and a significant interaction between the two factors ($F(6, 180) = 36.48, p \approx 0$). Concerning the subregion factor, it is natural that a given metric yields different numbers of discarded subviews at the PSE on different subregions, as those exhibit varying degrees of view-dependency. The strong significance of the interaction effect means that the relative performances with respect to each other of the four subview discarding strategies depend on the content.

Post-hoc analysis was then performed using a standard Newman-Keuls test for pairwise comparisons among means. Regarding the discarding strategy factor, there were only two cases where two strategies' outcomes were not significantly different. One was between SSIM and regular on the window subregion ($p = 0.057$). This can be attributed to the much higher variance in the PSEs resulting from the regular discarding strategy: participants disagreed more with each other as to when degradations started appearing with this strategy than they did with our framework (using any metric). The other was between SSIM and PDIFF on the *stacks* subregion ($p = 0.052$), providing the exception to the rule that each metric used in our framework yielded statistically different results.

A reading of the chart in Figure 7 shows that both SSIM and PDIFF yield results that are consistently better than those of the regular discarding strategy, with a slight overall advantage for PDIFF. RMS's performance is very erratic: it is by far the worst strategy for both the glass and stacks subregions, yet strongly outperforms the competition for the window subregion. This could be explained by the fact that RMS is a very one dimensional metric compared to PDIFF or SSIM. In particular, its extremely local focus (pixel difference) makes it very sensitive to noise and blur. This would explain why it performs well on the window subregion, which contains complex patterns of intertwined small branches in the forefront, whose inaccurate fit with the geometric proxy yields strong blurring as subviews are discarded.

In Figure 8, we plot the aggregated psychometric function over all participants for each subregion. Those plots let us compare the discarding strategies at different levels of probability that participants will spot visual degradations. The steepness of each curve at its point of inflection reflects how consistent participants were with themselves in reporting when they started noticing degradation: the steeper, the more consistent and the more predictive the psychometric function. In this respect, the regular discarding strategy caused the most confusion in participants, while PDIFF yielded the best results. Interestingly, this means that the lower the desired probability of detection, the better PDIFF compares to SSIM. Thus, in the case of the stacks subregion, where SSIM performed better at the PSE, PDIFF overtakes it for detection probabilities below 37%. In the context of visual content authoring, those are the probability levels that matter.

5.2 Impact on Rendering Performance

The subviews of each unstructured lumigraph were perceptually ordered by our greedy pruning process with the PDIFF metric. Processing times are given in Table II. We then navigated the unstructured lumigraphs while changing the number of retained subviews within the ordered list, allowing us to

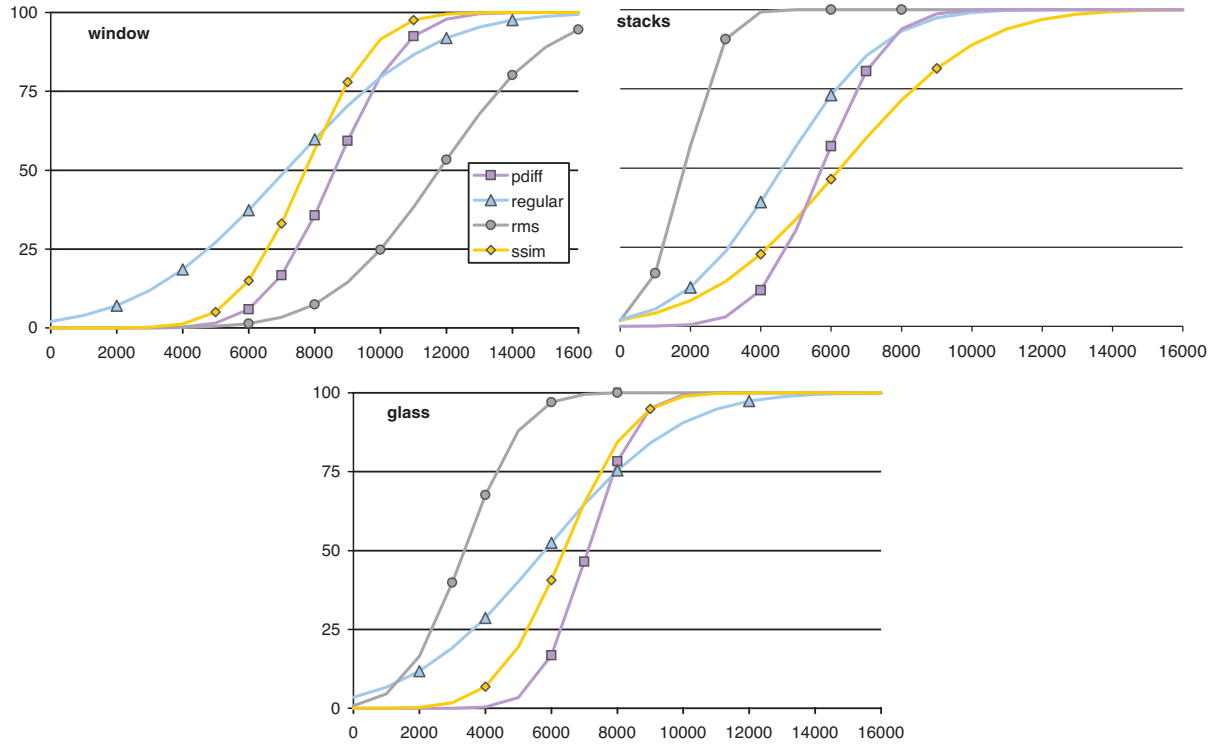


Fig. 8. Psychometric functions aggregated over the 16 participants for each subregion. The Y axis represents the probability of a participant reporting some visual degradation. The X axis shows the number of subviews discarded.

Table IV. Trimming results. “Size” stands for total texture memory footprint. Frames per second are measured on a 3GHz Pentium 4 computer equipped with an ATI Radeon 9800 XT graphics card. The NA rating means that there was no sustainable frame-rate due to texture swapping

	Size (Megabytes)		Frames per second	
	Original	Trimmed	Original	Trimmed
Exterior	164.3	90.9	9.5	11.5
Library	104.5	85.7	10.5	11.5
Mezzanine	78.1	10.5	7	11
Objects	201	29.25	NA	12

determine the lowest number before degradations became noticeable. Retained subview numbers for each test scene are given in Table III.

Resulting texture memory footprints and frame-rates are summarised in Table IV. They vary significantly because of the wide differences between the properties of the test scenes. The mezzanine scene’s geometric proxy is more detailed and fits the actual geometry better. The library contains glass surfaces that introduce strong view-dependence. In the exterior scene, the geometric proxy matches the branches of the tree very poorly. This illustrates the importance of a human operator to make a judgement call on the data/quality compromise—a call that our technique facilitates. Indeed, our use of

Table V. Results expressed in terms of texture memory usage ratio and number of candidate views per vertex ratio

	Texture	Views/vertex
Exterior	55.3 %	55.5 %
Library	82 %	71 %
Mezzanine	13.5 %	22.5 %
Objects	14.5 %	48.4 %



Fig. 9. An input view of the exterior scene after processing: the geometric proxy is drawn in white. Triangles in black belong to vertex rings that contain a triangle that is not fully visible in that view, which makes them unusable in the first place. Triangles in red belong to subviews that were discarded by our technique.

fidelity metrics lets us compare the importance of subviews relative to each other but does not provide an objective assessment of the quality of the dataset at a given point during the pruning process. The high-texture memory savings obtained for the Objects scene is partly explained by the fact that its geometric proxy does not fill the whole view frustum in most views: unused areas are not packed in the texture atlas.

Two interesting quantities can be used to measure the success of our approach. First is the texture memory ratio between the original lumigraph and its trimmed version. Second, the ratio of blendable views per vertex. Indeed, the bottleneck of the ULR technique lies in the computation of the blending weights, as the algorithm has to order each candidate blendable view using a penalty function, before renormalizing the weights over the k best views. The ratio of blendable views per vertex is, therefore, a good hint at the speed-up that any ULR implementation can expect thanks to our technique. The frame-rates quoted in Table IV illustrate this point: as expected, the rendering speed-up correlates with the number of vertices of the geometric proxy. Table V contains both ratios.

Figure 9 illustrates the behaviour of our technique on an input view. In this test scene, the proxy is fitting the ground and the wall but is jutting towards the viewer at the tree's trunk and branches. Our technique correctly discards subviews (in red) where little view dependence is present (i.e., where the proxy is modelling the scene properly and the surface there does not exhibit view dependent phenomena). Comparisons of renderings of the three other scenes before and after discarding subviews are shown in Figure 6.

Table VI. Spearman- ρ correlation of the ranks of the first candidate subview to discard for different choices of touchstone views. (See 5.3 for an explanation of cases)

	Case 1	Case 2	Case 3	Case 4	Case 5
Case 6	0.25	0.74	0.2	0.31	0.18
Case 5	0.74	0.26	0.32	0.19	
Case 4	0.24	0.75	0.17		
Case 3	0.76	0.25			
Case 2	0.33				

5.3 Choice of Touchstone Views

As mentioned in Section 3.4, we chose to use every second primary view (PV) as touchstone view (TV) for the purpose of our user study. In order to measure how much influence the initial choice of touchstone views has on our approach, we gathered data obtained with six different touchstone views configurations for the exterior scene. The configuration cases were defined as follows: Case 1, every other PV is a TV, starting with the first PV; Case 2, as Case 1, but starting with the second PV; Cases 3, 4, 5 and 6, every fourth PV is a TV, starting with the first, second, third, and fourth PV, respectively. For each case, we recorded the rank that each of the 18,638 candidate subviews would have at the first removal iteration (i.e., ordered according to increasing perceptual degradation as measured by PDIFF). Table VI shows the pairwise correlation of ranks obtained using Spearman's rho.

The roughly medium correlation coefficients indicate that the choice of touchstone views has a lot of influence on the set of retained subviews. Predictably, configuration cases that share touchstone views (1 versus 3 and 5; 2 versus 4 and 6) have higher correlation. What is interesting is that those correlations are consistently high (>0.74), despite the fact that cases 3, 4, 5, and 6 contain half as many touchstone views as cases 1 and 2. This suggests that there is some leeway in terms of the number of touchstone views to use. The low correlation coefficients for cases that have no touchstone views in common do not imply that different configurations will result in widely varying rendering quality at equal numbers of subviews discarded, but merely that a bias is introduced with respect to which view is considered redundant in a pair of views containing the same visual information.

According to our time complexity analysis (see Section 4.2.2), using cases 3 or 5 over case 1 would have sped up the subview selection process by a factor of 2. The correlations obtained here suggest that the outcomes would have been somewhat similar.

5.4 Adjusting ULR Parameters

We have applied our framework to optimise the parameters of the ULR to each of our test scenes. Figure 10 shows plots of the PDIFF metric as a function of k and β for the exterior scene. Results show that in this particular case, the penalty_{res} term is best ignored altogether. 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. Optimal values of k vary for different regions of the scene, depending on the amount of view dependence it exhibits.

Regions with little view dependence, such as the ground in this scene, are little affected by low k values. The fact that high k values do not degrade the perceptual grade indicates that the camera is well calibrated. Otherwise, calibration residuals would introduce some blur as more views get blended, with no benefit of added visual information.

Evaluating the metric over all the faces of each test scene for the $13 \times 7 = 91$ parameter combinations took an impractical amount of time (days). However, limiting the evaluation to a selected fraction of the faces provided a good estimate for an adequate choice of parameters.

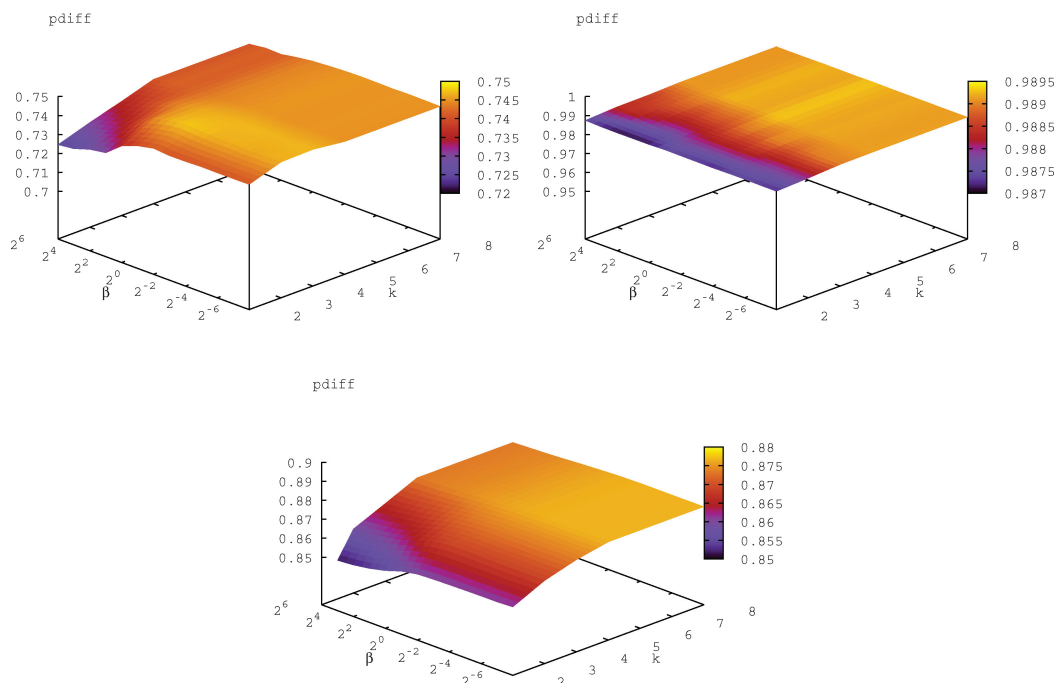


Fig. 10. PDIFF metric result plotted against k and β for the test scene shown in Figures 1 and 2. Top left: metric for a single face corresponding to the tree branch. Top right: metric for a single face corresponding to the ground. Bottom: metric aggregated over all the faces of the geometric proxy.

6. DISCUSSION AND FUTURE WORK

Our method has shown great potential on our test scenes, resulting in significantly better results than regular discarding when using either PDIFF or SSIM, and this consistently over subregions of varying nature. Once it has ordered the initial subviews by increasing perceptual importance, an artist can quickly explore renderings of the scene using varying numbers of remaining subviews until he is satisfied with the compromise between visual quality and resource consumption (memory as well as rendering time). We have shown that our framework also makes it possible to automatically adjust the ULR parameters for a given scene. This is a valuable step in the process of authoring unstructured lumigraphs, be it preceded or not by a selection of subviews.

When it comes to comparing the performance of different metrics within our framework, Yee and Newman's [2004] PerceptualDiff appears to win. As acknowledged by Wang et al. [2004] in their citation of the research conducted within the Modelfest framework and by the Video Quality Expert Group, debate is strong in the field of image fidelity metrics. It would perhaps be worthwhile to investigate ways to predict which metric is most appropriate depending on the type of content, both generally and locally.

As mentioned in the introduction, the closest techniques with which to compare our work deal with interactive rendering from compressed light-fields. Be they based on vector quantization, DCT or wavelets, they typically achieve higher memory savings than what we obtain, thanks to their much finer granularity. The most dramatic results are, however, obtained with stronger requirements about the geometric information than we make. From the point of view of rendering performance, typical numbers quoted in the literature hover around 8 frames per second [Xin and Gray 2003]. Since we chose to implement

our framework in Haskell for the prototyping ease it provides, the performance figures we obtain are probably not representative of a heavily optimised ULR implementation. However, at roughly 7 to 15 fps (measured on hardware similar to that used in the literature, see Figure 4), our approach already appears to be competitive. There are less easily quantifiable factors to consider in the comparison. One is the flexibility of the ULR representation, which results in better authoring convenience: the artist has more freedom to choose a geometry *vs.* sampling compromise, and he is not bound by sampling regularity, which light field techniques tend to enforce rigidly. Another is its higher suitability for inclusion in a three-dimensional engine. A formal comparison of the techniques would be very worthwhile, but it is a challenge because of the slight variations in requirements and features that make the choice of test scenarios that are suitable across the board difficult.

In its current implementation, the main drawback of our method is its computation time, considering that the number of input views can grow considerably with the area of the viewing region that the artist wishes to cover. To combat this, a statistical approach could be used at each removal step to avoid considering all remaining subviews, and only a subset of them. Our preliminary study of Section 5.3 suggests that it is possible to considerably reduce computation time by playing with the number of touchstone views considered when applying the perceptual metric. This is an area of future experimentation as the choice of touchstone views, or possibly their weighting by importance, would be a natural way for an artist to tune view selection to areas of particular importance in the viewing region from which the scene is intended to be seen.

REFERENCES

- BUEHLER, C., BOSSE, M., McMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *Computer Graphics (Siggraph 01)*. 425–432.
- COOMBE, G., HANTAK, C., LASTRA, A., AND GRZESZCZUK, R. 2005. Online construction of surface light fields. In *Rendering Techniques*. 83–90.
- DALY, S. 1993. The visible differences predictor: an algorithm for the assessment of image fidelity. In *Digital Images and Human Vision*. 179–206.
- DEBEVEC, P., BORSHUKOV, G., AND YU, Y. 1998. Efficient view-dependent image-based rendering with projective texture mapping. In *Rendering Techniques 98 Proc. Eurographics Workshop on Rendering*.
- DUMONT, R., PELLACINI, F., AND FERWERDA, J. A. 2003. Perceptually-driven decision theory for interactive realistic rendering. *ACM Transactions on Graphics* 22, 2, 152–181.
- FLEISHMAN, S., COHEN-OR, D., AND LISCHINSKI, D. 2000. Automatic camera placement for image-based modeling. *Computer Graphics Forum* 19, 2, 101–110.
- GORTLER, S., GRZESZCZUK, R., SWELISKI, R., AND COHEN, M. 1996. The lumigraph. In *Computer Graphics Proceedings, Annual Conferences Series, Siggraph'96*. 43–54.
- HALE, J. G. 1998. *Texture Re-Mapping for Decimated Polygonal Meshes*. Edinburgh University.
- HEIGL, B., KOCH, R., POLLEFEYS, M., DENZLER, J., AND GOOL, L. J. V. 1999. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM-Symposium*. 94–101.
- HLAVAC, V., LEONARDIS, A., AND WERNER, T. 1996. Automatic selection of reference views for image-based scene representations. In *Proc. ECCV*. 526–535.
- LUBIN, J. 1995. A visual discrimination model for imaging system design and development. In *Vision Models for Target Detection and Recognition*, E. Peli, Ed. World Scientific, 245–283.
- MANTIUK, R., DALY, S., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2005. Predicting visible differences in high dynamic range images—model and its calibration. In *Human Vision and Electronic Imaging X, IS&T/SPIE's 17th Annual Symposium on Electronic Imaging (2005)*, B. E. Rogowitz, T. N. Pappas, and S. J. Daly, Eds. Vol. 5666. 204–214.
- MYSZKOWSKI, K., ROKITA, P., AND TAWARA, T. 1999. Perceptually-informed accelerated rendering of high quality walkthrough sequences. In *Proceedings of the Tenth Eurographics Workshop on Rendering*. 5–18.
- RAMASUBRAMANIAN, M., PATTANAIK, S. N., AND GREENBERG, D. P. 1999. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of ACM Siggraph 1999*. ACM Press/ACM Siggraph, 73–82.

- SCHIRMACHER, H., HEIDRICH, W., AND SEIDEL, H.-P. 1999. Adaptive acquisition of lumigraphs from synthetic scenes. In *Computer Graphics Forum (Eurographics '99)*, P. Brunet and R. Scopigno, Eds. Vol. 18(3). The Eurographics Association and Blackwell Publishers, 151–160.
- STOKES, W. A., FERWERDA, J. A., WALTER, B., AND GREENBERG, D. P. 2004. Perceptual illumination components: A new approach to efficient, high quality global illumination rendering. In *ACM Siggraph Conference Proceedings*. ACM Press.
- VAZQUEZ, P., FEIXAS, M., SBERT, M., AND HEIDRICH, W. 2001. Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference (VMV01)*. IOS Press, Amsterdam, 273–280.
- VQEG. 2000. Final report from the video quality experts group on the validation of objective models of video quality assessment. Tech. rep., VQEG.
- WALTER, B., GREENBERG, D. P., AND PATTANAIK, S. N. 2002. Using perceptual texture masking for efficient image synthesis. In *Eurographics Computer Graphics Forum*. Vol. 21.
- WANG, Z., BOVIK, A. C., AND LU, L. 2002. Why is image quality assessment so difficult? In *Proceedings of the IEEE International Conference on Acoustics, Speech, & Signal Processing*. Vol. 4. 3313–3316.
- WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing*. Vol. 13, no. 4.
- WILLIAMS, N., LUEBKE, D., COHEN, J., KELLEY, M., AND SCHUBERT, B. 2003. Perceptually guided simplification of lit, textured meshes. In *Proceedings of the 2003 ACM Siggraph Symposium on Interactive 3D Graphics*.
- XIN, T. AND GRAY, R. 2003. Interactive rendering from compressed light fields. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 11, 1080–1091.
- YEE, H., PATTANAIK, S., AND GREENBERG, D. P. 2001. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Trans. Graph.* 20, 1, 39–65.
- YEE, Y. H. AND NEWMAN, A. 2004. A perceptual metric for production testing. In *Siggraph '04: ACM Siggraph 2004 Sketches*. ACM Press, New York, 121.

Received March 22, 2008; revised August 18, 2008; accepted August 27, 2008