



Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Perceptually Guided Authoring of Image Based Representations

by

Yann Morvan, D.E.A.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

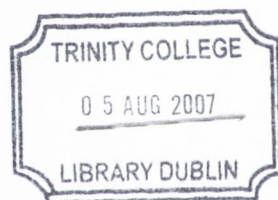
of the requirements

for the Degree of

Doctor of Philosophy

University of Dublin, Trinity College

April 2007



THOS'S
8172

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

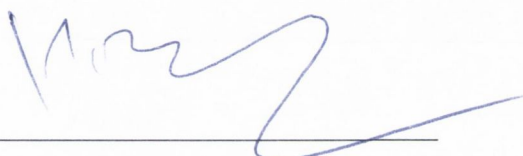


Yann Morvan

April 27, 2007

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.



Yann Morvan

April 27, 2007

Acknowledgments

My thanks go first to Carol O'Sullivan for all her help, advice and encouragement. Then to Kadi Bouatouch and Gerard Lacey, who were selfless enough to plod through yet another typo ridden PhD manuscript. And to all the colleagues in the lab, in no particular order: Richard, Tarun, Thanh, Al, Andrew, Zozo John, Lepton John, Cesar, Keith, Chris, Sarah, Rachel, Steven, Simon, and Clodagh.

YANN MORVAN

University of Dublin, Trinity College
April 2007

Perceptually Guided Authoring of Image Based Representations

Publication No. _____

Yann Morvan, Ph.D.

University of Dublin, Trinity College, 2007

Supervisor: Carol O'Sullivan

Perceptually Guided Authoring of Image Based Representations

Yann Morvan

Supervised by Carol O’Sullivan

Abstract The creation of digital visual content has become a major activity in a broad range of industries. A huge quantity of labour is invested in related tasks, from modeling to de-noising 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 informed graphics, albeit so far predominantly explored to achieve machine-hours savings, is another area of great potential.

Research in image-based rendering 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. Less effort has been devoted to facilitating the authoring of image-based representations, a task that is non-trivial because of the great amounts of data involved and the engineering issues that come with accurate data capture.

In this thesis, we present techniques to assist the authoring of image-based representations using a perceptual approach. A framework to ease the selection of input views for unstructured image based rendering is developed. We apply it to unstructured lumigraphs and refine it to perform selection at a sub-view level. User studies are designed and performed to evaluate the results of the framework. A texture re-packing scheme is proposed to capitalise on the savings incurred from discarding sub-views while maintaining the spatial coherence of remaining sub-views. We introduce a method to automatically adjust the user parameters of image based rendering algorithms and apply it to the unstructured lumigraph rendering algorithm. To facilitate the edition of useful geometric information, we design and implement an image based modelling system making use of human 3D perception. Finally,

we propose a solution to capturing problematic scenes by tracking a camera with an optical motion capture system.

Contents

Acknowledgments	iv
Abstract	v
List of Tables	xi
List of Figures	xii
Chapter 1 Introduction	1
1.1 Motivation	1
1.1.1 Image-based Representations	2
1.1.2 Perceptually informed graphics	4
1.2 Objectives and Scope	5
1.3 Organisation of the Thesis:	6
1.4 Contributions of the Thesis	8
Chapter 2 Background	9
2.1 Image Based Rendering	9
2.1.1 Basic notions	10
2.1.2 Techniques	14
2.2 Authoring of Image Based Representations	22
2.2.1 Capture	23
2.2.2 Geometric information	24
2.2.3 View Selection	26
2.2.4 Compression	27
2.2.5 Summary	28

2.3	Visual Fidelity Metrics	29
2.3.1	Physically based	29
2.3.2	The human visual system	31
2.3.3	Perceptual metrics	35
2.3.4	Structural Similarity Index	37
2.4	Perceptually adaptive rendering	39
2.4.1	Global illumination for static images	40
2.4.2	Interactive rendering of global illumination solutions	42
Chapter 3 Perceptual importance of indirect lighting as a function of luminosity.		
	A psychophysical experiment.	45
3.1	Controlled test environment	46
3.1.1	Physical framework	46
3.1.2	Modeling	47
3.2	Experimental design	49
3.3	Results	50
3.3.1	Pre-analysis	50
3.3.2	Correlations	51
3.4	Conclusions	55
3.4.1	Regarding the experiment	55
3.4.2	Influence on subsequent research	56
Chapter 4 Scene acquisition		58
4.1	Camera tracking	59
4.1.1	Scenes with suitable features	59
4.1.2	Problematic scenes	60
4.2	Geometric proxy editing	72
4.2.1	Stereoscopic 3D positioning system	73
4.2.2	Stereo pairs acquisition	77
4.2.3	Discussion	78
Chapter 5 View selection framework		82
5.1	Design	82
5.1.1	Context: Unstructured Lumigraph Rendering	84

5.1.2	Perceptual measure of view utility	84
5.1.3	Selecting Whole Views	86
5.1.4	Selecting at a Sub-View Level	89
5.2	Implementation	92
5.2.1	Organization	93
5.2.2	Unstructured Lumigraph Renderer	96
5.2.3	Implementation of the Structural Similarity Index	100
5.3	Texture Atlas Generation from Selected Sub-Views	103
5.3.1	Triangle Packing	103
5.3.2	Maintaining Perspective Correct Projective Texture Mapping	104
Chapter 6 Results and evaluation		106
6.1	Whole view selection	106
6.1.1	Comparisons	107
6.1.2	Test scenes	108
6.1.3	Experiment	109
6.1.4	Conclusions	113
6.2	Sub-view selection	115
6.2.1	User study	116
6.2.2	Rendering performance	120
6.2.3	Discussion	123
6.3	Parameter adjusting	126
6.3.1	Case of the unstructured lumigraph	126
6.3.2	Finer tuning	127
6.3.3	Results and discussion	128
Chapter 7 Applications, conclusions and future work		130
7.1	Practical applications of the contributions	130
7.2	Future work	132
Appendices		134
Bibliography		143

List of Tables

3.1	ANOVA results.	51
3.2	Overall correlations.	53
3.3	Correlations per region.	54
4.1	Scenes	81
6.1	Test scenes statistics (note that we do not count sub-views that are not blend- able for lack of visibility of the whole triangle ring)	116
6.2	Approximate time taken to order all sub-views by increasing perceptual im- portance (hours).	122
6.3	Number of retained sub-views for each test scene.	123
6.4	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 framerate due to texture swapping.	123
6.5	Results expressed in terms of texture memory usage ratio and number of candidate views per vertex ratio.	123
1	Test scenes statistics (note that we do not count sub-views that are not blend- able for lack of visibility of the whole triangle ring)	134

List of Figures

1.1	The Getaway, on the upcoming PlayStation 3 ©2005 Sony Computer Entertainment Europe	2
1.2	Davy Jones in Pirates of the Carribean 2, ©2006 Disney	3
1.3	Zak McKracken and the Alien Mindbenders, on Atari ST ©1988 LucasFilm Games	4
2.1	Leonardo’s thought experiment.	11
2.2	Perspective projection in the pinhole camera model.	12
2.3	Father Ted: “These are small. But the ones out there are far away.”	13
2.4	The trade-off between the geometric information available and the number of views necessary in image based rendering	16
2.5	A light slab as used in Light Field and Lumigraph rendering.	17
2.6	Concentric mosaics parameterization. The length of the beam being fixed, the pose of each camera is only parameterized by θ . If (u, v) are the coordinates of a pixel in the corresponding photograph, each ray of the concentric mosaic is identified by the triplet (θ, u, v)	18
2.7	The Stanford multi-camera array (128 cameras) [VWJL04]	24
2.8	Behaviour of root mean square error on two kinds of image alteration. . . .	30
2.9	Schematic diagram of the human eye. From Wikipedia.	32
2.10	Benham’s disc.	33
2.11	Two representations of a checker board.	34
2.12	Simultaneous contrast. The central patch appears increasingly bright from left to right even though its luminosity is constant	35
2.13	The VDP, diagram.	37

2.14	A pair of images and the corresponding VDP using the contrast sensitivity function proposed by Daly. (<i>Images courtesy of Karol Myszkowski [VMKK00]</i>)	44
3.1	Experiment set-up, diagram.	47
3.2	Experiment set-up, photograph.	48
3.3	Left: all objects (ALL) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the bottom shelf (B). .	50
3.4	Left: objects of the bottom and middle shelves (MB) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the middle shelf (M).	51
3.5	Left: no objects (NO) rendered with less resources allocated to indirect illumination, Right: photograph (P) of the real scene.	52
3.6	Left: objects of the top and bottom shelves (TB) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the top shelf (T).	53
3.7	Objects of the top and middle shelves (TM) rendered with less resources allocated to indirect illumination (one bounce).	54
4.1	Error volume of the Vicon as a function of distance.	63
4.2	Tilt error.	65
4.3	Orientation error upper bound.	66
4.4	Error due to translation.	66
4.5	The camera rig.	68
4.6	Calibration object and recovered features.	71
4.7	The GeoWall projectors and their filters on the adjustable stacker.	74
4.8	Trackable pincers and stereo glasses.	75
4.9	System setup. Two of the Vicon cameras can be seen in the background. The bright glares are the reflective markers.	76
4.10	Top view of a scene being modeled with one binocular viewpoint. Position 1 of the 3D cursor coincides with that of a feature and thus gets projected to the same point as that feature in both viewports. Positions 2 and 3 differ from that of the feature: even though their projections coincide with that of the feature in the left viewport, they do not in the right viewport. Thus the user correctly perceives them as misaligned.	80

5.1	Aggregate fidelity metric over a set of n input views, called touchstone views. Each touchstone view is compared to a rendering from the same viewpoint using an image fidelity metric. The fidelity scores are summed to yield the aggregate score. (The red triangles correspond to the projection of a face of the geometric proxy in each view.)	83
5.2	At a given removal step and touchstone view $\mathcal{TV}$, for each vertex, the $(k + 1)^{th}$ best reference view is needed, otherwise we would be left with only $k - 1$ reference views to blend at that vertex when rendering the face after having removed a candidate view (marked with a $?$). Note that $\mathcal{TV}$ is excluded from the set of k best views.	89
5.3	The depth of an image plane grid point has to match that of the closest surface.	90
5.4	Erroneous triangulation due to an overlap of the geometric proxy in the desired view.	91
5.5	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.	92
5.6	Four sub-views, corresponding to the same two geometric proxy vertices seen in two different input views.	93
5.7	Application organization diagram.	94
5.8	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.	98
5.9	The six fragment shader passes used to compute SSIM indices.	102
5.10	Triangle packing.	104
5.11	A packed texture.	105
6.1	General views of the test scenes and their geometric proxy.	110
6.2	Several input views for the pub scene, such as might be captured from the street with a video camera.	111

6.3	Left: conventional rendering. Middle: unstructured lumigraph rendering using 8.5% of initial input views selected by hierarchical clustering. Right: ULR rendering using the same number of views selected by our perceptual method. Note how most of the depth of the celtic motif is lost in the middle image. The loss is much more conspicuous when the viewer moves around the scene as the parallax effect is damped.	112
6.4	Coverage of the celtic logo's geometric proxy by the remaining input views (brighter shades indicate better coverage). Columns correspond to the number of remaining views. Top row: perceptual approach, bottom row: regular.	113
6.5	Photograph showing the setup of the first experiment.	114
6.6	Percentage of times that participants preferred the perceptual view selection to the regular view selection. On the left, by scene. On the right, by number of input views removed.	115
6.7	Photograph showing the setup of the second experiment.	119
6.8	Proportion of sub-views discarded for each strategy for each test sub-region at the PSE (standard errors are shown).	121
6.9	Psychometric functions aggregated over the 16 participants for each sub-region. The Y axis represents the probability of a participant reporting some visual degradation. The X axis shows the number of sub-views discarded.	121
6.10	Exterior scene. Left: frame of digital video footage. Right: a novel view rendered after trimming of the unstructured lumigraph.	122
6.11	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 sub-views that were discarded by our technique.	124
6.12	Aggregate metric against k and β for the test scene shown in Figure 5.	128
7.1	Types of suitable image-based representations as a function of distance to the observer.	131
2	Geometric proxy seen from two angles and several input views for the Mezzanine scene.	135

3	Geometric proxy seen from two angles and several input views for the Objects scene.	136
4	Geometric proxy seen from two angles and several input views for the Library scene.	137
5	Geometric proxy seen from two angles and several input views for the Exterior scene.	138
6	Novel views of the Mezzanine scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 11508 and 13008 sub-views out of 14508 according to Tables 6.1 and 6.3 of Section 6.2	139
7	Novel views of the Objects scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 10674 and 15674 sub-views out of 20674 according to Tables 6.1 and 6.3 of Section 6.2	140
8	Novel views of the Library scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 4125 and 8125 sub-views out of 15625 according to Tables 6.1 and 6.3 of Section 6.2	141
9	Novel views of the Exterior scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 7638 and 11638 sub-views out of 18638 according to Tables 6.1 and 6.3 of Section 6.2	142

Chapter 1

Introduction

1.1 Motivation

Because of the great creative freedom that it allows, computer assisted visual content creation has become a staple feature of all media industries, from the movies, through broadcasting and publishing, to video games and advertising. Over the past decade, the public has become accustomed to visual content of irreproachable quality, regardless of its origin. Video games with a realistic bent now depict complex environments in a quasi-photographic manner (Figure 1.1). In film, virtual imaginary creatures (Figure 1.2) are now held to the same standard of “realism” as flesh and bones actors. It is now impossible for production companies aiming at a mass audience to settle for quality standards that would have been acceptable some years before (Figure 1.3). Spectators or players would be deterred by the negative impact on the feeling of immersion.

Recent advances in rendering techniques and algorithms, as well as the ever increasing computing power of machines, have made it possible for computer graphics to match the demand in quality at reasonable capital costs. The challenge now lies in the very labour intensive process of authoring the models for the machines to render. Computer artists constitute an increasing proportion of the payroll of major studios, and independent houses that try to compete on the same ground are known not to count hours. The list of tasks is long and includes: creating 3D models of characters, props and environments, designing the corresponding textures, writing circumstance specific shaders, capturing light environments, capturing existing object geometry and de-noising acquired data. Furthermore, we have not



Figure 1.1: The Getaway, on the upcoming PlayStation 3 ©2005 Sony Computer Entertainment Europe

even begun to discuss animation. Given the increasing trend in project aspirations, from feature films where scarcely anything is real, to massively multiplayer online games in which players expect to have whole universes to explore, the challenge of labour efficient visual content creation is unlikely to abate.

All of the tasks mentioned above have tedious and repetitive aspects and artists would benefit greatly from tools designed to alleviate them. With this in mind, two lines of research appear to have a lot of potential. They are image-based methods and perceptually informed graphics.

1.1.1 Image-based Representations

The principle behind image-based representations is to use photographs of the real-world to capture and re-use visual attributes to compose novel views, rather than using physical models and light simulation to compute them. Their main attraction in the context of authoring is that they have the potential to help alleviate, or even bypass altogether, both the 3D mesh modeling step and the texturing step. Some forms of image-based representations are already widely used in the various industries that employ computer graphics. Examples in-



Figure 1.2: Davy Jones in Pirates of the Carribean 2, ©2006 Disney

clude billboard techniques, environment mapping, image-based lighting, or even texture mapping itself.

More ambitious techniques tend to be shunned by the industry because they are not straightforward to author for two reasons. The first is that they require accurate knowledge of the camera parameters corresponding to each input photograph. This obstacle is rapidly becoming less cumbersome with the recent advent and increasing affordability of robust camera tracking techniques using structure from motion algorithms. The second obstacle lies in the huge memory requirements of image-based representations. Those requirements can be lowered with the provision of geometric information, and compression techniques have been proposed. Nonetheless, given a particular scene to represent, it is still very much an expert's task to choose the proper technique, find the appropriate level of geometric detail to use, and adjust the aggressiveness of the compression, all in order to minimise memory consumption while maintaining visual quality.

Providing artists with a tool allowing them to overcome the second obstacle would go a long way in making their authoring task easier by giving them access to the full convenience and power of image-based representations; a field that is still in rapid expansion, *cf.* the current vitality of computational photography [RT06].



Figure 1.3: Zak McKracken and the Alien Mindbenders, on Atari ST ©1988 LucasFilm Games

1.1.2 Perceptually informed graphics

The basic idea behind perceptually informed graphics is to try to avoid rendering, and therefore modelling, visual details that a human observer would not perceive given the viewing conditions. This idea is constantly used in a *ad hoc* way by computer graphics practitioners when they make design choices such as when to stop modelling with triangles and to start modelling with displacement mapping, what texture resolution to use, how many elements to subdivide an area light source into, etc. Each of those many choices adds one more difficulty to the authoring process and, in the absence of tools to guide decision making, solving them is mostly a matter of trial and error and hard-earned experience.

Most research that has been conducted on leveraging knowledge of the human visual system for computer graphics as focused on optimizing rendering algorithms, *i.e.* saving machine hours. An example is 3D model simplification techniques: they assume the availability of a finely detailed model and determine how much simplification the rendering algorithm can get away with without the observer noticing.

1.2 Objectives and Scope

The main purpose of this thesis is to help computer artists to author image-based representations. To this end, we propose the use of perceptual approaches to guide the artist's decision making. Image-based approaches are wide and varied, ranging from depth of field microscopy to arbitrary view-point video. The scope of our work is limited to the representation of **static** scenes under **constant illumination**.

A convenient way to express our objectives in more detail is perhaps to place ourselves in an imaginary real-world authoring scenario. Let us consider the case of an artist in charge of designing city blocks for, say, a skate-boarding simulation video game. Here is the sequence of steps that he could go through:

1. **Game design choices:** In this first step of the pipeline, the artist familiarises himself with the features of the game that are relevant to his task. For instance, the fact that the project is a skate-boarding game places constraints on the player character's viewpoint: it is roughly at standing eye level and it is limited to the areas that the game designers have made skateable within the city maps.
2. **Representation choice:** Here, the artist collaborates with the game programmers to decide on a suitable representation for the city blocks. From the map design, it is possible to anticipate an interval of distances from which each block will be seen by the player. Those distance intervals can guide the choice of representation, but that choice is constrained by software issues. Programmers could be reluctant to implement rendering techniques that are orthogonal to what they use for the rest of the game, most likely a traditional 3D engine. At this point, programmers will also give the artist a budget in terms of texture memory and polygon count. *Is there a representation that offers enough flexibility to be suitable in all cases?*
3. **Capture of relevant building textures:** The artist decides which buildings will be created from existing architecture. He then sets out to obtain collections of photographs of them with corresponding camera parameters. *What is the most convenient way to obtain this data?*
4. **Modelling of relevant buildings:** Using the registered photographs as a guide, the artist creates geometric models of the buildings. He can adjust the maximum level

of detail for each depending on the distance interval from where they will be seen and the polygon budget he was given. *Is it possible to automatically give the artist an idea of a sufficient level of detail? Conversely, is it feasible to let the artist model at his convenience and then use the photographs to compensate for missing geometric detail? If so, can we automatically determine how many and which photographs should be used?*

5. **City layout and refining the end result:** The artist now has a collection of buildings, some of which are represented using an image-based technique. *To what extent can the representations of the image-based buildings be compressed without sacrificing visual quality ?* He now arranges the buildings to design city blocks according to the city maps. *Is it possible to use the city layout to automatically leverage occlusions in order to further compress the image-based buildings? Will the compression affect rendering performance?*

Although specific to a video game setting, this thought experiment is easily transposable to other contexts, such as the creation of a virtual set for film-making, or the authoring of software for the navigation of heritage sites.

1.3 Organisation of the Thesis:

Chapter 2 Background: This chapter starts with a survey of the computer graphics topic of image-based rendering. After presenting the main techniques and the data representations that they require, the body of work aimed at facilitating the authoring of such representations is reviewed, leading to the conclusion that unstructured methods are the most convenient. Some background is then provided on visual fidelity metrics, of which this work makes use. Finally, an overview of perceptually guided rendering is given.

Chapter 3 Perceptual importance of indirect lighting as a function of luminosity. A psychophysical experiment: The initial impetus of our research was in the direction of perceptually guided rendering. In this chapter, we explain how the process of conducting a psychophysical experiment for application in global illumination algorithms, and its outcome, gave us insights that motivated us to further refine and direct our line of research.

Chapter 4 Scene acquisition: In this chapter we provide some details on scene capture solution that we adopted, which consists of using commercially available match-moving software based on structure from motion algorithms to track a hand-held video camera. Like all camera tracking techniques that operate on the captured images of the tracked camera itself, the method fails when the scene lacks strong features. We describe how we worked around this difficulty. Finally, we present an image-based modelling system aimed at facilitating the authoring of a geometric proxy.

Chapter 5 View selection framework: This chapter presents the perceptual framework that we propose to facilitate the selection of a set of input views that minimises visual redundancy in the context of unstructured image-based rendering. We discuss implementation issues in the case of unstructured lumigraph rendering and explain how they led us to modify the original algorithm. Our method to capitalise on the discarding of input views is then presented.

Chapter 6 Results and evaluation: In the course of our work, a number of datasets were produced. Some were generated from renderings of synthetic scenes, while others were captured. They were used as test data for user studies that we conducted to evaluate our view selection techniques. This chapter starts by describing these user studies and analysing the results they yielded. We then provide an analysis of the texture memory savings and rendering speed-ups achieved by re-packing remaining sub-views. Finally, we explain how the principle behind our view selection approach can be applied to automatically tune the user parameters of image-based rendering techniques and provide design details in the case of unstructured lumigraphs.

Chapter 7 Applications, conclusions and future work: In these closing remarks, the ideas and contributions of the thesis are summarized with applications in mind. Finally, suggestions for future research are provided.

Appendix: The appendix contains figures that depict the various test scenes used in our evaluation of the view selection framework, as well as renderings of novel views obtained after discarding input views through our method.

1.4 Contributions of the Thesis

- We introduce a new framework to ease the selection of views for unstructured image-based rendering.
- We apply our framework to unstructured lumigraph rendering and refine it to perform selection at a sub-view level.
- The view selection framework is validated in studies analysing user responses to its visual outcome.
- We adapt and implement a texture re-packing scheme to capitalise on the removal of sub-views, while maintaining the spatial coherence of remaining sub-views. An analysis of the resulting rendering performance is conducted.
- We introduce a novel method to automatically adjust the user parameters of the unstructured lumigraph rendering algorithm.
- We create a novel image-based modelling system making use of human 3D perception.
- Problematic scenes are captured in an original way by tracking a camera with an optical motion capture system.

Chapter 2

Background

This chapter starts with a survey of the computer graphics topic of image based rendering. After presenting the main techniques and the data representations that they require, the body of work aimed at facilitating the authoring of such representations is reviewed. Some background is then provided on visual fidelity metrics, of which this work makes use. Finally, an overview of perceptually guided rendering is given.

2.1 Image Based Rendering

In computer graphics, rendering is the process of generating an image from an electronic representation of a scene. In most cases, the process is analogous to image formation on a camera's film plate or the eye's retina: a projection model, along with viewing parameters, is used to determine which virtual rays of light contribute to the colour of each pixel.

The classical approach to rendering in three dimensions is to use a description of a scene in terms of the geometry of its objects, their surface properties and the characteristics of the sources of light present. With this representation, the rendering problem consists of simulating the physical phenomenon of light energy transfer.

In image based rendering, reference images, instead of geometry, surface properties and lights, constitute the primary scene representation. The set of reference images constitute a measurement of the scene's *light field*, from which the properties of the light rays relevant to the desired image's formation can be extracted.

We first explain the notion of light field, then describe the pinhole camera model, the

most common way of constructing images that constitute a sampling of the light field. Then, we provide a survey of different image based rendering techniques.

2.1.1 Basic notions

The light field and the plenoptic function

The term light field originates in A. Gershun's description of the radiometric properties of light in space [Ger39]. The *plenoptic function* was introduced by Adelson and Bergen [AB91]. Both phrases denote the same concept of "all that there is to be seen", as captured by Leonardo da Vinci in his notebooks, quoted by Adelson and Bergen as translated by J. Richter [Ric70]:

"I say that if the front of a building or any open piazza or field which is illuminated by the sun has a dwelling opposite to it, and if, in the front which does not face that sun, you make a small round hole, all the illuminated objects will project their images through that hole and be visible inside the dwelling on the opposite wall which may be made white; and there, in fact, they will be upside down, and if you make similar openings in several places in the same wall you will have the same result from each. Hence the images of the illuminated objects are all everywhere on this wall and all in each minutest part of it."

Leonardo's images on the "opposite wall" are formed by the projection of each ray of the light pencil going through each hole. They are analogous to the images that would form on the retina of an observer peeking through each hole. By moving Leonardo's imaginary wall around, one would capture the light information at arbitrary viewing locations $\mathbf{v}$. In turn, each ray of the captured pencil can be parameterized by two angles θ and ϕ representing its direction. Physically, putting aside phase considerations, each light ray is characterized by an electromagnetic spectrum associating an energetic intensity to each visible wavelength λ . Finally, in a dynamic world, observations will vary over time t . Thus, the plenoptic function, whose purpose is to make possible the formation of a picture taken at any point from any angle at any time and is thus a formalisation of the light field, takes the ideal form: $P(\theta, \phi, \lambda, t, \mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z)$.

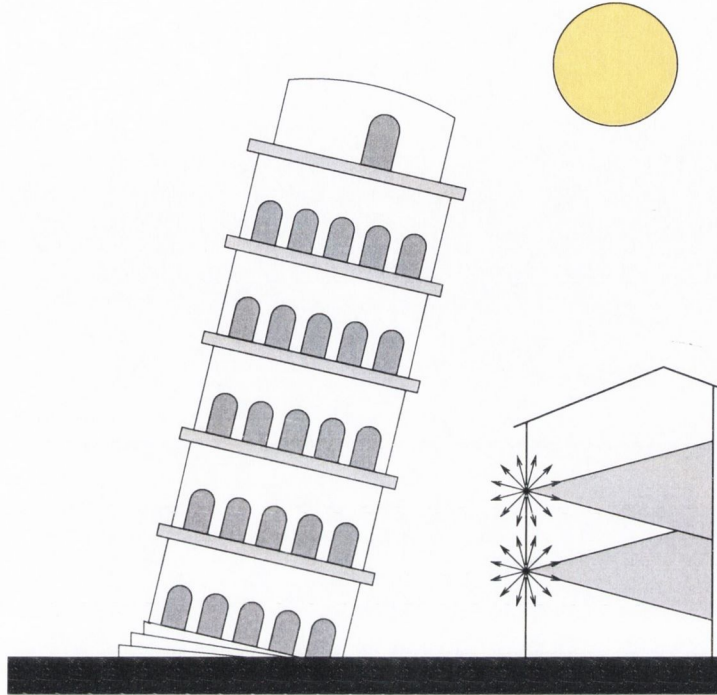


Figure 2.1: Leonardo's thought experiment.

Pinhole camera model

Leonardo da Vinci's quote, describing a phenomenon first recorded by the Chinese philosopher Mo-Ti (5th century BC), contains the principle of the *camera obscura*, a device in which light passes through a tiny hole into a dark chamber where it is projected onto a plane. This projection of three-dimensional objects onto a two-dimensional surface by straight lines passing through a single point is called perspective projection, it is illustrated in Figure 2.2.

In the pinhole camera model, the projection surface is a plane, the image plane, also known as focal plane. It is orthogonal to the direction that the camera is pointing in, which is characterized by the optical axis. The distance f from the image plane to the centre of projection is called focal length. By choosing a coordinate system as illustrated in Figure 2.2, it is straightforward to derive the coordinates (x,y,z) of the image of a point (X,Y,Z) as being:

$$x = f \frac{X}{Z} \quad y = f \frac{Y}{Z} \quad z = f \quad (2.1)$$

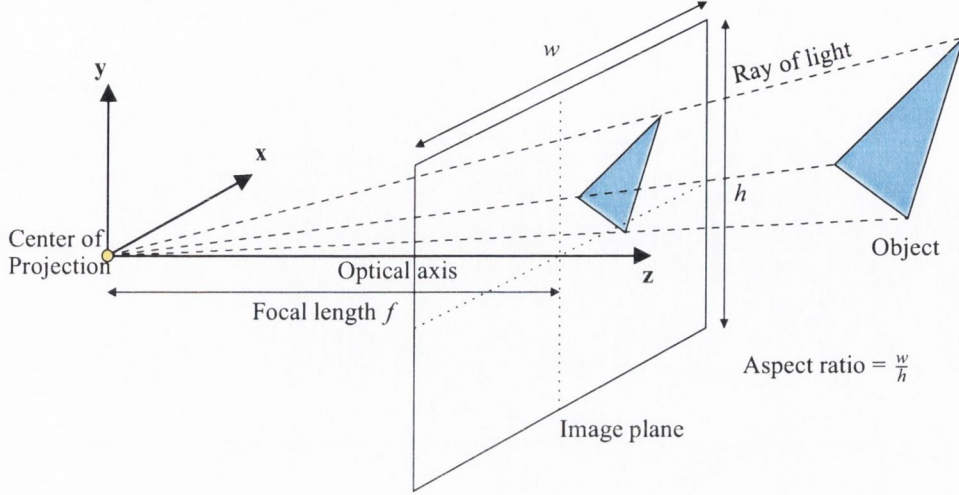


Figure 2.2: Perspective projection in the pinhole camera model.

This relationship is nonlinear because of the perspective division, which accounts for the foreshortening effect explained by Father Ted in Figure 2.3. It is, however, linear in projective space, and can be expressed as a matrix product using homogeneous coordinates, thus introducing the perspective matrix P :

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x'}{w} \\ \frac{y'}{w} \\ \frac{z'}{w} \\ \frac{w}{w} \end{bmatrix} \sim \begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

P

Which leaves the perspective division (to the left of the $\sim$ symbol) to be done last when going back from projective space to Euclidian space. Homogenous coordinates let us represent rigid transformations in one matrix multiplication, which is useful to incorporate the change of coordinate system M between the camera's and the world's as the former is moved around the latter. Finally, the coordinates of the projected point on the image plane can be re-mapped to take into account internal parameters of the camera, such as the dimensions of its photographic plate (or CCD sensor) and where it intersects the optical axis, by multiply-

ing to the left by an affine transform V . To sum up, if $\mathbf{p}_W$ and $\mathbf{p}_C$ are the positions of a point expressed respectively in the world's coordinate system and the camera's, and $\mathbf{p}'_C$ and $\mathbf{p}'_P$ are the positions of the image of that point expressed respectively in the camera's coordinate system and with respect to the photographic plate then:

$$\mathbf{p}'_P = VPM\mathbf{p}_W \quad \text{with} \quad \mathbf{p}'_P = V\mathbf{p}'_C, \quad \mathbf{p}'_C = P\mathbf{p}_C \quad \text{and} \quad \mathbf{p}_C = M\mathbf{p}_W \quad (2.2)$$

VPM is the camera's projection matrix, let us call it C . VP accounts for the camera's *intrinsic* parameters, while M accounts for its *extrinsic* parameters more often referred to as the camera's pose.



Figure 2.3: Father Ted: “These are small. But the ones out there are far away.”

In practise, an ideal pinhole camera cannot be built, because even if it was possible to create a hole the width of a ray of light, it would be impossible to record accurately the minuscule light energy that could pass through it. Real cameras use an optical system made of lenses to focus beams of light rays onto the focal plane, so that their aggregated energy matches the sensitivity range of the photographic plate. The camera's aperture, as well as its distance to the scene, determine the actual width of the beams of rays that get recorded. There is therefore always a compromise between the sampling accuracy of a light field, the lighting conditions of the captured scene, and the optical hardware used.

2.1.2 Techniques

We limited the scope of our work to the authoring of representations of static scenes under constant illumination. This choice is consistent with the fact that image-based representations are typically chosen to create and display content that is outside of the immediate vicinity of the virtual observer, which eliminates the need for interactive animation or lighting. Besides, the consideration of the time dimension and of the lighting conditions only introduce dimensionality problems that are orthogonal to the contributions we make. This survey therefore does not cover research on dynamic image based rendering, like for instance work on arbitrary viewpoint real-time video like that of Yang *et al.* [YEBM02] or Nozick *et al.* [NMA06]. It does not cover image based lighting either, though it is interesting to point out the duality between lighting and rendering. Image based techniques do indeed consist essentially of sampling and storing the light field in such a way that desired light rays can efficiently be queried, and can either be used to determine what a virtual observer sees, or to illuminate a scene. The work of Sen *et al.* [SCG⁺05] clearly illustrates this point. By taking pictures of a scene illuminated by a projector emitting different structured light patterns, they are able to produce a picture of the scene as seen from the viewpoint of the projector. Symmetric photography [GTLL06] goes further by demonstrating the first capture of a full (static) reflectance field, an 8 dimensional structure linking a scene's incoming light field to its outgoing light field.

It has been shown that when the captured scene's plenoptic function is well behaved, knowledge of the geometry of the scene can be leveraged to greatly reduce the sampling rate of the lightfield necessary for artefact free rendering [GGSC96], [CCST00]. A "well behaved" radiance function corresponds to a scene that exhibits few problematic light phenomena, such as transparencies over wide depth ranges or sharp anisotropic reflections. In such a scenario, it is intuitive to see that information on the position of surfaces that make up the scene can help to predict how the light field is structured and therefore sample it more accurately.

This places image based rendering at a junction between computer graphics and computer vision, where the problem of recovering geometric information from pictures is a central challenge. Although many advances have been made on this topic in recent years, it remains far from solved in the general case. This means that the availability of geometric information comes at the cost of making assumptions on the representation task. These can

be of different strength, depending on the desired information accuracy. At the lower end, rough implicit geometric information such as approximate optical flow fields can be relatively easily extracted via automatic feature tracking. In the middle ground, accurate depth information is likely to involve more resource intensive techniques, such as range scans. At the higher end, the creation of precise scene models still requires heavy human intervention in all but the simplest cases. There is thus a compromise between how finely to sample the light field and how much geometry to assume given or recoverable.

It would be beyond the scope of this thesis to provide a review of the various computer vision techniques that have applications in image based rendering. We refer to Olivier Faugeras's excellent book [Fau93] and Hartley and Zisserman's more recent treatment of the topic [HZ04] for further reading.

Figure 2.4 illustrates how the various image based rendering techniques presented in this survey are positioned with respect to the light field sampling vs. geometric information compromise (some techniques that should occupy roughly the same spot are placed side by side for legibility purposes, so the diagram should be read in broad terms). They will be presented roughly in order of increasing geometric knowledge of the scene. It is important to keep in mind that not all techniques reflect the same ambitions concerning the light phenomena that they can reproduce, which is why some appear to strike a better compromise. Restricting assumptions are mentioned in each case.

Light slabs and the Lumigraph

A classical parameterization of the light field was presented in two 1996 papers by Levoy and Hanrahan [LH96] and Gortler *et al.* [GGSC96]. It is based on a reduction of the plenoptic function to 4 dimensions. As seen earlier, Adelson and Bergen's original function is 7 dimensional. Time can be ignored with the assumption that scenes remain static. Furthermore, adopting the computer graphics representation of light in a 3-component space, the spectral dimension λ can be gotten rid of by evaluating the plenoptic function in that space. The fifth dimension is eliminated by observing that in empty space, the properties of a ray of light remain constant along its length. This means that the original 3D positions can be reduced to 2D positions along a surface of unit depth complexity that does not intersect the represented scene. The assumption is then made that the acquired light field will not be used to render novel views taken from within the enclosing surface, as this could violate the empty space

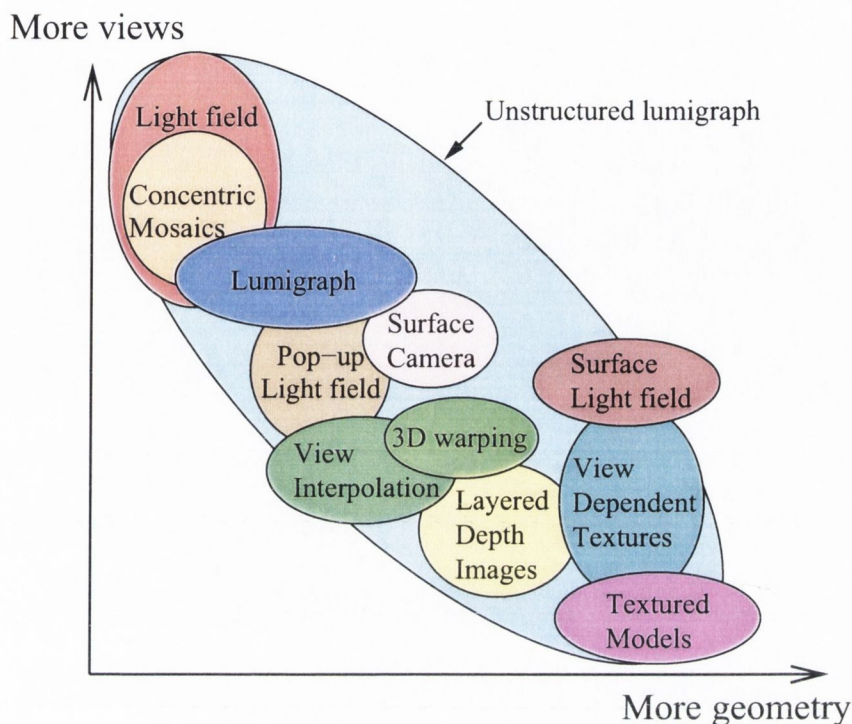


Figure 2.4: The trade-off between the geometric information available and the number of views necessary in image based rendering

assumption.

The 4D parameterization chosen is that of *light slabs*, consisting of two parallel planes with aligned Cartesian coordinate systems. Light rays are uniquely determined by the coordinates (s, t) and (u, v) of the two points at which they intersect the planes. Figure 2.5 illustrates the light slab concept. It is impossible to enclose a non empty volume with a single light slab without violating the empty space assumption. The traditional approach to representing a light field using light slabs is to arrange 6 of them in a cube shape enclosing the scene. Such an approach is also necessary when the desired viewing volume is located inside the captured scene, first because a single light slab does not discriminate between light ray directions (incoming or outgoing) and second because light slabs very badly sample rays that are close to being tangential to the planes.

The correspondence between a discrete computer representation of a light slab and the pinhole camera model is straightforward: rays can be captured by placing cameras so that

their optical centres lie on a regular grid along the (s, t) plane and their focal plane is parallel to the planes. (u, v) coordinates then correspond to pixels of the camera sensor.

Image formation for novel views is done by fetching the rays most relevant to each pixel from the ray database thus created, and then filtering them adequately. The main difference between Levoy and Hanrahan's *Light Field rendering* [LH96] and Gortler *et al.*'s *Lumigraph* [GGSC96] is that unlike the former, the latter takes advantage of available depth information to adjust the filtering kernel so that the reconstruction fits the plenoptic function better.

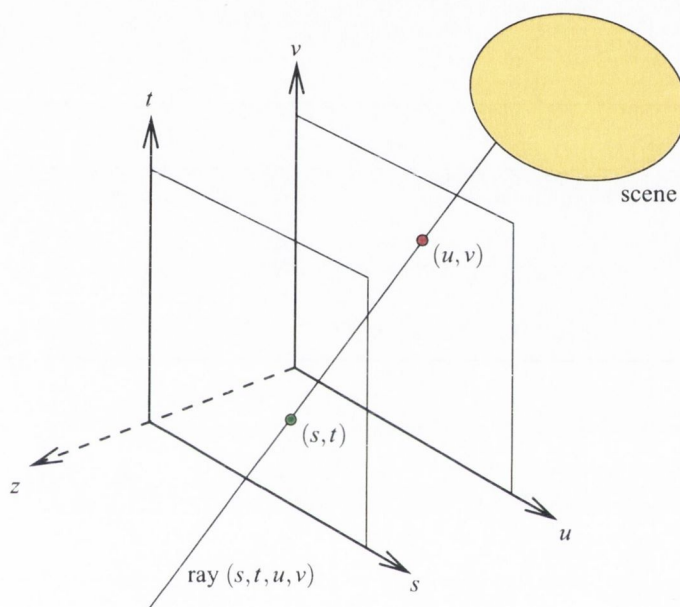


Figure 2.5: A light slab as used in Light Field and Lumigraph rendering.

Concentric Mosaics

With their *Concentric Mosaics* technique, Shum and He [SH99] place themselves in the particular case of a human observer walking around a scene, whose eyes are at a more or less constant height. They limit the positions of the reconstructed viewpoints to a plane, which allows them to propose a 3D representation of the light field. This is done by making the empty space assumption and parameterizing the position of the camera centres in polar coordinates (ρ, θ) while setting ρ to a constant radius R .

Scenes are captured by attaching a camera at the end of a rigid beam linked to the ground

at a fixed height by a vertical hinge joint. The camera is then rotated regularly and pictures taken at even intervals. Figure 2.6 illustrates the parameterization as well as the capture process. The pixel columns of each picture can be seen as the output of many vertical slit cameras. Novel views are generated column by column by interpolating relevant light slits. The authors point out that the approach is not physically accurate as it introduces vertical distortions. Results are nonetheless visually convincing and can be corrected to match reality if depth information is available.

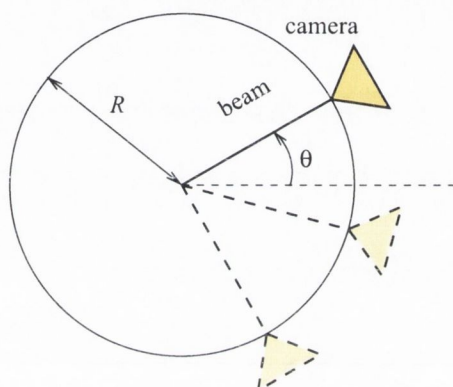


Figure 2.6: Concentric mosaics parameterization. The length of the beam being fixed, the pose of each camera is only parameterized by θ . If (u, v) are the coordinates of a pixel in the corresponding photograph, each ray of the concentric mosaic is identified by the triplet (θ, u, v)

Pop-up and Surface Camera light fields

Pop-up light fields and *Surface Camera (Scam) light fields* are two extensions of light field rendering that aim at taking advantage of sparse geometric information. With Scam light field rendering, Yu *et al.* [YMG02] propose to regroup the rays that pass through known surface points of the scene into auxiliary data structures called surface cameras (or Scams). Surface points are recovered from correspondences between images that can be obtained through automatic feature matching or user intervention. These correspondences are typically few compared to the total number of pixels. Rendering is therefore done in a hybrid fashion, using standard light slab interpolation for rays that could not be factored into surface cameras and projecting the relevant surface camera information for those that could.

The idea behind the pop-up light field [SSY⁺04] is to segment the captured scene into a few planar layers approximating its geometry, using transparency to handle occlusions. Shum *et al.* introduce a Bayesian matting technique to produce alpha mattes for each layer that are coherent from view to view. The scene segmentation and the matting process are user guided. Their rendering process is based on the Lumigraph, with modifications to correctly blend colours around transparency boundaries, and exploits the approximate depth information captured by the layers.

Dense correspondence techniques

Some techniques make use of dense geometric knowledge of the scene but without any topological structures. Such knowledge can be of different types: Implicit, as in the case of the optical flow, which describes how individual pixels move from one view to another as the viewpoint changes, or disparity maps, which store the amount by which each pixel gets displaced from one view to another; Or explicit in the form of a depth value for each pixel. They are all equivalent to solving the correspondence problem, which consists of identifying the projections of a given physical point into a pair of pictures.

Because of the lack of topological information, these techniques exhibit artifacts when reconstructing novel views from input views that are far apart when changes in visibility occur. This is because the flow of a pixel corresponding to a point seen from different positions is not injective: Pixels corresponding to different points in one view can come to overlap in another, meaning that a point that is visible in one view may not be in another.

View interpolation, proposed by Chen and Williams [CW93] uses the optical flow to compute intermediate viewpoints. This is done by simply interpolating each pixel's position along its optical flow vector between an image pair. Their method is only physically correct when the camera moves orthogonally to its optical axis from one view to the other, but it gives very convincing results when the initial views are not far apart.

Building upon the work of Eric Chen on the Quicktime VR format [Che95], McMillan and Bishop's *plenoptic modeling* [MB95] uses a cylindrical, instead of planar, projection model that allows them to conveniently represent panoramas. They transpose the traditional epipolar geometry of the planar case to the cylindrical case, which allows them to use standard computer vision techniques to solve the correspondence problem between cylinders obtained at different positions. Unlike Chen and Williams, they exploit the dense correspon-

dences to recover viewing positions for each cylinder and perform geometrically correct re-projections of the panorama as seen from novel positions. Finally, they introduce an efficient algorithm based solely on viewpoint position to ensure back to front rendering of the re-projected points.

The process of re-projecting an image from a different viewpoint using explicit pixel-wise depth information is called *3D warping* [MMB97]. It has become popular over the years, particularly to convey an impression of relief on textured polygons [OBM00].

Layered Depth Images were proposed by Shade *et al.* [SGHS98] to alleviate the one-to-many mapping problem from physical points to their projections in different views. Their principle is to store several colour values at different depths in the scene for each pixel, instead of only storing the value corresponding to the surface closest to the viewer (the one he sees).

All of the techniques presented in this paragraph are not targetted at faithfully reproducing complex view-dependent light phenomena, such as specular highlights or sharp reflections. They were presented in decreasing order of ability to do so. Intuitively, they tend to infer pixel colours primarily from how the projections of scene surfaces change in novel views, but pay decreasing attention to how the colours of these surfaces themselves change with the observer's viewing position.

View-dependent Texture Mapping

An extension of standard texture mapping, called *View-dependent Texture Mapping*, was proposed by Debevec *et al.* in [DTM96], though its most popular incarnation, described here, appeared in a subsequent paper [DBY98]. It takes as input a mesh describing the scene and a set of pictures of it, as well as corresponding camera parameters. Novel views are rendered using projective texture mapping to colour the mesh as if lighting it with projectors emitting input images with the same projection parameters as the cameras that were used to take them. The authors introduce a data structure called a *view map* to efficiently decide at rendering time which input views to use to texture each polygon. The criterion used to build the view map is viewing direction. For each polygon, the 2D space of viewing directions is sampled using a regular triangular grid. Each vertex of the view map gets assigned the input view that matches its viewing direction best. In order to ensure visual smoothness as the user changes viewpoint around the scene, the barycentric coordinates of the desired view's

viewing direction in the triangle it falls into in the view map are computed. They are used as weights to blend the three projected input views assigned to each vertex of the view map triangle.

A drawback of this method is that the only criterion considered to determine what input views are used to render a polygon is viewing direction. When inputs views are taken from different distances or with varying resolutions, it can happen that a view that matches closely the desired view's viewing direction is in fact less desirable than another because the resolution with which it samples the polygon is mismatched, thus causing aliasing.

Surface light fields

Like view-dependent texture mapping, Wood *et al.*'s [WAA⁺00] *surface light field* approach is based on the assumption that a mesh model of the represented objects is available. Even though it actually pre-dates Yu *et al.*'s technique, it can be understood as a special case of Scam Light Field rendering, where the entire light field information is accounted for by a dense set of surface cameras (called here *lumispheres*) evenly distributed over the mesh. Each lumisphere encodes outgoing light at a point of the scene's surface, so it is parameterized over directions (θ, ϕ) . In turn, a lumisphere's position on the scene's surface mesh can be encoded using two coordinates. Surface light fields are therefore 4-dimensional data structures.

Unstructured Lumigraph

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 other methods lack. It is therefore the approach that we chose to focus on in our work, so we present it here in more detail than the others.

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*, which 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 colour of each pixel of the desired view is to be modulated from the colours 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 novel 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 the triangulated geometric proxy.

Rendering is thus achieved in three steps:

1. The image plane is triangulated using the constrained Delaunay triangulation algorithm [Del34]. 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 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 non-null blending weights.

2.2 Authoring of Image Based Representations

There can be up to four steps in the process of building an image based representation of a scene, depending on the chosen rendering technique. They are:

1. **Capture** of the pictures that the dataset will be made of, along with the parameters of the camera model used to map pixels to light information.
2. **Addition of geometric information:** This is an optional step, as some techniques do not use any.
3. **View Selection**, which consists of choosing which views are actually used to compose the dataset. Note that in some cases view selection can be determined before capture is performed, for instance when the parameterization of the lightfield is rigid, such as with light slabs.
4. **Compression** of the dataset to make it more tractable to render.

We review the different approaches proposed in the image based rendering literature for each of the authoring steps.

2.2.1 Capture

The amount of effort needed to acquire views varies considerably from technique to technique. On the higher end are techniques that rely on a rigid parameterization of the light field. The apparatus required to record concentric mosaics has already been described. The original, and still widespread, way of acquiring a light field parameterized in light slabs is to use a camera array such as the one shown in Figure 2.7.

Recently, Ng *et al.* [NLB⁺05] have demonstrated a handheld camera for light slab capture. It functions by inserting a micro lens array between a regular SLR camera's optical system and its CCD sensor plate. Each micro lens splits the beam of light rays that would have been focused on a single CCD cell so that each sub-beam illuminates a different neighbouring CCD cell. This system in effect replicates the behaviour of a camera array. The down side is that the resolution in the (u, v) plane decreases by a factor of the square of the desired resolution in the (s, t) plane. They nonetheless demonstrate workable resolutions of $12 \times 12 \times 292 \times 292$ using a high end CCD sensor.

Gortler *et al.* [GGSC96] proposed a method to acquire a Lumigraph by taking multiple views of the scene with a consumer handheld camera. They use standard camera calibration techniques [Tsa92], tracking fiducial markers placed around the scene to recover intrinsic parameters and pose information. However, a lossy re-binning step is necessary to convert

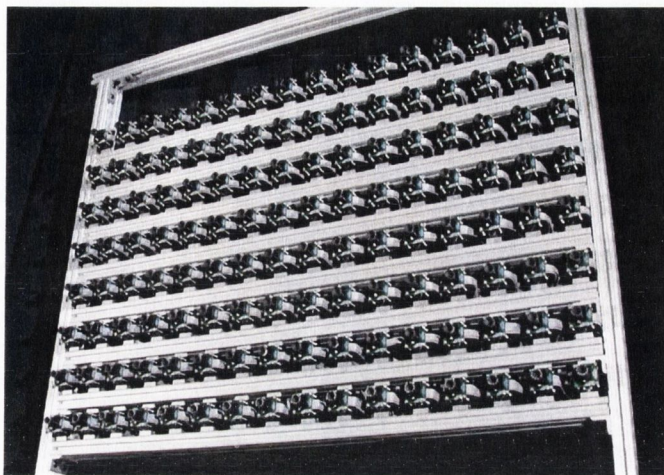


Figure 2.7: The Stanford multi-camera array (128 cameras) [VWJL04]

the unstructured light information obtained into a light slab. Coombe *et al.* [CHLG05] made that approach feasible for the construction of surface light fields by developing an online view insertion algorithm. The initial surface light field capture process [WAA⁺00] involved attaching the camera to a spherical gantry arm.

The other techniques surveyed do not rely on a regular sampling of the light field. As a result, they benefit greatly from recent advances in camera tracking (also known as *match-moving*) techniques that make it possible to recover camera pose information from sequences of images. Mature commercial products used in the industry include Boujou [2d3], PFTrack [The] or MatchMover [Reab]. Recently, the university of Hannover released a full-featured non-commercial tool called Voodoo [Dig].

2.2.2 Geometric information

Depending on the amount and quality of the required information, the acquisition of a scene's geometry can be a major obstacle to convenient authoring of image based representations.

Match-moving methods provide information on the optical flow, as they solve for camera parameters by extracting point correspondences from the scene. The set of correspondences obtained is however not as dense as required by image based rendering methods that expect implicit geometric information (with the exception of scam light field rendering). The quality of the output of conventional stereo methods that seek to extract dense correspondences from

pictures is still very sensitive to the content of the scene. Phenomena such as high spatial frequency occlusion (foliage), specularities, refractions and reflections all introduce noise that will cause artifacts upon rendering.

Methods that expect explicit depth information benefit from active research on the topic of range imaging. A somewhat wide choice of techniques is available, with different trade-offs between accuracy, cost and how cumbersome they are to deploy. They include:

- **Laser range scanning:** The principle is to emit laser rays towards the scene and deduce depth by triangulation between the emitted ray and its reflection [HHS⁺97].
- **Structured light techniques** work by projecting a regular light pattern, such as a grid, onto the scene. Depth is then obtained by taking a photograph of the lit scene and studying how the regular pattern was deformed when reaching the scene surfaces [ZCS02].
- **Depth from focus and defocus:** Here, several pictures of the scene are taken from the same viewpoint but with varying focal lengths. Depth is derived by analysing the blur in each picture to determine at which focal length the surfaces appear the sharpest [XS93].

The first two approaches share the weakness of the stereo correspondence techniques with respect to surfaces that are transparent or highly reflective. Depth extraction from defocus is more robust, and involves less cumbersome hardware, but it currently yields much lower depth accuracies.

The process of creating accurate geometric models of real scenes is even more challenging. The most common approach is to obtain accurate range scans from different viewing positions, register them together by matching a set of known features, and then run a marching-cube like algorithm on the resulting point cloud. The complexity of such a process limits its range of use to specific highly demanding settings where physical accuracy is important, such as the digital capture of heritage objects. A famous example of this process is the Digital Michelangelo Project [LPC⁺00].

As seen in the survey, image based rendering techniques that do make use of geometric models of the represented scene can make do with varying degrees of approximation. Coarse meshes can be created by an artist using the input images as a guide, a process called *image-based modelling*. Research in this area aim at facilitating the artist's work. Software tools

that have been proposed include Paul Debevec's Facade [Deb97] or RealViz's ImageModeler [Reaa]. They exploit epipolar constraints to help the user place feature points and make use of alignment constraints to guide modelling. Facade is specifically aimed towards the modelling of architectural scenes, taking into account additional volume constraints that arise in this case through the use of parameterized building primitives.

2.2.3 View Selection

Image-based representations tend to be memory intensive, their size scaling with the number of input views. It is therefore important to choose those views appropriately to maximize utility, a task that is, however, rarely intuitive.

Techniques that are based on a rigid parameterization of the light field present the advantage that it is possible to formalize conservative parameter offsets between views as a function of the scene's depth extent. Such formalization was done by Chai *et al.* [CCST00] and Lin and Shum [LS00] in the context of light slabs, but with the assumption that the scene surfaces are lambertian.

Hlavac *et al* [HLW96] were the first to study the problem of view selection for unstructured image based rendering, 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 [FCOL00], 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* [VFSH01] propose a similar technique inspired by information theory, using viewpoint entropy to guide the selection process.

Schirmacher *et al* [SHS99] 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* [CHLG05] 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 compressed light field using an online singular value decomposition algorithm. Their method pre-supposes a reasonably accurate geometric model of the captured scene.

2.2.4 Compression

Compression is another authoring concern where techniques can be split along the structured vs. unstructured divide.

Unstructured approaches make it difficult to exploit inter-view redundancies because there is no immediate way to formalize how light information stored in each view relates to that stored in the others. This makes the selection of an optimal set of input images even more crucial. Standard image compression techniques such as JPEG for disk storage and S3 for texture memory can of course be used to leverage intra-view redundancies.

Structured approaches on the other hand are well suited for the application of usual compression techniques thanks to their rigid parameterization that makes it possible to leverage data correlation along all dimensions. Uncompressed light slabs, concentric mosaics or surface light fields typically take up hundreds of megabytes. This makes it impractical to load them completely into main memory for interactive rendering. The challenge here is thus to provide efficient just-in-time decompression of local light information relevant to the current desired view. Shum *et al.* provided a survey of light field compression techniques [SKC03] in which they classified them into three categories: pixel-based, disparity prediction, and model-aided methods.

Pixel-based methods only take into account correlations between neighbouring rays. Levoy *et al.* [LH96] and Shum and He [SH99] use vector quantization in the context of light slabs and concentric mosaics respectively. Random access look-up can be speeded up by setting the vector size to a fixed value. These techniques yield modest compression ratios, in the order of 10 : 1.

Disparity prediction methods work along the same principle as MPEG encoding: views are subdivided into blocks whose motions from one view to another are estimated. As such, they exploit implicit geometric information in the form of a coarse optical flow. Similarly to MPEG, views are classified as either anchor views or predicted views. Discrete cosine transform-like techniques are used to compute the residual errors that allow for the reconstruction of predicted views from anchor views. Solutions were proposed by, among oth-

ers, Zhang and Li [ZL00] for concentric mosaics and Magnor and Girod [MG99] for light slabs. Wavelet based methods have also been proposed [WLLZ00] but their look-up speed is slower. Compression ratios achieved by those techniques are in the order of 100 : 1.

Wood *et al.* [WAA⁺00] proposed a compression scheme based on singular value decomposition when they introduced surface light fields (where the 3D geometry of the scene is known). In the same context, Magnor *et al.* proposed two approaches. In the first, data is converted into view-dependent texture maps that are wavelet compressed [MG00]. Their second approach uses discrete cosine transforms to predict new views, while the prediction maps are built by projecting the geometric model in each view to determine how corresponding pixels move from one view to another [MEG00]. The performance of both methods is sensitive to the accuracy of the geometric model, but the authors claim compression ratios of 1000 : 1 at peak signal to noise ratio (*PSNR*) of 30 decibels, which corresponds to perceivable but acceptable quality degradations.

All of the methods we found in the literature are lossy compression schemes, which is understandable as loss-less approaches would likely yield uninteresting ratios under the fast retrieval constraint. The implication of this is an additional hurdle for potential authors. Indeed, as explained in the next section, usual error indicators such as *PSNR* are only very rough indicators of perceived quality, particularly when compression is pushed aggressively. The optimisation of the compression for a particular scene thus involves a trial and error loop where the user evaluation step does not consist of simply looking at a picture or video sequence, but involves actively navigating around the rendered scene. This can become tedious if the compression process is not interactive. Coombe *et al.*'s online view selection and compression method introduced in the last sub-section constitutes an answer to this concern, albeit in the case where accurate geometric information is present.

2.2.5 Summary

Unstructured image-based rendering techniques such as view-dependent texture mapping and unstructured lumigraphs have the upper hand in terms of authoring ease. The unstructured lumigraph is particularly convenient because it gives the artist access to a wide range of compromises between the accuracy of geometric information and the amount of visual data necessary. With these techniques, the inter-views compression is implicitly done by selecting input views that are little redundant with each other. This makes the adequate selection

of views the main difficulty in creating such representations with a reasonable memory footprint.

2.3 Visual Fidelity Metrics

In computer graphics, particular effort has been expended to develop metrics and heuristics to measure or predict the fidelity of images. At the Campfire on perceptually adaptive graphics, James Ferwerda discussed "Hi-Fi rendering" [Fer01] and noted that physical accuracy is neither necessary nor sufficient to produce visually realistic images. He described three standards of realism that might help to define the criteria needed for good Image Fidelity metrics and has since elaborated on these ideas [Fer03]. Three types of realism are defined:

1. Physical realism, where the image provides the same *visual stimulation* as the scene depicted
2. Photorealism, in which the image produces the same *visual response* as the scene, and
3. Functional realism, where the focus is on providing the same *visual information*.

In this work we are preoccupied with the second type of realism, *i.e.* make use of visual fidelity metrics to facilitate the authoring of photorealistic image-based representations. This section briefly presents physically based metrics, provides an overview of the human visual system (HVS) and describes two perceptual metrics that exploit knowledge of its behaviour.

2.3.1 Physically based

Physically based metrics quantify the value-wise similarity between two signals $\mathbf{x}$ and $\mathbf{y}$. Traditional formulations are the root mean square error (*RMSE*) and the *PSNR*, expressed below assuming the signals are registered and sampled by n corresponding values encoded with B bits:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n |\mathbf{x}_i - \mathbf{y}_i|^2} \quad PSNR = 20 \log_{10} \left(\frac{2^B - 1}{RMSE} \right) \quad (2.3)$$

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.

Physical fidelity metrics indeed do a good job at confirming that two images will be perceived identical when the numerical differences between the two are small (typically for $PSNR > 35db$). However they do not model human judgement well. This is illustrated in Figure 2.8: on the left, the gold standard image (bottom) was slightly blurred while on the right, colourful scribbles were added. The $RMSE$ is higher for the blurred image than for the scribbled one, because in the first case tiny differences were compounded over the whole picture, adding up to more than the high but local differences of the second case. Thus, $RMSE$ yields a verdict that is the opposite of what a human observer would report.

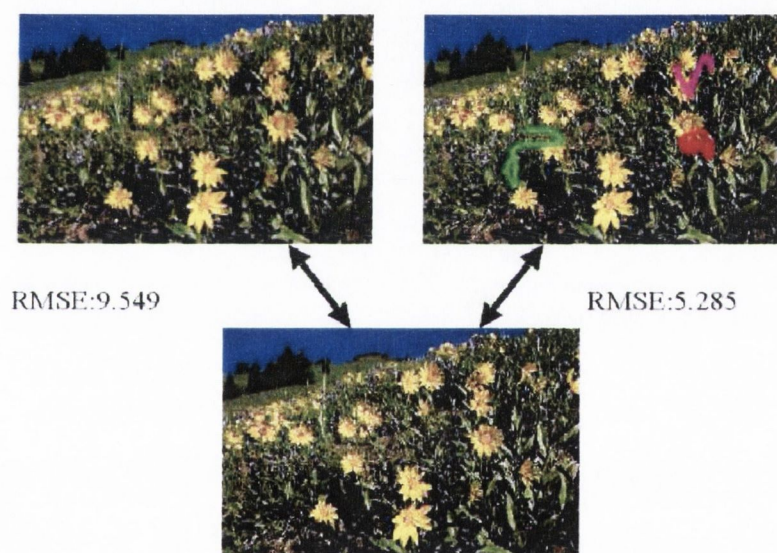


Figure 2.8: Behaviour of root mean square error on two kinds of image alteration.

The main avenue of progress for metrics that seek to quantify the similarity of images strictly from comparing pixel values is to express those values in a colour space that fits the behaviour of the human visual system better. We refer to the work of Ramanath *et al.* [RSH02] for a discussion of several CIE models.

2.3.2 The human visual system

Physiology and neurophysiology

Vision is a complex process that results from the visual cortex's interpretation of the information transmitted by the eyes through the optical nerves. A rough overview of the physiology of the human visual system is presented here. David Hubel's book can be consulted for further reference [Hub89].

The transformation of visual stimuli into nerve signals takes place according to the following mechanism:

Light enters the eye through the cornea. The pupil lets a certain amount of that light, depending on the wideness of its aperture, pass through the crystalline lens, whose function is to focus the light on the retina. Changes of focus, which allow the eye to accommodate for objects at different distances, are achieved thanks to the muscles of the ciliary body, to which the flexible crystalline lens is connected by strong microscopic fibers. Figure 2.9 shows a diagram of the human eye. The retina contains two kinds of photo-sensitive cells: rods, which are more light sensitive, and cones, which have faster response times and of which three types exist: red, green and blue depending on the wavelength range they are most sensitive to. Cones are thus responsible for colour vision and colour blindness is caused by the absence of one type of cone. The remainder of the retina is constituted of layers of neurons that process the electrochemical signal output by the photo-receptors and feed the result to the optical nerves. Within this structure, the individual outputs of single photo-receptors are grouped in clusters. Rods are grouped in clusters of much bigger size than cones. The resulting aggregate signal is thus very sensitive to light, at the expense of spatial resolution. This explains the predominance of cones in detail perception.

Varying illumination conditions result in three phases of vision: In the scotopic phase, which corresponds to very low light levels, such as on a moonless night, only rods are stimulated. Colours and details are thus not perceived. The intermediate phase, corresponding to moonlight illumination and in which rods and cones contribute in similar proportions, is called mesopic. In daylight, cones are dominant, which is the photopic phase. The adaptation symptoms that occur with changing illumination conditions is due to the physiology of the photo-receptors, which get damaged by high light intensities, but continuously regenerate over time.

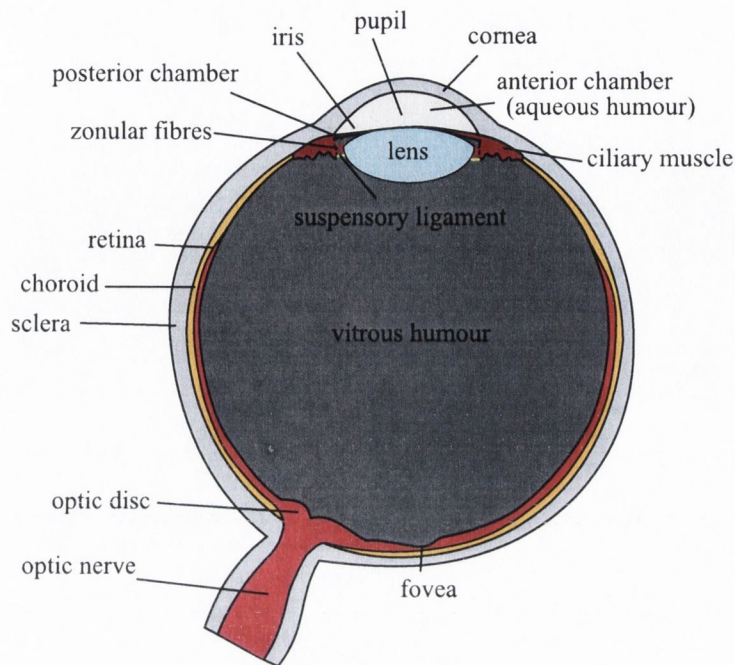


Figure 2.9: Schematic diagram of the human eye. From Wikipedia.

Psychophysical behaviour

We perceive the world in 3 dimensions through a variety of depth cues. Binocular disparity designates the differences between the images formed on the retina of the left and right eye respectively. The visual cortex interprets those differences and converts them into distance and depth information. There is a variety of monocular cues that contribute to depth perception. In no particular order, they are: occlusions, motion induced parallax, atmospheric perspective (which corresponds to contrast attenuation as light travels through more air the longer the distance), shading gradients, perspective effects on straight lines and object sizes.

These cues broach the subject of how the brain interprets the visual information relayed by the eyes. In spite of very intense neuroscience research, knowledge on this topic is still lacking. Besides, even a perfect knowledge of the neural mechanisms triggered by visual stimuli might not be enough to shed light on the sensation it generates. This is the domain of psychophysics, founded by Fechner in 1860 [Fec60] to investigate the relationship between physical stimuli and how they are perceived by subjects. Numerous psychophysical experiments have since made it possible to establish certain features of the human visual system,

including some perceptual invariants:

- **Luminosity invariance:** This refers to the ability of inferring the luminosity of a surface in spite of illumination or shading changes.
- **Colour invariance:** *ditto* for colours.
- **Shape invariance:** Designates the ability of attributing a fixed shape to an object seen from different viewing angles.
- **Size invariance:** How objects are perceived as having a constant size in spite of being observed from varying distances.

In order to exploit experimental data for image generation purposes, it is necessary to build computational models based on them. This task is made difficult by the nature of the HVS, which functions as a whole, as opposed to a sequence of separable processes, whereas experiments tend to isolate individual characteristics without shedding light on how they are linked to others.

Benham, a toy maker from the 19th century, designed a painted spinning top that illustrates such a link when observed while rotating with speed [Ano94]. One half of the spinning top's disc is painted black, while the other half is white, with black curved segments arranged in groove-like patterns (cf Figure 2.10).

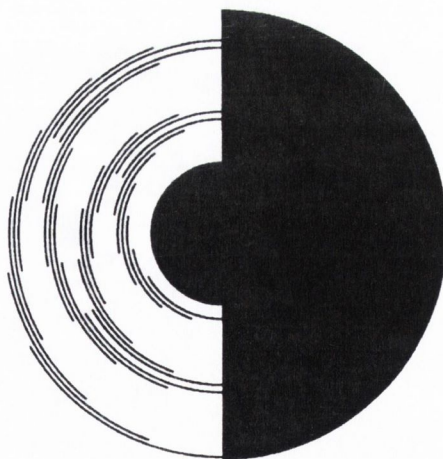


Figure 2.10: Benham's disc.

When the disc turns rapidly, red, yellow and green coloured concentric circles are observed, even though it is only made of black and white. These colours are called Fechner colours, after G. Fechner, who first described the phenomenon [Fec38]. They are a kind of subjective colour. To this date, there is no definitive explanation for this phenomenon. However, it clearly illustrates a case of interaction between the spatial and temporal channels of the HVS. The disc's rotation causes the intensity of perceived light to vary differently over time at neighbouring radii. This spatio-temporal stimulus perturbs the mechanism of colour perception. One conjecture is that the perturbation involves the slower response to light of the blue cones compared to the other types.

Another example of interaction is illustrated in Figure 2.11. On the right, the pattern is drawn flat, while on the left it is folded into a parallelepiped seen from an angle. Squares marked with an X have the same reflectance but are perceived to be of different reflectance on the parallelepiped. Conversely, squares marked with an O actually have different reflectances but give the impression of being of similar reflectance on the parallelepiped. Thus, the brain uses the inferred position in space of each square to account for the resulting differences in illumination when estimating their surface properties.

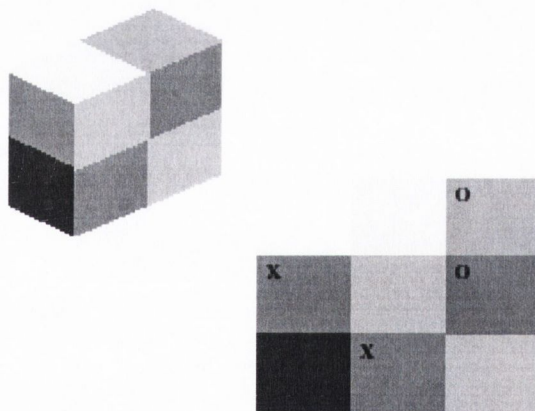


Figure 2.11: Two representations of a checker board.

Notwithstanding such difficulties, experiments have made it possible to quantify some aspects of the human visual system's behaviour, in particular, its response to contrast, which is of primary importance for image fidelity metrics.

Contrast is defined as the difference in colour or brightness that makes a part of an image stand out relative to neighbouring elements. It is usually computed as:

$$\frac{l_{max} - l_{min}}{l_{max} + l_{min}} \quad (2.4)$$

Where l_{max} and l_{min} are typically maximum and minimum luminosities within the region of interest, but can also be chroma component values within an appropriate colour space. Human perception of contrast exhibits several non-linear properties:

- It is adaptive: the higher the overall luminosity of the environment, the higher the contrast necessary to result in an identical perceived change in luminosity.
- Perceived brightness or colour depends on neighbouring values, which is called simultaneous contrast and is illustrated in Figure 2.12.
- Sensitivity to contrast changes with the spatial frequency of the observed signal.
- It decreases when the signal contains overlapping patterns of similar spatial frequencies, a phenomenon called visual masking.

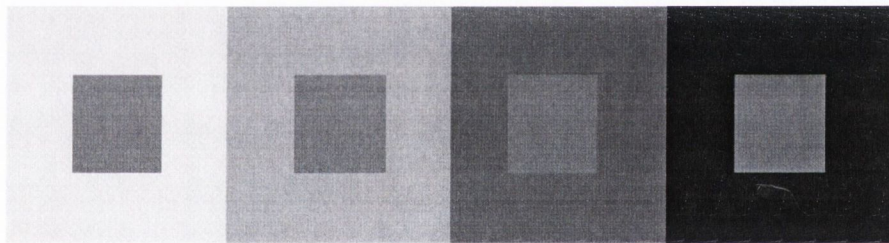


Figure 2.12: Simultaneous contrast. The central patch appears increasingly bright from left to right even though its luminosity is constant

The contrast sensitivity function is thus dependent on many parameters. More details on how it is modelled and incorporated in computer image fidelity metrics are given in the next sub-section.

2.3.3 Perceptual metrics

Perceptual metrics aim at predicting perceptible differences between images based on a simulation of how the HVS would respond to them. Here we present two such metrics that are among the most refined.

Daly's Visible Differences Predictor

The Visible Differences Predictor (VDP, [Dal93]) takes as input two images and returns a map containing the probability of a difference being detected between the two at each pixel. Figure 2.13 shows a diagram of the different steps of the process.

In order to model the adaptive behaviour and non-linear response of retinal neurons, a first transformation is applied to each image. Under the hypothesis that adaptation is a local phenomenon, Daly modifies the value of each pixel according to a function approximating the relationship between perceived luminosity and luminance. The function he uses is a compromise between the Weber-Fechner logarithmic law and Stevens's power law [Kru89], as it transitions from a cubic root at low levels of intensity to a logarithm at higher intensities.

The purpose of the next step is to account for the variation in the eye's sensitivity to contrast depending on the spatial frequency of the visual stimulus. This variation is quantified by fitting a numerical model to psychophysical measurements obtained in experiments. It associates a normalized sensitivity measure to the number of cycles per degree (*cpd*) of the visual signal. A map of spatial frequencies is produced for each image, which after application of the model yields a map of sensitivities that are used to weight each pixel's value.

The following steps model the detection mechanisms that take place in the visual cortex. Neurological recordings as well as psychophysical experiments have shown that these mechanisms can be separated over different spatial frequency ranges as well as different signal orientation intervals. Thus, the VDP performs a multi-resolution decomposition of each image using wavelet filters. This yields a channel for each frequency range/orientation interval pair. (Following experimental results, Daly chose to use 6 frequency ranges and 6 orientation intervals.) This decomposition is also practical to implement visual masking, a feature of the HVS that makes differences in a signal of a certain frequency less detectable when it is overlapped with a signal of similar frequency. By examining neighbouring channels, a detection threshold elevation map is computed for each channel. Those elevations are used to weight the contrast difference at each pixel when comparing each channel with its counterpart in the other image. This yields a pixel-wise probability of a difference being detected for each channel. The probabilities are then summed over all channels using the standard method (complement of the product of the complements) to obtain the final result. Figure 2.14 illustrates the application of the VDP to a pair of images.

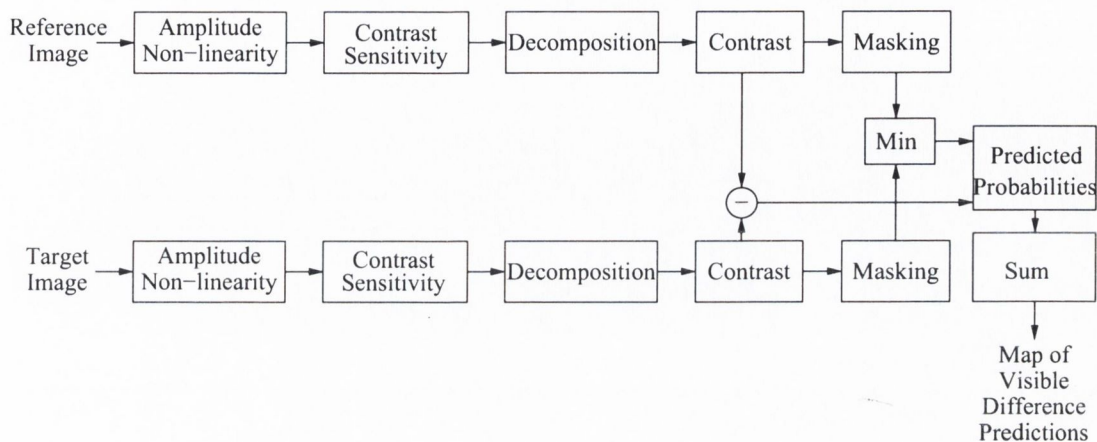


Figure 2.13: The VDP, diagram.

Sarnoff Visual Discrimination Model

This method, developed by Jerry Lubin [Lub95] for the Sarnoff laboratory, exploits roughly the same knowledge of the HVS. Contrary to Daly's, which uses a frequency representation, Lubin's technique stays in the spatial domain. To this aim, both images are first re-sampled so that their resolution matches the density of cones in the retina (120 pixels per degree (*ppd*)). This requires an estimate to be made on the distance from which the images are seen. This estimate is also implicitly present in the VDP under the form of Daly's choice of contrast response function. Lubin's choice of representation is a Laplace pyramid, a multi-resolution structure whose levels contain versions of the image of coarser and coarser resolution (which, for an analogy with frequency domain operations, can be viewed as resulting from the application of bandpass filters). Detection models are then implemented as convolutions of appropriate kernels.

2.3.4 Structural Similarity Index

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* [VQE00].

Wang *et al.* [WBL02] 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. They report encouraging preliminary user study results at a fraction of the cost of metrics simulating the response of the visual system.

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

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (2.5)$$

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 $\mathbf{y}$. C_1 and C_2 are constants linked to the dynamic range of the signal introduced for stability. We refer to [WBSS04] for a detailed derivation of this formula from luminance, contrast and structural similarity terms. 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 normalized to unit sum. This leads to a rewriting of μ_x , σ_x^2 and σ_{xy} over each window as follows:

$$\begin{aligned} \mu_x &= \sum_{i=1}^N w_i x_i & \sigma_x^2 &= \sum_{i=1}^N w_i (x_i - \mu_x)^2 \\ \sigma_{xy} &= \sum_{i=1}^N w_i (x_i - \mu_x)(y_i - \mu_y) \end{aligned}$$

where w_i are the Gaussian weights of the window.

2.4 Perceptually adaptive rendering

The basic idea behind perceptually adaptive rendering is to try and avoid computing visual details that would not make any difference to a human observer. As such it concerns itself with Ferwerda's second type of realism mentioned earlier: photorealism.

For quite some time now, progress in graphics hardware has made it possible to achieve photo-*plausible* rendering of dynamic environments at interactive framerates. What is meant by plausible, as opposed to realistic, is that corners are cut to produce visually convincing results without accurately simulating light transfers within the rendered scene. Examples of such shortcuts include: using texture maps to simulate reflections, using static pre-computed textures to account for indirect lighting, approximating indirect illumination with a constant ambient term or, more recently, ambient occlusion.

This illustrates that the main challenge of realistic rendering is to efficiently account for subtle light energy transfers between surfaces, as opposed to transfers from light sources to surfaces. These energy transfers can be formalized using a finite elements approach by a system of integrals relating the radiance of each surface patch to the irradiance of the others. Such a system cannot be solved in closed form and solutions are arrived at by convergence of iterative computations. Thus, a common feature of the many global illumination algorithms that have been proposed is to progressively take more and more light samples until convergence is reached.

From a purely physical standpoint, the minimum number of samples needed for convergence is given by the Nyquist-Shannon sampling theorem. Below that number, aliasing noise will occur. The perceptual angle is thus to ask how much aliasing noise one can get away with before an observer starts noticing degradation, or, put differently: what is the bandwidth of the incoming visual information that the human visual system (HVS) can discern? Some works in this direction are discussed below. Next, Dumont *et al.*'s perceptual framework for the rendering of global illumination solution is presented [DPF03]. It is interesting to note that ultimately, what a global illumination simulation yields is nothing else but a light field, thus Dumont's framework is amenable to image-based rendering.

2.4.1 Global illumination for static images

Studies have shown that the human eye is most sensitive to noise at intermediary spatial frequencies [Sar77]. Indeed, while spatial frequencies of up to 60cpd are perceptible, maximum response to noise takes place around 4.5cpd . Thus, the farther away the rendered visual signal gets from that frequency, the less necessary it is to super-sample it. Mitchell [Mit87] proposed a method to combat aliasing noise based on this observation by using adaptive super-sampling. Image areas where super-sampling is necessary are determined from a coarse preliminary rendering step. Zones where spatial frequencies are critical are detected using a contrast estimation function on pixel neighborhoods. Orthogonally, Mitchell's method uses a non-uniform sampling strategy in order to shift the aliasing noise towards higher frequencies, where it is less perceptible.

As explained earlier, the acuity of the HVS varies as a function of wavelength and is greater for luminance information than it is for chromatic information. To exploit this fact, Meyer and Liu [ML92] developed an adaptive rendering algorithm that uses a representation of light decomposed in chromatic and achromatic channels. They use an adaptive subdivision technique introduced by Painter and Sloan [PS89] to build a k-D tree representation of the image as it is being rendered, based on achromatic information alone. The higher the spatial frequency of the signal is in a region of the image, the deeper the branch of the tree corresponding to it is. Exploiting experimental results describing the response of the HVS to chromatic stimuli of varying spatial frequencies, Meyer and Liu limit the traversal of the tree depth-wise when computing the chromatic components. They conducted psychophysical experiments using images produced with their technique as stimuli. Results showed that taking fewer samples for chromatic channels had a lesser impact on reported image quality than doing so for achromatic channels. The starting hypothesis was thus validated, resulting in computation time savings. Bolin and Meyer [BM95] further refined this rendering algorithm to take into account visual masking and the non-linear nature of the eye's response to changes in luminosity (*cf.* 2.3.2).

Baining Guo [Guo98] proposed a progressive radiance evaluation technique that can be considered to be perceptually informed, even though it is not based on a comprehensive model of the human visual system. Its underlying principle consists of computing radiance samples in such an order that will maximize the speed of convergence to the final result. The process is driven through iterative construction of a Directional Coherence Map (DCM), which is an

irregular subdivision of the image into basic blocks that either correspond to smooth regions (smooth block) or discontinuities (edge block, complex or simple). At each iteration step, the current configuration of the DCM guides where to take new samples in order to refine the block classification. Roughly, children of smooth blocks are considered smooth if none of their corner values vary sensibly from the value obtained by interpolating their parent's corners. Otherwise, they are classified as edge blocks. Edge blocks are sampled along their boundaries to determine if they are complex (i.e., they contain more than one image edge). For simple edge blocks, a discrepancy direction is computed, which is used for interpolating the values to be compared with the samples taken at the next iteration for classification of the children. Rendering is done by linear interpolation of the samples, following the discrepancy direction in the case of edge blocks. How blocks are flagged (smooth or edge) in the initial regular grid has a great impact on convergence speed. Guo uses a perceptual criterion based on contrast over the samples corresponding to the corners of each block.

Even though the aforementioned approaches only take advantage of simple models of a limited set of features of the HVS, they gave promising results and encouraged the development of more comprehensive techniques. While they use known features of the HVS to modify existing rendering algorithms in a forward manner, the more ambitious techniques incorporate a simulation of the behaviour of the HVS as part of the rendering process to guide resource allocation and validate their output. This simulation takes the form of a visual fidelity metric measuring the perceptual closeness of the rendering to a reference image. In the case of Myszkowski's work [Mys98], Daly's Visual Difference Predictor [Dal93] is used while Bolin and Meyer [BM98] use the Sarnoff Visual Discrimination Model [Lub95]. Of course, in a typical rendering task, no ground truth image is available to us as a reference. However, the computing of global illumination solutions is a process that is easily broken down into a sequence of successive refinements. Examples include: taking more and more samples over a pixel or a surface patch, computing more and more bounces of a traced ray as it is reflected and refracted, or progressively refining the subdivision of surfaces used to compute radiosity energy transfers. Thus, the idea is to use the output of the previous rendering iteration as the candidate image and the current result as the reference image. The fidelity metric then provides information on the convergence speed of the solution in different parts of the image. Areas where refinements will not produce perceptible improvements can be ignored and resources allocated elsewhere.

In these techniques, the reevaluation of the fidelity metric at each iteration can outbal-

ance the computation savings made on the actual rendering task, particularly if the metric is modeling complex HVS mechanisms that are costly to simulate. Ramasubramanian [RPG99] proposed an optimisation based on the observation that most of the spatial-dependent component of perceptual fidelity metrics can be accounted for by direct illumination, because it reveals most of the high frequency content of the image (such as that resulting from geometric or texture details). Since direct illumination is essentially free in the context of global illumination, and considering that the contribution of indirect illumination is typically low frequency, this allows for the spatial component of the metric to be precomputed and reused at each step, leaving only the less costly luminance dependent component to be continuously updated.

2.4.2 Interactive rendering of global illumination solutions

Dumont *et al.* [DPF03] present a general framework, based on a decision theory approach, which uses perceptual criteria to handle resource constraints in interactive rendering of pre-computed global illumination solutions. The rendering of a frame is seen as the result of a set of rendering actions, each with an associated cost and utility. The cost represents the amount of resources needed to take the action. The utility measures the contribution of the action to the result. Resource constraints can then be met by running a resource allocation algorithm that will maximize utility.

Three applications of their framework are outlined in the following paragraphs. A common property is that a gold standard is available in the form of the global illumination solution. The utility is therefore defined as a measure of fidelity to this gold standard, as provided by a Visible Difference Predictor (VDP) [Dal93]. A key point is that, at equal cost, an ordering of the utilities of rendering actions is sufficient, as opposed to an absolute estimation.

The first application deals with diffuse texture management. The constraint here is the amount of texture memory available. The rendering actions then involve choosing the mip-map level at which each texture is stored. The utility function used is a customised version of the VDP proposed by Ramasubramanian *et al.* [RPG99] in their global illumination guiding framework: the spatial frequencies component, which accounts for visual masking, is pre-computed when the mip-map levels for each texture are generated.

The second application is an extension of the previous method to non-diffuse reflections. The authors choose to render them using prefiltered environment maps. The constraints are

the memory space available for environment maps as well as the time needed to compute and filter them. Because of view dependence, the spatial frequency term of the VDP has to be approximated.

The third application aims at simplifying the radiosity mesh from the global illumination solution so as to meet rasterisation limitations, which constitute the constraint here. Rendering actions here involve the level of subdivision at which radiosity mesh elements are displayed; The more finely subdivided, the more costly. The utility function is again a specifically tailored VDP.

In situations where low frame rates occurred when no allocation scheme was used while resources were scarce, the use of perceptual allocation allowed interactive frame rates to be achieved with minimal perceptual degradation.

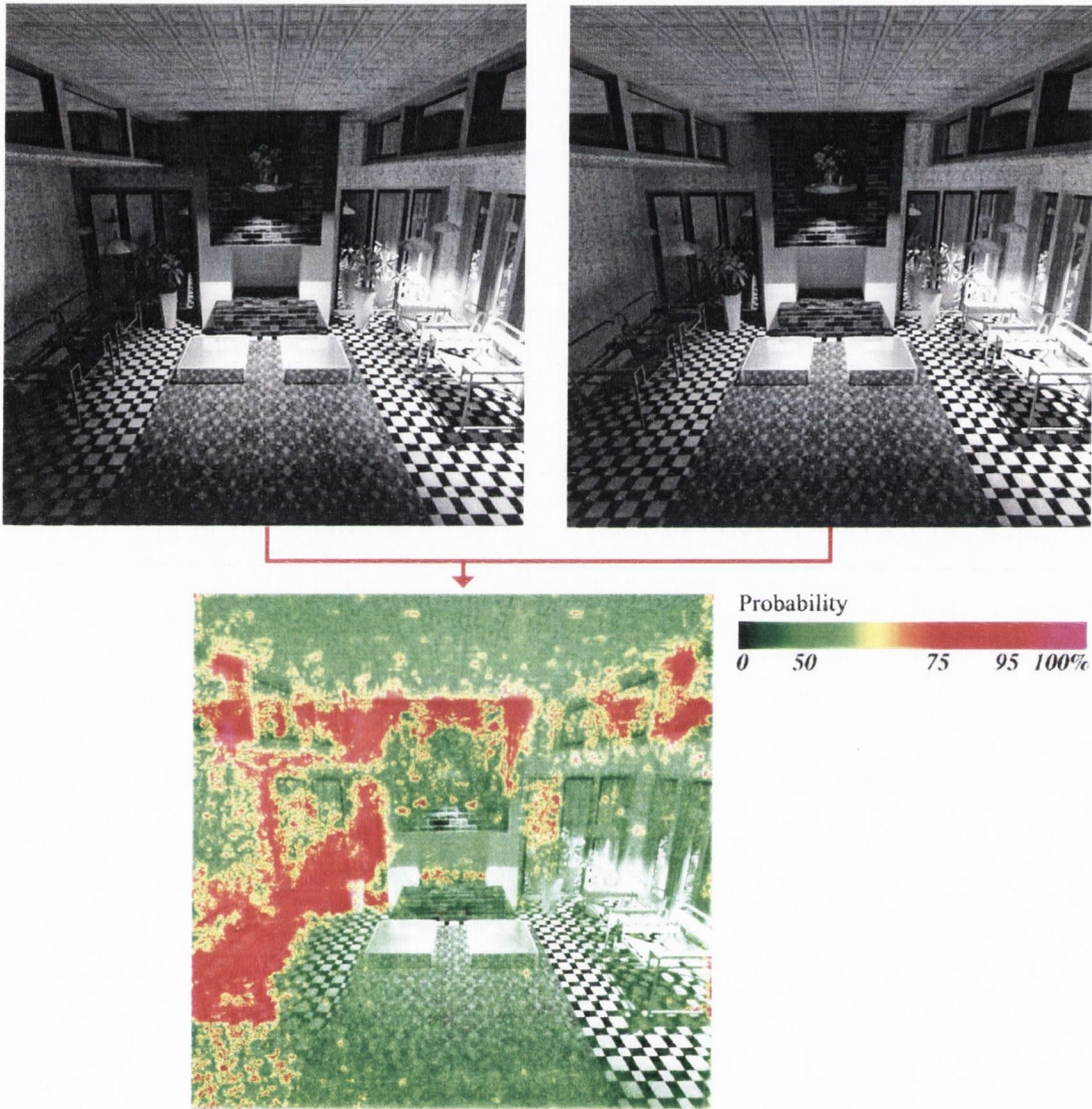


Figure 2.14: A pair of images and the corresponding VDP using the contrast sensitivity function proposed by Daly. (*Images courtesy of Karol Myszkowski [VMKK00]*)

Chapter 3

Perceptual importance of indirect lighting as a function of luminosity. A psychophysical experiment.

In this chapter, we first introduce rendering challenges where a consideration for human visual perception can make a difference and discuss a selection of influential works on the topic. We then describe a psychophysical experiment that we conducted as a contribution to this topic. Finally we discuss the conclusions that this preliminary work made us reach and how they influenced us to shift the focus of this Ph.D. Namely from leveraging knowledge of human visual perception to save machine-hours in the form of rendering time savings, to doing so to save man-hours in the form of reducing visual content authoring times.

The initial impetus of our research was in the direction of perceptually guided rendering, which we have introduced in the background section (2.4). In this chapter, we explain how the process of conducting a psychophysical experiment for application in global illumination algorithms, and its outcome, gave us insights that motivated us to alter our line of research.

Gilchrist *et al.* demonstrated that studying the perception of surface lightness is particularly fruitful to gain insight into the behavior of the human visual system (HVS) [AGJ83, GJ84]. Several perceptual cues, including depth, occlusion, surface orientation and perceived illumination are known to be involved in the correct perception of lightness.

The experiment presented here is an extension of work by Ann McNamara et al. [McN00, McN01]. Following their work it is based on a lightness matching task. Comparison of

perceptual responses to images rendered at different quality levels with responses to a real scene had hinted at conditions where quality degradation had little impact on perception.

This study investigates the effect of one such condition, namely the level of illumination. Image quality is adjusted through the indirect illumination term, as it is the most computationally expensive component of lighting simulation and has therefore the most time saving potential.

We first describe the controlled test environment that was created for the experiment, detailing the physical framework and the modeling process. We then present the experimental design. An analysis of the results is then performed, followed by a discussion detailing how the experience gathered influenced our research direction.

3.1 Controlled test environment

The nature of the experiment, where the real world is compared to computer generated pictures, necessitates the creation of a physical scene that can be accurately modeled. The most famous ancestor of such an endeavor is the Cornell box [GTGB84] and our setup, partly borrowed from McNamara [McN00], is quite similar to it.

3.1.1 Physical framework

The real scene The test environment is a five sided box of $915 \times 915 \times 915\text{mm}$ dimensions. Its interior is painted with white matt paint. The box is divided into three regions of equal height using two horizontal shelves. In each of those regions, five objects are positioned in no particular order, yet in a way that guarantees proper visibility for an observer looking at the inside of the box. The positions of each triplet of objects are identical from region to region. The objects are: a cube, a parallelepiped, a sphere, a pyramid and a cylinder. Each triplet of objects is painted in a random grey level.

Illumination The light source consists of a single 24 volt quartz halogen bulb mounted on optical bench fittings at the top of the box. It is fed by a stabilized 10 amp DC power supply, stable to 30 parts per million in current. The light shines through a 70mm by 115mm opening at the top of the box. A wide slit (155mm) running from back to front is made in the top

shelf to allow some light to flow into the middle region. A narrower similar slit (28mm) in the bottom shelf lets an even smaller amount of light reach the bottom region.

Set-up A 21-inch monitor is used to show the rendered images. The observer is separated from the test environment and the monitor by a thick black curtain. Monocular vision is permitted through a small aperture in the curtain, since the use of a single monitor does not allow the rendered images to be presented stereoscopically. The monitor is placed on a wheeled table for ease of positioning. Marks on the floor indicate the position of the table so that the observer sees the monitor through the aperture, while hiding the box behind. Black cardboard is used around the box and the monitor to conceal other parts of the environment that could be seen through the aperture. Two diopters (lenses) are used to ensure that both the real environment and the rendered images look the same size to the observer. A slot on the opposite side of the aperture from the observer allows alternation of the diopters. The set-up is illustrated in Figures 3.1 and 3.2.

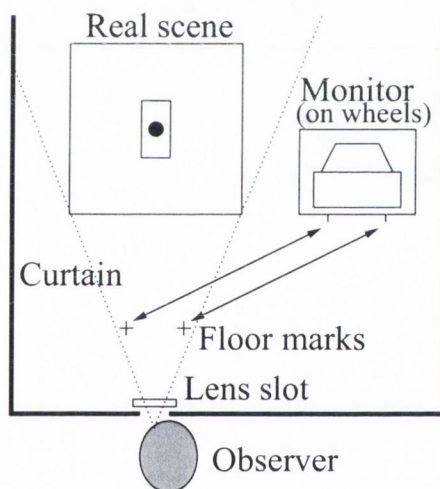


Figure 3.1: Experiment set-up, diagram.

3.1.2 Modeling

The images used in the experiment were produced using the RADIANCE [War94] package. RADIANCE is a physically based suite of rendering programs, capable of generating accurate,

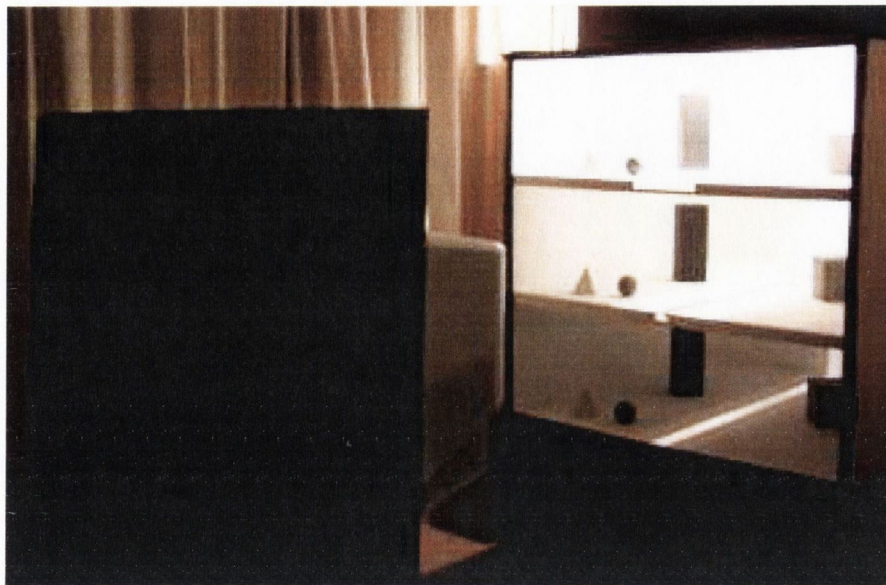


Figure 3.2: Experiment set-up, photograph.

highly realistic imagery. For the lighting simulation to be as accurate as possible, great care was taken when modeling the test scene.

Geometry. As the objects were simple shapes, the box itself and the test objects were modeled using standard primitives. The dimensions and positions were reproduced with a precision of the order of one millimeter.

Lighting. The chromaticity of the light source was obtained by illuminating an Eastman Kodak standard white powder - known to reflect 99% of light - and measuring it with a chromameter. Being the only light source used in the RADIANCE simulation, its intensity could be arbitrarily set. The brightness of the final images was matched with that of the actual photograph by adjusting the exposure parameter. All images were carefully gamma corrected for optimal display on the monitor.

Materials. The material values were measured using a chromameter to give Y_{xy} values. Based on the reflectance of the Eastman Kodak Standard the reflectance values for each material was calculated using standard Y_{xy} to RGB transformations [Tra91].

3.2 Experimental design

Participants. A single group of fifteen participants was used throughout the whole experiment. All were naive as to the purpose of the experiment. All reported to have normal or corrected to normal vision.

Task. The participants were presented with various visual stimuli: the real scene, a series of rendered images and a photograph (shown on the monitor). For each of those, they were asked to rate the lightness of each object on a thirty level scale. Unlike in McNamara's work, we did not use a grey level reference chart. There are two reasons for this choice: Firstly, we were interested in the relationship between the ratings and not their absolute values. Secondly, a chart either requires the participant's vision to switch between adaptation to the stimulus and adaptation to the chart, or that the participant learns the chart beforehand if that is to be avoided.

Ordering effects. In this experiment, the only factors whose impact on lightness perception interests us are the level of illumination and the nature of the visual stimulus. A good way to make sure that data is not corrupted by the effects of other influences is to factor such influences out in the experimental design. In our case, we wished to eliminate any undesirable effects of the order of presentation of the stimuli and the order in which each participant was asked to rate the objects. To counteract any such ordering effects, the order in which each stimulus was presented was random, as was the order in which participants rated the lightness of each object within the stimulus configuration under consideration. Since there were ten different configurations with fifteen objects to rate in each, it was impossible to get the same random sequences by chance. There was therefore no need to introduce manual counterbalancing.

Stimuli. Along with the real environment (**R**) and the photograph (**P**), eight images were presented, each rendered at various levels of quality. What varied between them was the amount of effort put into the computation of the indirect illumination term for each region's objects. Four levels of *bouncing* (*i.e.* the contribution of reflected rays is taken into account until the fourth iteration) is an appropriate value to get a good lighting simulation under RADIANCE. A test image was computed using this setting for the whole scene (**NO**). In

each of the other images, a set of objects was chosen to be rendered using only one level of bouncing. Those sets were: the top region objects (**T**), the middle region ones (**M**), the bottom ones (**B**), both top and middle (**TM**), both top and bottom (**TB**), both middle and bottom (**MB**) and finally all objects (**ALL**). The images are shown in Figures 3.3 to 3.7. The narrowness of the slits ensure that the amount of light reflected by objects in a region have a negligible effect on objects in other regions. Thus, altering the amount of bouncing for objects in a region has no significant consequence on the amount of light received by objects in other regions.

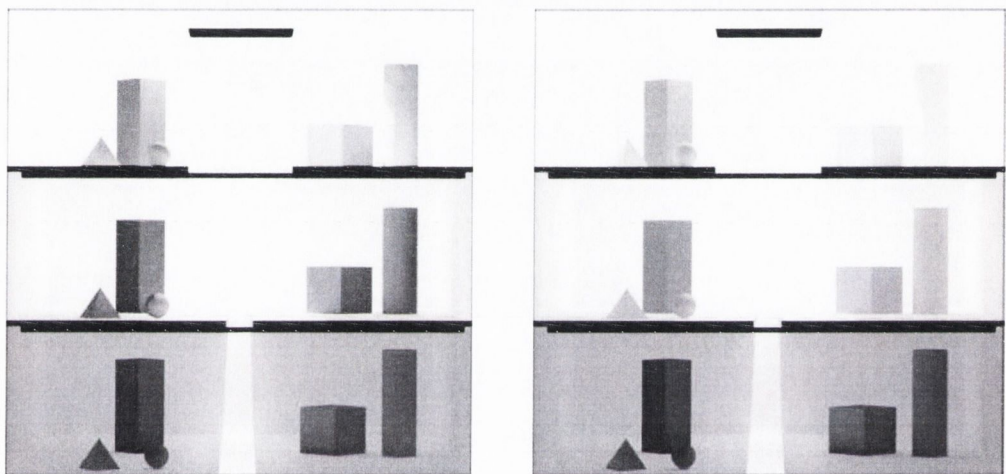


Figure 3.3: Left: all objects (**ALL**) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the bottom shelf (**B**).

3.3 Results

3.3.1 Pre-analysis

We singled out two factors in our experiment design:

1. The nature of the image presented to the participant.
2. The amount of light an object receives, *i.e.* the region of the box it belongs to.

We are interested in determining whether those factors have an impact on how close the participants' perception of each image is to their perception of the real environment. There-

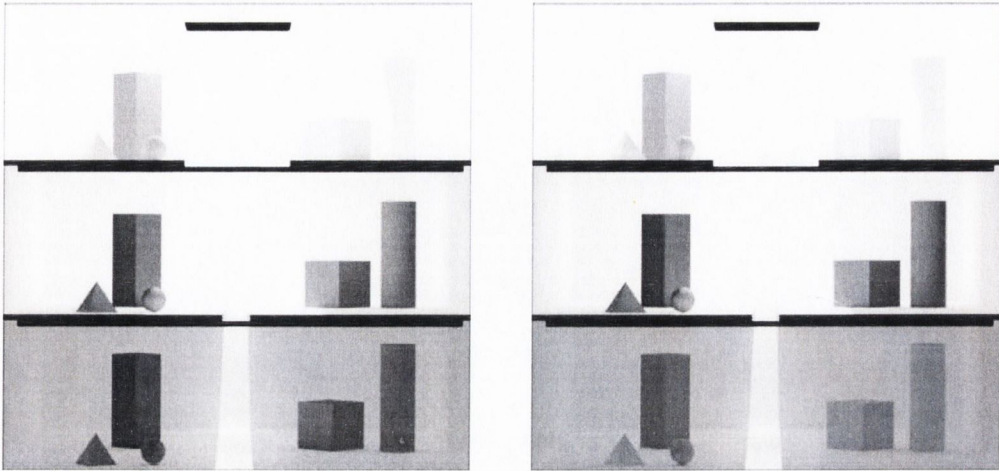


Figure 3.4: Left: objects of the bottom and middle shelves (**MB**) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the middle shelf (**M**).

	F ratio	Hypothesis <i>df</i>	Error <i>df</i>	Significance
Image	12.394	8	67	0
Region	10.887	2	73	0
Interaction	9.92	16	59	0

Table 3.1: ANOVA results.

fore, the values that we use are not the actual ratings but the absolute value of their difference with respect to their counterpart given in the real environment. We use analysis of variance (ANOVA) to test for significance. Since we used the same group of people throughout the whole experiment, we choose a repeated measures design. Having two factors, we directly perform a two way ANOVA. The results are given in Table 3.1. They show that both factors have a significant effect (at a level of confidence better than 0.01), and that there is also a significant interaction between them. This leads to further investigation to determine what these effects are.

3.3.2 Correlations

Statistical Correlation measures the degree to which two variables are linearly related. If they behave the same, their correlation coefficient is one. A coefficient of zero indicates that

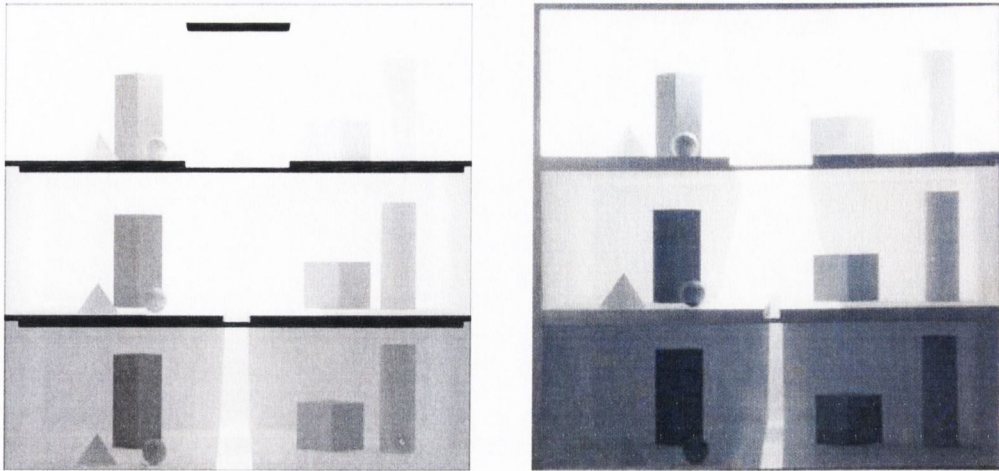


Figure 3.5: Left: no objects (NO) rendered with less resources allocated to indirect illumination, Right: photograph (P) of the real scene.

there is no linear relationship. Correlation between the ratings obtained with an image and the ratings obtained with the real environment is therefore a good indicator of the perceptual fidelity of that image. The correlation formula used here is Spearman's ρ . The use of Pearson correlation has been excluded because the values being considered are subjective ratings and not exact measurements. We use a modification of the Spearman formula to take into account the presence of tied values in our data. Each correlation coefficient is associated with a probability that this level of correlation could be obtained by chance. Below a certain threshold (0.05), the correlation is considered significant.

Overall First, all pairs of values available (from each participant for each object) were used to compute the correlation between each image and the real environment. They are given in Table 3.2. Because there were as many as 225 (15 participants $\times$ 15 objects) pairs of value for each image, results are always significant at the 1% level even when the correlation is weak.

The results show that the photograph is perceptually the closest to the real scene. However, it is followed very closely by the computed image for which no savings have been done on indirect illumination. The fidelity is greatly degraded whenever savings are introduced, to the notable exception of the top region. This can perhaps be explained by the fact that the top objects receive much more direct light than their bottom and middle counterparts, making

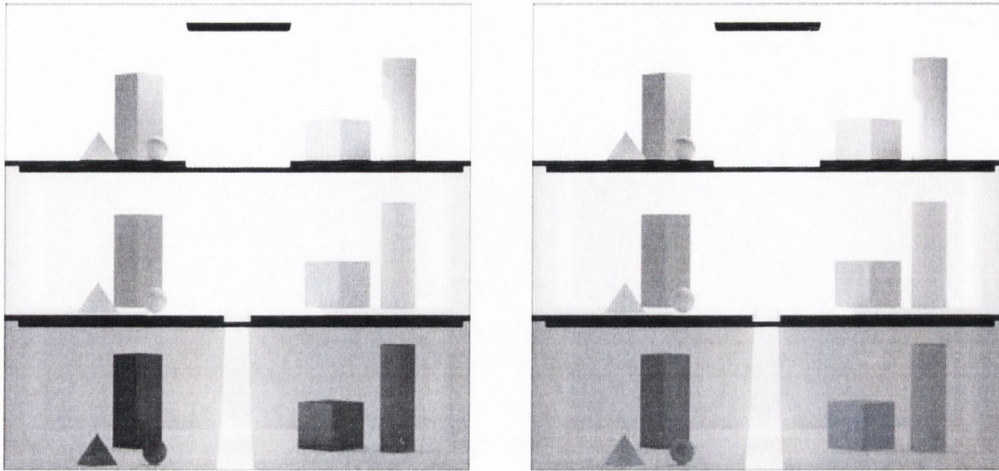


Figure 3.6: Left: objects of the top and bottom shelves (TB) rendered with less resources allocated to indirect illumination (one bounce), right: only the objects of the top shelf (T).

ALL	B	MB	M	NO	P	TB	T	TM
0.378	0.342	0.431	0.395	0.623	0.645	0.388	0.592	0.378

Table 3.2: Overall correlations.

the indirect illumination term negligible, or at least sacrificable at a smaller perceptual cost.

Across regions Here the values taken for correlation are the participants’ ratings for all objects of each region taken independently. The results are given in Table 3.3. The second value is the probability of the correlation occurring by chance. Both values are written in bold if significance is at the 5 % level or better. than .05.

Reducing the number of bounces used to render a region reduces the correlation for this region. This confirms that omitting the indirect illumination term tends to degrade perception.

In accordance with the previous paragraph, the top region is the one that suffers least from savings.

As the ANOVA analysis hinted through the interaction between factors, an alteration of the rendering conditions for one region has a perceptual impact on regions whose conditions remained the same. In our view, this phenomenon is a consequence of the adaptative nature of the HVS and the fact that it is sensitive to relative differences in luminosity, as opposed

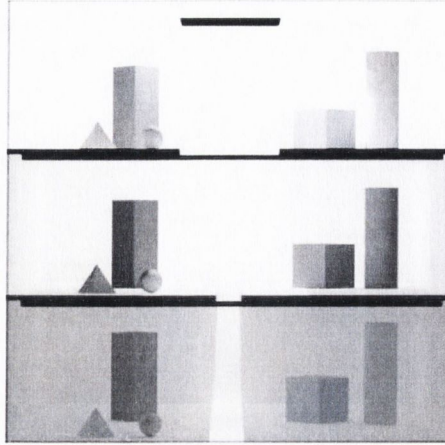


Figure 3.7: Objects of the top and middle shelves (TM) rendered with less resources allocated to indirect illumination (one bounce).

	ALL	B	MB	M	NO	P	TB	T	TM
Top	0.639	0.615	0.535	0.567	0.706	0.702	0.508	0.546	0.58
	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>
Middle	0.147	-0.025	0.622	0.224	0.631	0.799	0.429	0.623	0.214
	<i>0.209</i>	<i>0.831</i>	<i>0</i>	<i>0.054</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0.065</i>
Bottom	0.387	0.354	0.364	0.488	0.618	0.716	0.298	0.542	0.428
	<i>0.001</i>	<i>0.002</i>	<i>0.001</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0.009</i>	<i>0</i>	<i>0</i>

Table 3.3: Correlations per region.

to absolute values. Altering luminosity in a part of the image then leads to an alteration of perception on the whole image.

In some cases, it even appears that perception for a certain region can be more degraded by savings made elsewhere than savings made in this very region. Besides, the middle region results are conflicting. On the one hand, the image where only the bottom region is excluded from accurate indirect lighting yields negative non-significant correlation (-0.025 , the worst result by far). On the other hand, a fairly good correlation is observed when savings are introduced in both middle and bottom region. Those facts could be an indication of the relative importance of the adaptation effect with respect to the actual omission of the indirect lighting term. This interpretation is made all the more plausible as the middle region luminosities are situated at the middle of the luminosity interval for the scene. Intuitively,

altering the extrema of this interval has a great impact on the perception of those more subtle values.

3.4 Conclusions

3.4.1 Regarding the experiment

An experimental framework to assess the impact of indirect lighting on lightness perception has been presented and the results discussed.

It appears that indirect lighting is of great perceptual importance, especially for objects receiving little direct light. The results also show that it is a sensitive term to tamper with, since a local modification can have global repercussions. However, such repercussions are only likely to appear if the local luminosity perturbation was high enough to be perceptually unacceptable.

Currently, RADIANCE allows for computational savings on the indirect lighting term. It is done by providing a global parameter that specifies under which proportion of light contribution a ray can be ignored. This allows for two approaches to improve efficiency. First, instead of proceeding on a ray by ray basis, negligibility estimations could be done at a coarser resolution. Secondly, instead of using a user-defined parameter that is constant over the whole image, it could be locally adjusted by a mechanism controlled by the luminance Contrast Sensitivity Function [Dal93].

This line of research has been further elaborated upon by Stokes *et al.* [SFWG04], who subsequently presented their approach to high quality global illumination rendering using perceptual illumination components. Their work is based on the fact that the illumination of a surface can be split into components that are separately computable, namely: direct, indirect glossy, indirect diffuse and indirect specular illuminations (if one is to neglect the interaction between the latter three).

Their aim was to produce a perceptual metric taking this decomposition into account to drive rendering. They started by conducting a perceptual experiment to obtain data on the relative visual importance of each component. A test scene representative of typical global illumination scenarios was rendered from different viewpoints, each component separately, then the results blended in all possible combinations to serve as visual stimuli for the experiment (a full global illumination solution was also computed to serve as gold standard).

Participants were asked to sort the images by perceived quality. The results confirmed the marginal perceptual contribution of light path interactions between the three indirect components (only present in the gold standard). The paper explains thoroughly how a mathematical model was fitted to the experimental data to formulate the metric. Rendering of novel scenes was then driven using the metric to predict the relative importance of each component (scene-wise) as a function of the materials visible from the desired viewpoint.

3.4.2 Influence on subsequent research

We did not share Stokes *et al.*'s confidence that a qualitative observation on the varying importance of a component of the light information could be quantitatively leveraged on a variety of scenes. The principle of using a metric derived from a particular scene, or set of scenes, however representative, to guide the allocation of computational resources to different light components for the rendering of novel scenes with non trivial composition seems either risky or without effect. Let's consider the presence of convex mirror in a small portion of a novel scene that is otherwise made of lambertian surfaces. The light phenomenon induced is not local: the mirror will reflect light coming from a wide portion of the scene. There are two possible behaviors for the metric: either it neglects the presence of the mirror and allocates little resources to indirect lighting, resulting in erroneous rendering. Or it does not, allocating a proper amount of resources to indirect lighting over the whole scene, as would have happened had it not been used. This is not to say that perceptually guided global illumination does not yield results (the opposite is stated in the background section 2.4.1), but rather that our initial research direction seemed unpromising to us.

As seen in the introduction, the challenging topics of computer graphics are starting to move away from rendering issues. Techniques to perform global illumination computations entirely on the graphics card have already been proposed [BSKS05], and it will likely not be long before they mature. GPUs are essentially parallel computing devices. Perceptual approaches do not sit well with this fact because they introduce conditionality: they function by deciding whether or not to compute. Branching makes it more difficult to leverage parallelism because it introduces path-dependency between variables. Perceptual metrics also invariably involve summations of the visual signal over large areas, a type of computation that does not lend itself well to parallelization and that graphics processors are not good at.

Regarding highly realistic rendering that is still done on CPUs, the perceptual approach

can be a non starter if physical realism is the goal. When it is applicable, it deals with stakes that have been considerably reduced since the beginnings of computer graphics: Machine-hours have become cheaper and cheaper over the years and their “productivity” has been increasing at the speed of Moore’s law. It is now possible for independent animation studios to render their projects internally on small farms of commodity PCs.

Visual content creation is an area of computer graphics where the stakes are on the increase in the industry. Video game and film projects now employ large teams whose task is to painstakingly create 3D models, texture them and develop shaders in order to build believable environments from the ground up. Image based modeling and rendering is a commonly used tool in this process. From a perceptual standpoint, image based techniques present the advantage that a gold standard is inherently available in the form of the input images that the artist works from. This makes it more natural to apply perceptual metrics than in the case of traditional rendering. The time constraints involved in content creation are several orders of magnitude longer than in rendering, as a result the time needed to evaluate perceptual metrics is much less of an issue.

These are the reasons why we changed our research direction from perceptually adaptive global illumination to using perceptual approaches to facilitate the authoring of visual content in the context of image-based representations.

Chapter 4

Scene acquisition

A major prerequisite of image based rendering techniques is the availability of accurate camera parameters for each image in the dataset. This requirement is not an issue when working from synthetic scenes, as all the necessary camera information is readily available, since it has to be specified to obtain a rendering in the first place. At first glance, it appears redundant to use image based rendering techniques to display scenes that could be - or even, have to be - rendered through traditional methods. The utility of such an approach resides in the computation time versus memory usage compromise. As stated earlier, image based rendering times scale with the resolution of the target display medium, as opposed to the complexity of the underlying scene or that of the lighting model used to generate the dataset. Thus, image based techniques are typically used to display walk-throughs of complex realistically lit scenes in real time, at the expense of memory usage. A ubiquitous image based approach of this kind is the practice of “baking” texture maps that incorporate pre-computed light simulation when authoring content for interactive applications such as video games.

However, image-based techniques are attractive primarily because they allow the user to bypass most of the painstaking modeling involved in traditional rendering approaches. This saving can only occur through the accurate sampling of the light field of a real world scene. In the background section, we presented a number of approaches proposed in the literature.

Here we provide more detail on the solution that we adopted, which consists of using commercially available match-moving software based on structure from motion algorithms to track a hand-held video camera. Like all camera tracking techniques that operate on the captured images of the tracked camera itself, the method fails when the scene lacks strong

features. We describe how we worked around this difficulty. Finally, we present an image based modeling system aimed at facilitating the authoring of a geometric proxy.

4.1 Camera tracking

One of the major advantages of unstructured lumigraph rendering is that the views that constitute the input dataset do not have to be regularly positioned or oriented. It is thus possible to directly use images taken with a hand-held camera, provided there is a way to recover each camera pose. In terms of authoring, not having to rely on cumbersome gantries or camera arrays is a big advantage.

4.1.1 Scenes with suitable features

In recent years, camera tracking techniques based on structure from motion algorithms have come to maturity and are commercially available. Products such as 2d3's Boujou, RealViz's MatchMover or The Pixel Farm's PFTrack are widely used in the movie, advertising and broadcasting industries. They allow for the accurate recovery of poses from a hand-held video sequence.

Structure from motion algorithms operate by identifying scene features and matching them from frame to frame as they flow according to camera motion and parallax. The 2D position of each feature in each image plane can be expressed as a function of that feature's position in the 3D scene and the parameters of the camera. Since features appear in many frames, but do not change position from frame to frame, the resulting system of equations is largely over-constrained. Determining the camera parameters thus reduces to an error minimisation problem on that system.

There are two main difficulties in this process. The first is in the identification and matching of features that actually do not change position. In a real world scenario, some objects in the scene might exhibit view-dependent phenomena such as specular highlights, others might be moving. Features belonging to those objects will not flow on the image plane in a direct relationship with the camera's movement. They will therefore throw off the minimisation algorithm. The difficulty here is to discriminate during minimisation which features are outliers and which are not, so that the former can be discarded. The second difficulty resides in the non-linearity of the system of equations. The relationship between the position of the

features in space and their position in the image plane is indeed projective. This makes the minimisation problem much less straightforward to solve.

4.1.2 Problematic scenes

As outlined above, there are composition criteria that will make a scene unsuitable for accurate camera tracking using structure from motion techniques. They are:

1. Lack of contrasting patterns.
2. Preponderance of view-dependent lighting phenomena such as reflections, transparency or specular highlights.
3. Excessive intra-scene motion (as opposed to camera motion).

While 3 is of little concern, as such a video sequence would anyway be of little help in capturing light information fit for image-based rendering, 1 and 2 need to be addressed. Commercial camera trackers help combat 2 and 3 by letting the user define key-framed masks that delimit regions within which features are bound to be unhelpful, such as windows, pedestrians or shiny cars. However, the mask might end up covering such a large area that 1 becomes an issue if there are few strong features in the remaining areas. In the industry, 1 is addressed by having technicians add artificial features to the scene by painting marks on surfaces or placing props. Since the 3D positions of the added features are recovered during the tracking process, they can be composited over in the final shot. In an image-based rendering context, this approach is somewhat contradictory as it amounts to tinkering with the captured light information. Moreover, cancelling the visual contribution of the added features can be quite complex. Added props might introduce shadows and occlusions, while both props and marks might get reflected on other surfaces in the scene.

Thomas *et al* [TJNU97] provided a solution for the BBC's virtual studios. It consists of rigidly attaching a secondary camera to the main camera's body. As the main camera moves, the secondary camera is oriented in such a way that its viewing frustum spans a volume that is distinct from that of the scene of interest. By placing suitable markers in this volume, they ensure that the secondary camera is tracked robustly and accurately. The pose of the main camera can then be derived with a simple rigid transformation. The assumption is made that the intrinsic parameters of the main camera are known in advance or available through sensors.

We propose to use an optical motion tracking device to recover camera poses. Such a choice can rightly be considered at odds with our overall goal of facilitating the authoring of image-based representations, because of the size of the investment that motion tracking devices represent, and because of the deployment restrictions they introduce. However, this choice must be compared to the available alternatives. On the one hand, as discussed above, adding artificial features to the scene could be undesirable. On the other hand, using custom made gantry mechanisms, or setups such as the one deployed in the BBC studio, is expensive and likely to restrict movement severely. We believe that optical motion tracking is maturing very quickly and could soon be available at a fraction of the cost of the devices currently used by the animation industry. Regarding the deployment restrictions, two arguments can be put forward. The first is that large-scale natural or built environments very rarely lack stable features for bundle adjustment methods to track (seascapes are an exception that comes to mind, but they are out of the scope of this work because of their dynamic nature), thus the candidates for capture with our method would likely fit in a studio. The second is that the main obstacle to the deployment of optical motion tracking systems in large-scale outdoors environment is the problem of ensuring that markers are seen properly by the tracking cameras. In our scenario, active markers could be used. The difficulty that would remain lies in the compromise between sensor resolution and motion capture volume, which we investigate.

Optical motion tracking system

Our lab is equipped with a Vicon MX motion capture device. Its *modus operandi* is based on the same computer vision research as that which subtends camera tracking software (Vicon has the same parent company as 2d3: Oxford Metrics Group), as is the case for all optical tracking devices.

The system consists of a number of stationary cameras whose viewing frustums define a capture volume within which passive reflective markers are tracked. Each camera is made of a high resolution and high frame-rate CMOS sensor interfaced with an on-board processing unit that extracts the centroid of visible markers and sends the information over a network interface. An array of infrared LEDs is attached to each camera to ensure optimal visibility of the markers, which are made of 3M ScotchLite material. The cameras are registered with one another by waving a calibration object made of three collinear and rigidly spaced

markers within the capture volume. This produces an over-constrained system of equations linking each camera's parameters, the markers' 3D positions and their 2D projection on each camera's image plane. Similar minimisation techniques as those involved in structure from motion algorithms are used to solve it.

Once the camera parameters are known, the 3D position of individual markers can be tracked through simple projective geometry from the position of their centroids on the image plane of each camera that sees them.

The Vicon is able to automatically label markers in real time. This is done by exploiting a description of each marker's spatial relationship to the others provided by the user. This skeleton can be as simple as fixed relative marker positions, in the case where the object tracked is a rigid body. For more complex subjects, such as humans, the skeleton consists of segments linked by joints of different types (ball, hinge etc.), along with angular constraints, and the markers' positions relative to the segment they track are defined with degrees of freedom. This allows for a generic skeleton to be calibrated so that it fits different individuals.

Numerical feasibility

In our case, the object being tracked is a video camera and can be modeled as a rigid body. Before implementing the system, we analyzed the numerical feasibility of the task at hand. For the tracking to have any use in the context of image-based rendering, the main criterion is that the re-projection error for each camera position be less than a pixel. For a given feature, the re-projection error is defined as the distance between its projection on the image plane using the recovered camera parameters and its position in the captured image. If this measure exceeds one pixel, then there is no guarantee that two pixels in two different views sample the same pencil of light even though the parameterization says that they should. As a consequence, when light information from both views is used to generate a novel view, blurring and ghosting artifacts (if the re-projection error is high) occur.

Our Vicon system is equipped with cameras of the MX13 model. According to their specifications their sensor has a resolution of 1280×1024 pixels and the on-board processing chip yields a 2D accuracy of 0.02 of a pixel, which gives discrimination powers of 1 : 64000 horizontally and 1 : 51200 vertically. We did not carry out experiments to validate this claim from the manufacturer. However it seems plausible to us for two reasons. The first is that each CCD cell samples the light intensity with 10 bits precision. The second is that the

accuracy in question is with respect to the centroid of a disk made of many pixels: the projection of a marker on the image plane. We can imagine that a circle fitting algorithm applied to a 10-bit signal could yield such sub-pixel accuracy. The cameras are fitted with various types of lenses, some of which have adjustable focal length and others not. The average focal length used in the ISG setup is 17mm , combined with the MX13's sensor dimensions of $15.36 \times 12.29\text{mm}$, and assuming that the lenses produce an image circle large enough to cover the sensor, we can compute the field of view using the usual formula $\theta \approx 2\tan^{-1}(\frac{d}{2f})$ where d is either the width or the height of the sensor, depending on whether horizontal or vertical field of view is desired. This gives an average field of view of approximately 48° horizontally and 40° vertically. Dividing by 64000 and 51200 respectively yields angular definitions of $7.5 \times 10^{-4}^\circ$ horizontally and $7.8 \times 10^{-4}^\circ$ vertically. To be conservative, we assume an angular definition of $\theta = 7.8 \times 10^{-4}^\circ$.

As illustrated in Figure 4.1, given the Vicon's angular definition and the maximum distance D of its camera to any point within the capture volume, it is possible to determine an upper bound r to the discrepancy between a marker's actual position and the reconstruction. It is the radius of the rough sphere carved by the intersection of light pencils of angle θ .

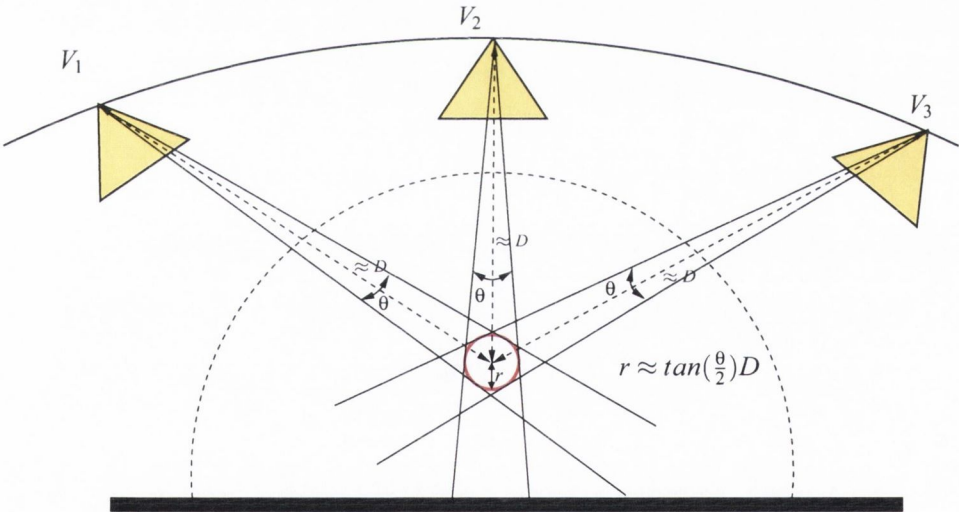


Figure 4.1: Error volume of the Vicon as a function of distance.

The tracked camera is a Canon XL1 DV camcorder. It is equipped with its original lens, which provides a horizontal field of view of 47° at its widest angle, according to the manufacturer. The camera grabs frames in PAL format, corresponding to a resolution of 720×576

pixels. This means that each pixel of the image covers a horizontal arc of approximately $\phi = 6.5 \times 10^{-2}^\circ$. Of course the light that contributes to a pixel takes the form of a conic volume. If we make the assumption that the horizontal and vertical arcs are of similar angle ϕ and that the cone has a circular base, we can express the solid angle seen by a pixel using the usual formula: $2\pi \left(1 - \cos\left(\frac{\phi}{2}\right)\right)$. However, since this relationship is bijective on $[0, \pi]$ and we are only interested in an error threshold, we can limit the study to one-dimensional angles.

In order to estimate whether the Vicon is able to track the camera with sufficient accuracy, we divide the re-projection error into two terms: one resulting from camera orientation error, and one resulting from camera translation error.

Camera orientation error The re-projection error caused by erroneous camera orientation is best analyzed separately in two parts: that which originates from misalignment of the viewing direction, or pan, and that which is caused by erroneous rotation of the camera around the viewing direction, or tilt.

- Pan: as computed previously, pixels on the camera sensor capture light emitted by the captured scene within an arc of roughly $6.5 \times 10^{-2}^\circ$ in both dimensions from the camera. Consequently, for such an arc of light to stop contributing altogether to a pixel, the camera needs to be rotated by that amount. In other words, the re-projection error is higher than one pixel if the recovered orientation of the camera differs by more than $6.5 \times 10^{-2}^\circ$ from what it was when the picture was taken.
- Tilt: given the resolution of the camera sensor, it is possible to estimate a rough lower bound of the tilt angle ρ that will cause the light initially captured by a pixel to fall entirely away from it. From Figure 4.2, we can write, assuming that the pixels are square:

$$\rho > 2 \tan^{-1} \left(\frac{\sqrt{2}}{2\sqrt{720^2 + 576^2}} \right) = 8.8 \times 10^{-2}^\circ \quad (4.1)$$

Since $\phi < \rho$, it appears that the XLI video camera is more sensitive to panning than tilting, . $\phi = 6.5 \times 10^{-2}^\circ$ will therefore be the upper bound of the acceptable camera orientation error.

To determine whether the recovered orientation error falls within the acceptable range, it is expressed in terms of marker position reconstruction error. Figure 4.3 depicts the case of

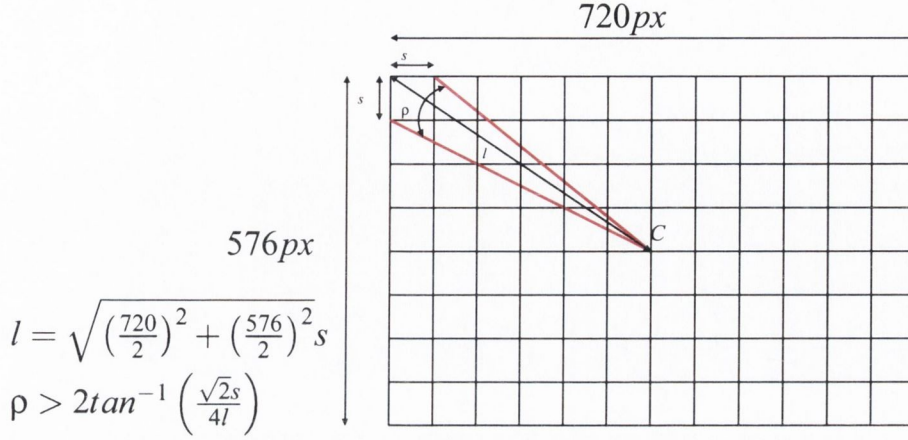


Figure 4.2: Tilt error.

maximum error in 2 dimensions. The further apart markers are, the more accurately can the direction they indicate be recovered. Putting it all together, we can write:

$$\gamma = \tan^{-1} \left(\frac{\tan\left(\frac{\theta}{2}\right) D}{l} \right) < \phi \quad (4.2)$$

Which allows us to express the minimum distance between two markers as a function of the distance between them and the Vicon cameras:

$$l > \frac{\tan\left(\frac{\theta}{2}\right)}{\tan\phi} D \quad \text{or numerically: } l > 6 \times 10^{-3} D \quad (4.3)$$

The dimensions of the space where the Vicon is installed in our lab are $10 \times 7 \times 3$ meters, which means that placing markers 10 centimeters apart is theoretically sufficient.

Camera translation error The reconstructed position of the camera's centre of projection is also subject to error. Figure 4.4 roughly illustrates the relationship between the camera's distance to the captured scene and the re-projection error. h is here the smallest camera translation that will cause light coming from objects at a certain distance d to project to a different pixel. Since the position of the camera's optical centre is estimated from markers, it is subject to the same reconstruction error radius r . We want to ensure that $h > r$, which

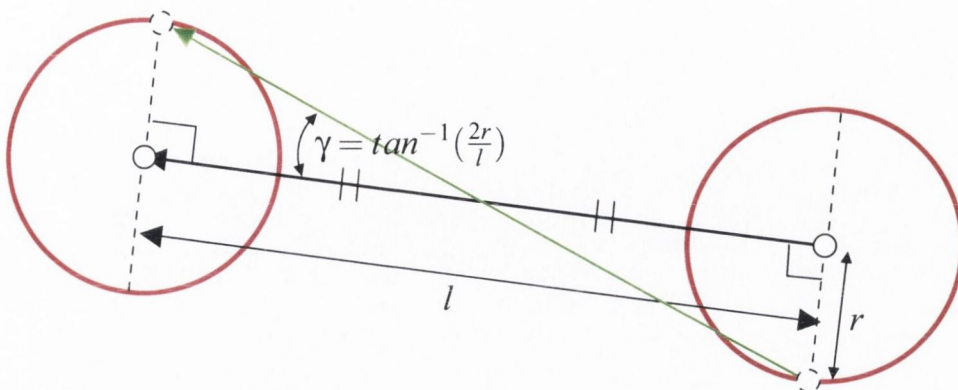


Figure 4.3: Orientation error upper bound.

gives:

$$2 \tan\left(\frac{\phi}{2}\right) d > \tan\left(\frac{\theta}{2}\right) D \quad \text{or numerically:} \quad d > 6 \times 10^{-4} D \quad (4.4)$$

In our lab's setup, this corresponds to a perfectly acceptable minimum distance of 10 centimeters (at which the XL1 camera cannot even focus).

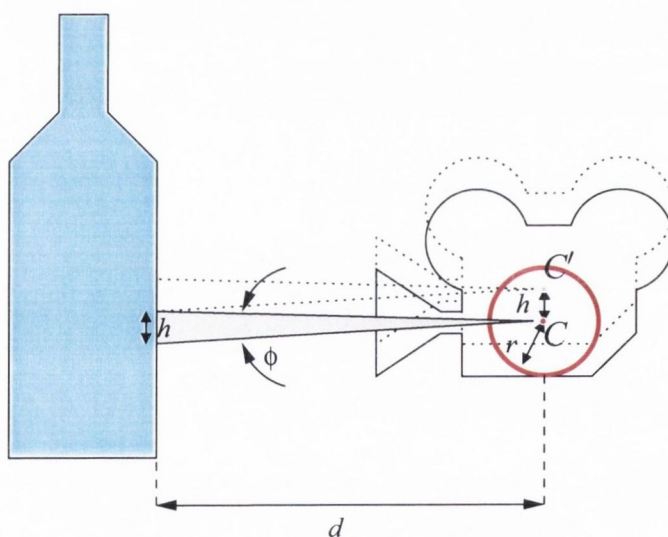


Figure 4.4: Error due to translation.

System

The body of the XL1 camcorder is quite small and intricate, which makes it difficult to attach adequately spaced markers that stay visible from wide angles. To address this, we attached the camera to a custom-built metallic rig. We verified the rigidity of the link between the camera and the rig by measuring the optical drift resulting from applying pressure on the camera. Reflective markers were then fitted to the rig. It is possible to describe the orientation of a rigid body by defining a local vector basis rigidly attached to it and expressing the coordinates of these vectors in a reference frame. In 3 dimensional space, the basis must consist of 3 vectors. Such a basis can be generated from 3 non-collinear points by constructing 2 linearly independent vectors and taking their cross product to obtain the third. Thus, a minimum of 3 markers rigidly linked to the camera would be sufficient for the Vicon to track its orientation, provided they always remained visible. However, in practice markers get occluded during capture. For the sake of robustness, we used 5 markers. The markers' relative positions were captured using the Vicon. From there, a skeleton was created using the Vicon modeling software, it consists of a simple rigid body whose local coordinate system is rigidly linked to the markers, which means the skeleton is pre-calibrated. This is important, as letting the Vicon perform the calibration from a generic skeleton for each capture would make it impossible to register once and for all the camera parameters to the rig's coordinate system. The rig is shown in Figure 4.5.

With the calibrated skeleton, the Vicon is able to track the rig in real-time. It broadcasts the rig's position and orientation over TCP/IP. We wrote a network application to translate the Vicon information into camera parameters on the fly. For this, the camera had to be calibrated and accurately registered to the rig's coordinate system.

Calibration

To calibrate the camera and establish the relationship between its pose and that of the rig, data had to be acquired linking actual camera poses to the corresponding rig positions and orientations as recorded by the Vicon. The data was then fed to a minimisation algorithm to estimate the parameters of the relationship, of which a model had been made.

Model The main difficulty in formalizing a model expressing the relationship between the camera's and the rig's poses is that there is no guarantee that the data recovered for each is



Figure 4.5: The camera rig.

expressed in the same reference coordinate system. Thus, the model has to incorporate the change of coordinate system as an unknown to be solved for just like the parameters that link the camera's pose to the rig's.

Let $\mathcal{B}_{C \rightarrow V}$ be the change of coordinate system (written in homogeneous notation) from that in which the camera's pose is expressed to the Vicon's. Let $\mathbf{P}_{Rig}$ be the position of the camera's centre of projection expressed in the rig's local coordinate system. Those are our unknowns. Through measurement, we have access to:

- The change of basis from the rig's local basis to the Vicon's: $\mathcal{R}_{Rig \rightarrow V}$.
- The origin of the rig's local coordinate system, as expressed in the Vicon's: $\mathbf{O}_V$.
- The camera's projection centre, as expressed in the camera calibration coordinate system: $\mathbf{P}_C$.

We know this relationship to hold:

$$\mathcal{B}_{C \rightarrow V} \times \begin{bmatrix} \mathbf{P}_C \\ 1 \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{Rig \rightarrow V} \times \mathbf{P}_{Rig} + \mathbf{O}_V \\ 1 \end{bmatrix} \quad (4.5)$$

We would like to be able to reformulate this into a simple linear equation system that we could then solve in the least squares sense. Each row of the system would express a constraint imposed by our measurements. Let us start by only leaving a known quantity on the right hand side:

$$\mathcal{B}_{C \rightarrow V} \times \begin{bmatrix} \mathbf{P}_C \\ 1 \end{bmatrix} - \begin{bmatrix} \mathcal{R}_{Rig \rightarrow V} \times \mathbf{P}_{Rig} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{O}_V \\ 1 \end{bmatrix} \quad (4.6)$$

If we write:

$$\mathcal{B}_{C \rightarrow V} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & t_x \\ b_{21} & b_{22} & b_{23} & t_y \\ b_{31} & b_{32} & b_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{R}_{Rig \rightarrow V} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}$$

Then by developing the matrix products it is possible to show that 4.6 is equivalent to the following:

$$\begin{bmatrix} [\mathbf{P}_C^T & 1] & [0 & 0 & 0 & 0] & [0 & 0 & 0 & 0] & [-\mathbf{r}_1] \\ [0 & 0 & 0 & 0] & [\mathbf{P}_C^T & 1] & [0 & 0 & 0 & 0] & [-\mathbf{r}_2] \\ [0 & 0 & 0 & 0] & [0 & 0 & 0 & 0] & [\mathbf{P}_C^T & 1] & [-\mathbf{r}_3] \end{bmatrix} \times \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ t_x \\ b_{21} \\ b_{22} \\ b_{23} \\ t_y \\ b_{31} \\ b_{32} \\ b_{33} \\ t_z \\ \mathbf{P}_{Rig} \end{bmatrix} = \mathbf{O}_V \quad (4.7)$$

Each measured camera pose yields 3 equations. The system can therefore easily be over-constrained by collecting data for a number of poses. What is left to compute after this is the change of basis $\mathcal{R}_{Rig \rightarrow C}$ from the rig's to the camera's. Thanks to our knowledge of $\mathcal{B}_{C \rightarrow V}$, we can express the vectors that make up the camera's local basis in the Vicon's coordinate

system, in which the rig's local basis is expressed. From there it is trivial to write a linear system to solve for $\mathcal{R}_{Rig \rightarrow C}$.

We wrote a program that takes as input a collection of pose information and generates an Octave [Joh] script that sets up and solves the resulting linear systems. The script also contains a model validation function that computes the average position and orientation uncertainties by comparing each camera pose as reconstructed from the rig's by the model to the actual measurement.

Data acquisition We initially recovered camera poses using the respective works of Zhengyou Zhang [Zha00] and Jean-Yves Bouguet [Int]. Both recover camera information from a small set of pictures of a calibration object containing sharp features whose positions are precisely known. We created such an object and took pictures of it from different stationary positions, using the photo feature of the XL1 camera. For each picture, the position of the rig was recorded with the Vicon. Using the OpenCV computer vision library [Int], we wrote a program to automatically label the calibration object's features and extract their position in the image plane with sub-pixel accuracy. Figure 4.6 shows a view of the calibration object and the numbered recovered features. Zhang and Bouguet have both made their camera calibration code available to the public. We ran both on several sets of collected feature correspondences, and fed the resulting poses, along with the Vicon data, to our camera/rig relationship solver. We were never able to obtain uncertainties of less than a tenth of a degree for camera orientation and less than 5mm for camera position. As discussed above, these figures do not match the accuracy required to ensure sub-pixel re-projection errors.

Given the proven precision of the Vicon system, the only probable cause of the low accuracy was a camera calibration and pose estimation problem. It is possible that our lack of results was due to our inexperience with camera calibration: we might not have made sure that the calibration object occupied most of the picture in each pose, which could lead to an improper modeling of lens distortion, and we might not have acquired pictures that covered a sufficient variety of angles. Ultimately, calibration techniques operate on feature re-projection errors. It is a well known fact that, given a set of real world features and a picture of them, it is possible to obtain negligible re-projection error over an interval of compromises between small rotations and small translations of the camera [SS00]. Michaels [Mic92] has studied that ambiguity in detail and formalized a relationship between the range of the compromise interval and the field of view of the camera. He has shown that it is only

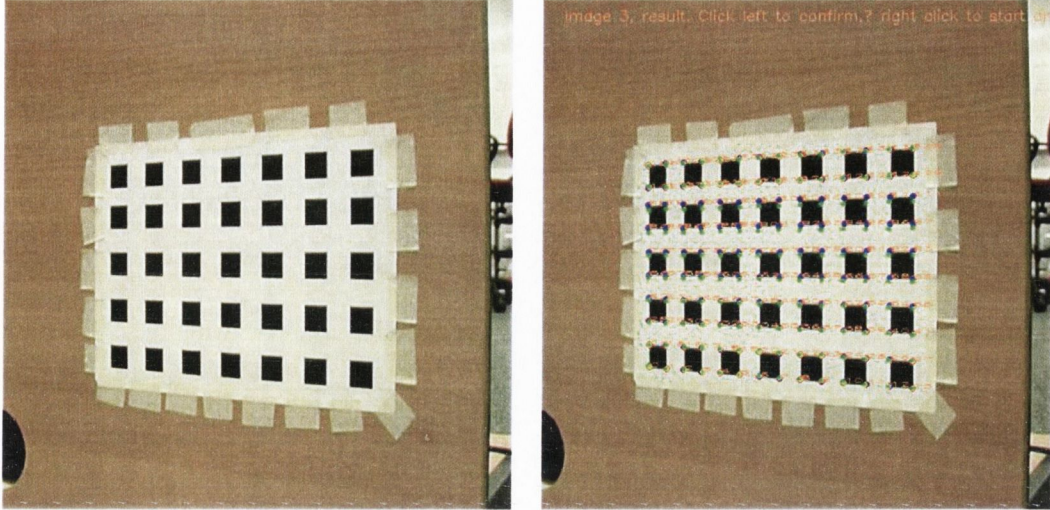


Figure 4.6: Calibration object and recovered features.

for field of views of 120° and higher that the ambiguity becomes negligible. Our camera, with its widest field of view of 47° , falls short. The purpose of Michaels' PhD thesis was to study whether calibration difficulties such as this one could be alleviated by taking time continuity into consideration. The success of current commercial structure from motion trackers is a testimony to the validity of this approach.

We therefore decided to use PFTrack from The Pixel Farm [The]. Working from video sequences, as opposed to sets of still pictures, introduced the problem of synchronizing the video stream to the Vicon's capture rate. Neither the XL1 camera nor our lab's Vicon installation allowed for hardware synchronisation. However, the Vicon is able to capture motion at a steady rate of 100Hz . This means that given a trigger event recorded simultaneously by the Vicon and the camera, the maximum time discrepancy between when the markers were captured and when the frame was grabbed is $\frac{1}{100}s$. According to the figures quoted in the numerical feasibility study, sub-pixel re-projection is guaranteed for angular velocities lower than $6.5^\circ s^{-1}$ and translation speeds of 0.12 m.s^{-1} , *i.e.* 0.34 m.s^{-1} when shooting from 3 metres. Those figures are somewhat low but manageable. Besides, the interlaced nature of the PAL format makes shooting at low speeds necessary to minimise temporal incoherence between half frames.

The trigger event that we use is the moment when the user picks up the camera from a stationary position. It can be easily localized in the stream of video frames as well as that of

marker positions.

4.2 Geometric proxy editing

Most 3D modeling applications let the user display a background image in workspace viewports. If camera poses are available, it is possible to import and keyframe them along with the appropriate background photograph so that they match. One can then obtain a viewport where the edited geometry is registered with the photographs of the scene by displaying camera views. The main inconvenience with this image-based modeling setup is that geometry placed in a viewport that appears to coincide with features of the background photograph can in fact occupy an infinity of positions along the rays of the camera projection for that view. Only by changing viewpoints can the user refine the initial placement. If the different viewpoints used do not have convenient alignment properties (*e.g.* orthogonality), this position refining process converges slowly. This is because changes made using one viewpoint will modify the projections in other viewpoints in a non-intuitive manner. This difficulty is compounded by the fact that changing the viewpoint has to be done using the time slider. No interface that we have tested allows simultaneous browsing of the time slider while moving geometry, leading to a painstaking back and forth between interaction modes. Moreover, the time slider is one dimensional and therefore not an ideal tool to browse through viewpoint positions.

Software that offers image-base modeling features, like Image Modeler [Reaa] or PF-Track [The] take a constraint driven approach to guide geometry editing. They rely on accurate initial features reconstructed by the program. Placing new landmarks can be inconvenient, particularly in regions where no views exhibit strongly contrasted patterns. Their 2D displays can also make the set of initial features (which consists of a vertex cloud) confusing to make sense of.

Many user interfaces have been proposed that combine stereo displays and 3D positioning devices. The work of Fiorentino *et al.* [FdMS02] makes use of motion tracked props and is a good starting point on the topic of 3D modeling interfaces. To the best of our knowledge, we are the first to propose an interface designed for image-based modeling based on these techniques.

4.2.1 Stereoscopic 3D positioning system

Physical setup

The modeling application is displayed on a conventional "GeoWall" passive stereo screen [SDW02]. The display installation thus consists of:

- A non-depolarizing back projected screen (dimensions $200 \times 150\text{cm}$). It is made of Stewart Filmscreen "Disney Black" material.
- Two NEC LT265 2100 lumen DLP projectors.
- A table-top Chief ASU-2000 stacker on which the projectors are mounted. Both shelves are accurately adjustable, allowing for the projected images to be properly aligned.
- Two circular polarizing filters. They are held at a distance from the projectors by protruding stands to spread the high powered beams of light over a larger surface, thus preserving them from premature damage. The filters' polarizing directions are made orthogonal to one another.
- A set of passive stereo glasses. They hold orthogonally polarized circular polarizing filters, ensuring that only the light emitted by one or the other projector reaches each eye.

The stereo effect is simulated by having each projector project images taken from viewpoints corresponding respectively to the position of the left and right eye of a virtual observer. Figure 4.7 shows a photograph of the projection system.

The Vicon system is used for motion tracking. We found that 6 tracking cameras are sufficient for our purpose. They cover a volume large enough to let the user navigate the space in front of the screen for comfortable viewing positions while letting him extend his arms fully. Motion is tracked at a rate of 100Hz , which falls within the range of 60 to 125 hertz that mouse users are used to.

We decided to track props rather than the user's body. The main reason is convenience: it is much easier to pick up a prop than to attach tracking markers to a person. Additionally, it allows us to extend the user interface in the future by letting the user pick-up different props corresponding to different modeling tools. The props that we currently identify and track are

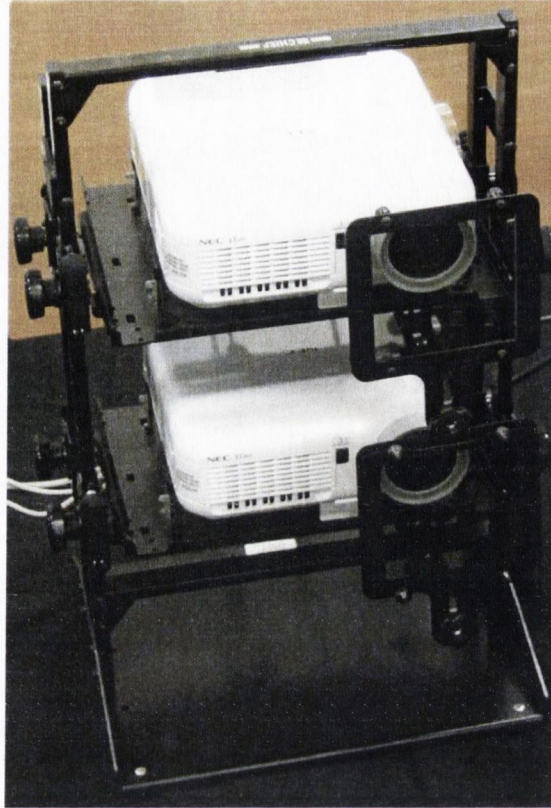


Figure 4.7: The GeoWall projectors and their filters on the adjustable stacker.

the pair of passive stereo glasses and some wooden pincers. Both are shown in Figure 4.8 along with the pre-calibrated skeletons that were modeled so that the Vicon could track them in real time.

A wireless mouse is placed in the user's non-dominant hand and its motion is not tracked. Figure 4.9 shows the setup of our system.

Software

The application consists of two processes running concurrently:

- The proprietary Vicon server which broadcasts the position of tracked objects (in our case, props) in real time to client programs that connect to it.
- Our modeling program, built on top of OpenGL and GLUT, which connects to the

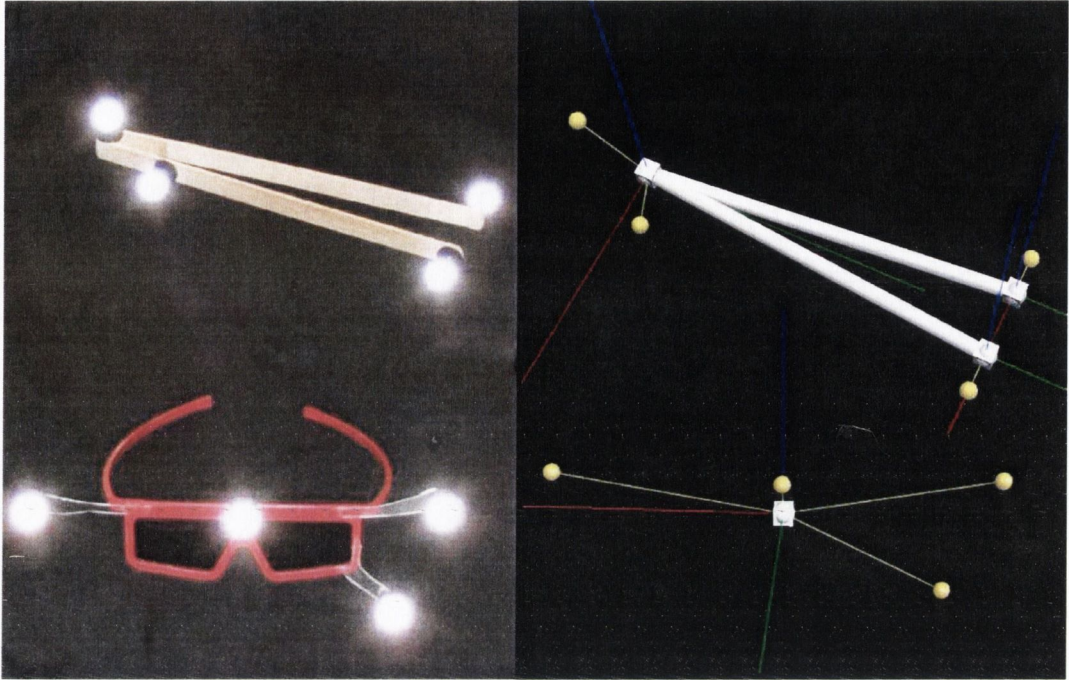


Figure 4.8: Trackable pincers and stereo glasses.

Vicon server through a TCP socket.

The modeling program takes as input a set of photographic stereo pairs as well as the parameters of the camera for each photograph (pose and intrinsic parameters) expressed in Cartesian coordinates. Optionally, a set of scene features, which are produced by most camera pose estimation programs, can be provided as initial vertices. The stereo pairs are assumed to be binocular photographs of the scene that the user wishes to model as seen from different viewing angles. The program behaves mostly like a polygonal mesh editing tool based on triangular faces, with a few key differences:

- The user is not free to navigate arbitrarily around the scene being modeled, but browses instead among the input (binocular) viewpoints.
- The stereo photographs corresponding to the current viewpoint are displayed as background images.
- The geometry being modeled is projected onto the viewport corresponding to each eye using the camera parameters that were used to take the corresponding photograph.



Figure 4.9: System setup. Two of the Vicon cameras can be seen in the background. The bright glares are the reflective markers.

As a result of these features, the 3D cursor manipulated by the user, as well as any geometry created, will appear to coincide with a feature of the modeled scene in both the left and right viewports if and only if their 3D coordinates match (in the metric space in which the camera poses are expressed). When this happens, the binocular disparity cue [CWE94] correctly gives the user the sensation that the cursor or geometry is situated at the same depth as the feature. Figure 4.10 illustrates this property.

User interface

The interface for our modeling application does not use any on-screen menus. We experimented with them but found that the juxtaposition of a 3D workspace and 2D menu was visually confusing for the user. Since we limit ourselves to polygon editing, the combina-

tions of prop states provide enough flexibility to cover all necessary features.

The pincers are held in the dominant hand. Their position as well as orientation is tracked, providing a 6 DOF cursor. The interface can detect whether the user is pinching or not. It also takes into consideration whether the cursor lies within the frustum of the current binocular camera. The wireless mouse is held in the other hand. Its buttons are used in the same way as the augments keys (shift, control, alt) are in a keyboard based user interface. The stereo glasses are only tracked for orientation. Tracking their position would indeed be counterproductive in this context for several reasons. The first is that the stereo effect works best from specific head positions with respect to the screen. The second is that non-depolarizing screens tend to have a reduced half-gain interval, this is the range of viewing angles outside of which projected light is reproduced below half intensity. Finally, translating one's head often requires moving the whole body, so the user could therefore not sit, and long editing sessions could become taxing.

We chose to implement an interface close to Blender's vertex edit mode [Ble]. One difference is the use of a helper to constrain transformations to desired axes or planes. It is displayed at the barycentre of the set of currently selected vertices.

Since different modes of interaction are used to manipulate the editing tools and the viewpoint, it is very fast and convenient to visualise the consequences of editing changes from different views. Our system exploits epipolar geometry to facilitate the placement of vertices at scene features. This is done by overlaying the epipolar line of the vertex being manipulated and updating it as the user browses viewpoints. By placing a vertex at a feature, browsing to a different view, and sliding the vertex along the epipolar line until it matches the feature's position in that view, the user can very quickly register new vertices to scene features. Table 4.1 summarizes the available operations.

4.2.2 Stereo pairs acquisition

Camera tracking software is the most convenient way of recovering camera poses that are consistent with each other within a metric space. However, such programs expect continuous video sequences to function robustly. This implies that, assuming a binocular stereo camera is available, the footage corresponding to one eye has to be tracked independently from that of the other, which leads to difficulties registering the two sets of poses with each other. Moreover, cameras capable of filming continuous stereo footage are uncommon and expen-

sive, and synchronization issues make building one somewhat difficult. We chose to take a single video sequence with a standard monocular camera, and extract satisfactory stereo pairs from the set of tracked poses. This was done by implementing a "stereo suitability" function: given a pair of poses, it computes a grade based on how close they are to being separated in the lateral direction by the typical distance between human eyes (≈ 7.4 cm), and how similar their orientations are. This approach places constraints on the video sequence used, which must consist of lateral translations that are slow enough to contain pairs of poses whose separation is close to 7.4 cm.

In practice, we attached the camera to a bicycle that we slowly pushed around the scene. Pose recovery was done using PFTrack. Its scene scaling feature allowed us to recover real world 3D coordinates for the camera poses from a single distance measurement between two identified features in each scene.

4.2.3 Discussion

Our system showed good potential to facilitate the modeling of existing scenes. The main gain is that it makes positioning vertices that match scene features much easier, thanks to the depth cue and the possibility to easily change viewing angle while placing a vertex. Displaying in stereo also makes it easier to edit faces, because vertices that would appear in a confusing cloud on a 2D display can be discriminated by depth.

In its current state, the system suffers from some limitations. One is that the editing options provided are very limited. We are working on adding primitive creation and manipulation through the use of additional props. In the longer term, curve and surface tools could be added. To extend the palette of actions that the interface can handle, we plan to implement a menu mechanism similar to Maya's hotbox: a temporary menu would be displayed in 3D within the workspace at the cursor's position when a button is held down. The user could then pick an action with the cursor. Another limitation is that the only depth cues [CWE94] that the system provides are binocular disparity, motion parallax and foreshortening. Improperly conveyed vergence, accommodation, and to a much greater extent occlusion cues can confuse the user when they blatantly contradict the sensation given by binocular disparity. The display device being a flat surface, there is no hope of improving vergence and accommodation cues. However, the use of range information (even inaccurate, provided it is conservative) combined with a depth buffer could greatly improve positioning ease.

From a prospective standpoint, one could argue that a system such as ours represents a significant investment. The problem does not lie with the stereo display, as these are already very affordable, but with the motion tracking device. We have stated earlier that we anticipate that such devices will be democratised in the short to middle term. It must also be said that we used the Vicon system because it was available and allowed us to easily experiment with different modes of interaction. The only essential feature of the interface we propose is the ability to position a cursor in three dimensions, this is achievable with a range of affordable 3D input devices.

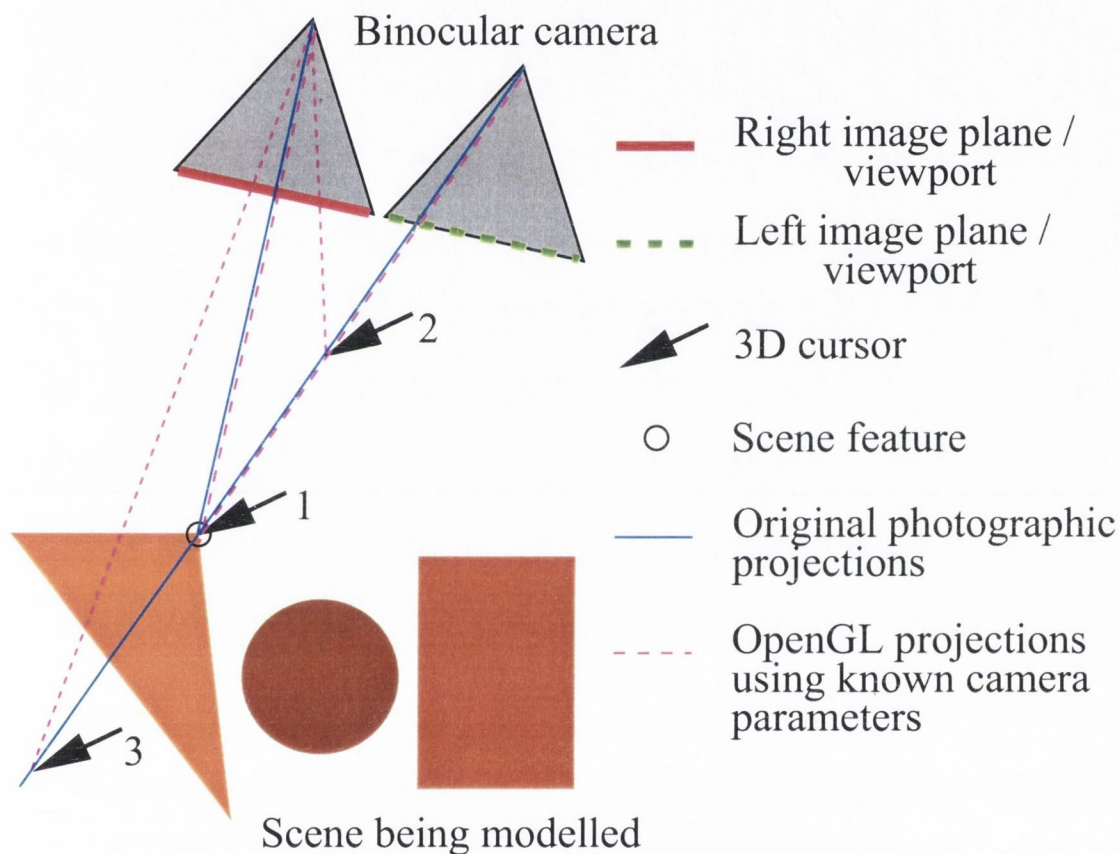


Figure 4.10: Top view of a scene being modeled with one binocular viewpoint. Position 1 of the 3D cursor coincides with that of a feature and thus gets projected to the same point as that feature in both viewports. Positions 2 and 3 differ from that of the feature: even though their projections coincide with that of the feature in the left viewport, they do not in the right viewport. Thus the user correctly perceives them as misaligned.

Operation	Means
Select*	Place the cursor on a vertex and pinch.
3D Box select*	Place the cursor at the first corner of the box, pinch, drag while still pinching to the opposite corner, then stop pinching.
Move	Place the cursor on a vertex of the current selection, or on an axis of the transformation helper, pinch, drag while still pinching, stop pinching when at destination.
Duplicate	Same as Move but with the left mouse button pressed down.
Delete	Same as Move but stop pinching when the cursor is out of the frustum.
Undo	Start pinching on an empty space, drag the cursor until it is out of the frustum, then stop pinching.
Redo	Move the cursor out of the frustum, start pinching, drag the cursor until it is back inside the frustum, then stop pinching.
Browse viewpoints	Hold the right mouse button down and rotate the head slightly in the desired browsing direction, release when the desired viewpoint is reached. The 3D cursor's position is frozen for the duration of viewpoint browsing. It can therefore be used to update the cursor's relationship to the user's hand position, letting him reach other parts of the scene more comfortably. It also helps the user ascertain very quickly whether the cursor is at the desired location by observing how it moves with respect to the scene when viewed from a different angle. Alternatively, if the user was manipulating a vertex, the epipolar line corresponding to its position is overlayed as the viewpoint changes. The user can move his right hand to slide the vertex's position along the epipolar line.
Create face(s)	Press the middle and right mouse buttons simultaneously. The current selection needs to contain at least three vertices. If more than three vertices are selected, a triangle strip is created based on their order of selection.

Table 4.1: List of available operations and the means by which they are invoked. *Selection operations are modified by the mouse buttons: if the left button is down, the new selection is added to the current one. If the middle button is down, it is removed.

Chapter 5

View selection framework

In Chapter 2 (Section 2.2.4), we explained the trade-off that unstructured image-based techniques make in favour of capture and authoring ease at the expense of straightforward exploitation of inter-view visual redundancy for compression purposes. This chapter presents the perceptual framework that we propose to facilitate the selection of a set of input views that minimises visual redundancy.

In Section 5.1, we describe the design of our framework. This is followed by implementation details. We then explain how our technique capitalises on the savings incurred by discarding views.

5.1 Design

Given a perceptual measurement tool that evaluates how similar an image is to a reference, it is straightforward to define the perceptual quality 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 image-based rendering (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 noting the difference in perceptual quality of renderings with and without it. The lower that degradation, the less view dependent information that input view captures. Which means that it is more redundant than views whose removal would cause a higher degradation and should therefore be pruned before them. For clarity, we will call the views used for the purpose of computing perceptual quality *touchstone views* (TV). Figure 5.1

illustrates the idea of aggregate fidelity metric.

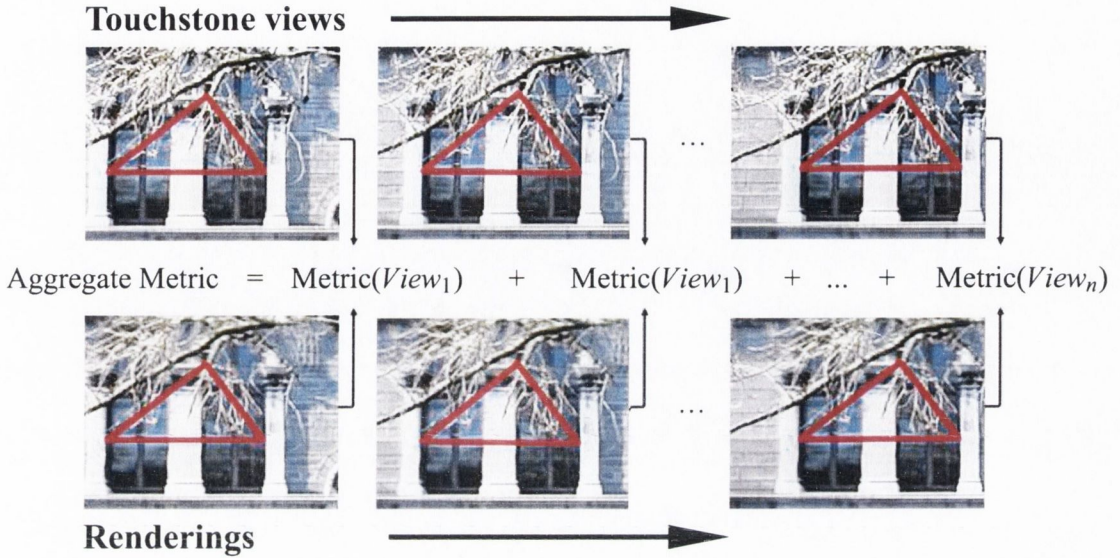


Figure 5.1: Aggregate fidelity metric over a set of n input views, called touchstone views. Each touchstone view is compared to a rendering from the same viewpoint using an image fidelity metric. The fidelity scores are summed to yield the aggregate score. (The red triangles correspond to the projection of a face of the geometric proxy in each view.)

From a general standpoint, if the perceptual quality of a subset of views cannot be predicted either from *a priori* knowledge of the scene or from the scores of previously explored subsets, then each one has to be explored in order to determine the best. If the number p of views to retain is known in advance, then nC_n^p evaluations of the perceptual measurement tool are necessary, where n is the number of initially available views. Assuming hundreds of initial views and an optimistic target of an order of magnitude lower number of selected views, this is clearly intractable.

We show how the unstructured lumigraph rendering algorithm makes it easy to exploit spatial coherence to speed up the computation of the measurement 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. Finally, we provide two view selection algorithms: one operating on whole views, the other operating at a sub-view level.

5.1.1 Context: Unstructured Lumigraph Rendering

The original unstructured lumigraph rendering (ULR) algorithm was introduced in the background section. One of its features is of particular interest for our purposes: *i.e.* the fact that it computes, for each vertex of the triangulated blending field, the list of k input views whose blending is non-null over it.

For a given touchstone view, it is thus 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 (*Cf.* red triangles in Figure 5.1).

Such a strategy is of little use when the entire captured scene is visible from each input view. However, this is not likely to be the case under real world constraints, *e.g.*, capturing the facade of a building from a narrow street.

5.1.2 Perceptual measure of view utility

For our framework, we considered three visual fidelity metrics: the traditional root mean square error (RMS), Scott Daly's Visible Differences Predictor (VDP) [Dal93], Yee and Newman's [YN04] PerceptualDiff (PDIFF) and Wang *et al*'s [WBL02] Structural SIMilarity index (SSIM). We made the choice not to consider chromatic information at first. The metrics thus operate solely on the luminance channel.

VDP

Karol Myszkowski kindly provided us with an implementation of the VDP that we used to experiment with. The VDP's functioning, described earlier (Section 2.3.3), relies on several transformations of the signal, including a passage through the frequency domain, which thus makes it very computation intensive. Within our view selection framework, it is possible to pre-compute half of the necessary transforms, because the first half of the compared visual signals (that contained in the touchstone views) remains constant. Even though the pre-

computed data structures (*e.g.*, spatial frequency maps for each touchstone view) can take up much space, the unstructured lumigraph algorithm and our framework naturally preserve locality, meaning that when considering a specific input view, the re-evaluation of the metric only needs to be recomputed for neighbouring touchstone views. In spite of these savings, we could not get our framework to order views using the VDP in a reasonable amount of time for typical real-world scene size scenarios (hundreds of views, hundreds of triangles).

SSIM

Contrary to most perceptual metrics, the SSIM does not rely on a simulation of the low level behavior of the Human Visual System (HVS). It is based on the observation that the function of the HVS is to extract structural information from visual stimuli. It therefore estimates how similar two images are by using statistical tools to quantify the structural difference between them. As a result, it presents the following features that led us to choose it:

- As Wang *et al* show, when predicting the visual fidelity of a wide range of images, their method can compare favourably to RMS or PSNR, and more importantly, techniques based on models that reproduce the contrast sensitivity of the HVS.
- Spatial frequency masking is a HVS property accounted for by those other techniques, unlike 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 on the object.
- The SSIM index is less computationally intensive because it takes a statistical approach as opposed to a signal processing one, which requires complex transforms to be applied.
- It has little computational overhead, an important advantage if we want to evaluate it on many small triangles and not just a few whole images.
- It is straightforward to implement as a multi-pass fragment shader.

PDIFF

Yee and Newman's [YN04] (PDIFF), which they put forward in the context of production testing, is based on Ramasubramanian *et al*'s [RPG99] simplified version of the VDP. Like the VDP, it accounts for three features of the HVS: amplitude non-linearity, 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, based on Laplacian pyramids, at the cost of a much more rudimentary modeling of the visual masking phenomenon (because interactions between signal components based on frequency and orientation similarity are not considered). The behaviour of PDIFF depends on the field of view occupied by the signal and its resolution, which depend on the target viewing conditions: cinema theatre in Yee and Newman's case, desktop monitor in our user study.

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 normalized score for each pixel, PDIFF's output consist 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.

5.1.3 Selecting Whole Views

In this first version of our framework, we use a greedy approach to iteratively remove whole input views from the initial set. At each step, the discarded reference view is the one whose removal causes the least perceptual degradation, as measured by the aggregate visual fidelity

metric over all touchstone views. Brief pseudocode is given in Algorithm 5.1.1.

Algorithm 5.1.1: VIEW ORDERING()

Compute the initial image fidelity metric scores with all input views in use
while there remain input views
 do {
 for each remaining input view
 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 input view
 Append the input view whose removal causes the least
 degradation to the list of ordered input views.
 Remove that input view from the list of remaining input views.
 }

Some book-keeping is necessary to avoid recomputing the visual fidelity metric on triangles that were not affected by the last view removal. An array contains the initial metric scores 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. We update the following data structures after each removal:

- A1 A cache containing the visual degradation, in the form of a metric score difference, over each triangle in each touchstone view for each primary view to be potentially next removed.
- A2 A cache containing the next visual degradation resulting from the potential removal of each view.
- L3 The list of remaining views.
- L4 The list of remaining input views ordered by ULR penalty for each vertex in each touchstone view.
- L5 The sub-list 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 view contributes.

A1 only needs to be updated when the last removed view “contained” that triangle. A2 is obtained by averaging the visual degradations of all triangles of a view over all relevant touchstone views. It gets updated if the removal of the last view affected the metric score 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 visual fidelity metric on a touchstone view, while using that same 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 metric score to increase all the more if their weight was important, because the relative weight of the touchstone view itself would be increasing. Such behaviour 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 metric score over a triangle resulting from the potential removal an input view that affects it, we need to render the triangle without that view. Consequently, the $(k + 1)^{th}$ input view that will fill the gap for each vertex where the removed view was present needs to be known.

When a view is selected for removal, other views 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 view. L6 stores that information and it is therefore updated by identifying which input view “filled the gap” left by the removal of the last view for each affected triangle in each touchstone view.

The output of the algorithm is a list of all initial reference views sorted in the estimated order in which they should be removed to minimize perceptual degradation.

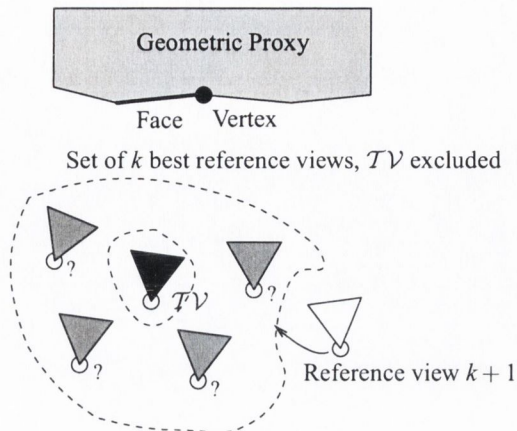


Figure 5.2: At a given removal step and touchstone view $\mathcal{TV}$, for each vertex, the $(k+1)^{th}$ best reference view is needed, otherwise we would be left with only $k-1$ reference views to blend at that vertex when rendering the face after having removed a candidate view (marked with a ?). Note that $\mathcal{TV}$ is excluded from the set of k best views.

5.1.4 Selecting at a Sub-View Level

The functioning of the unstructured lumigraph algorithm allows for a finer visual information granularity than the whole view level. 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. The second instance of our framework uses this property to perform the selection on areas of the initial input views.

In the original version of the algorithm, no straightforward relationship can be defined between the triangles that share a vertex and an area in a specific input view. This is because the triangles and vertices in question are touchstone view dependent, as a result of the image plane triangulation step. We therefore decided to skip this step altogether.

We first had to make sure this choice would not compromise the properties of the rendering technique.

The reason why image plane triangulation was introduced in the first place is because of blending field sampling regularity. 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 was needed at the lower end of that spectrum in order to generate additional points at which to sample the blending field given a desired view.

Buehler *et al.* [BBM⁺01] do not discuss an 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 5.3). This can be solved, albeit slowly, by reading back from the depth buffer after having drawn the geometric proxy in a first pass.

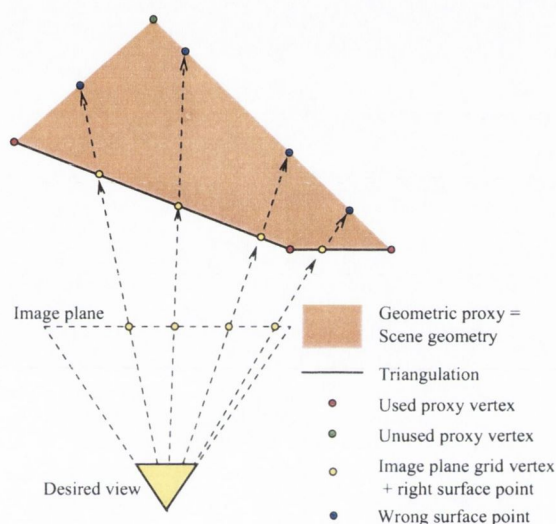


Figure 5.3: The depth of an image plane grid point has to match that of the closest surface.

The second problem 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 accurately describes the scene, or it does not. If it is, simply triangulating the points yields a surface that departs from the actual scene geometry, as illustrated in Figure 5.4. 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 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 the actual geometry, the discontinuity introduced by that approach would produce a rendering artefact, instead of reproducing a feature of the scene (See Figure 5.5). Thus, without a way to discriminate between accurate

and un-accurate edges, triangulation results in an unresolvable situation.

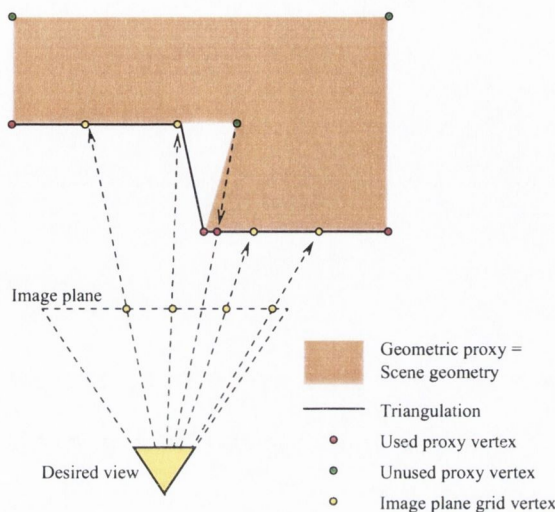


Figure 5.4: Erroneous triangulation due to an overlap of the geometric proxy in the desired view.

The paper [BBM⁺01] 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. The only extra cost 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. In practise, we found that this was counterbalanced by the savings incurred by scrapping the triangulation step.

The view areas in question consist of the projection of the triangle ring surrounding each vertex of the geometric proxy into each input view. We call them *sub-views*. Straightforwardly, each sub-view is identified by the input view it belongs to, along with a vertex of the geometric proxy (the centre of the projected triangle ring). Figure 5.6 illustrates the concept of sub-view.

There is in fact no need to reformulate our view-selection framework to accomodate the concept of sub-view, the explanations given in the previous sub-section (5.1.3) still hold by simply substituting “sub-view” for “input view” in the text.

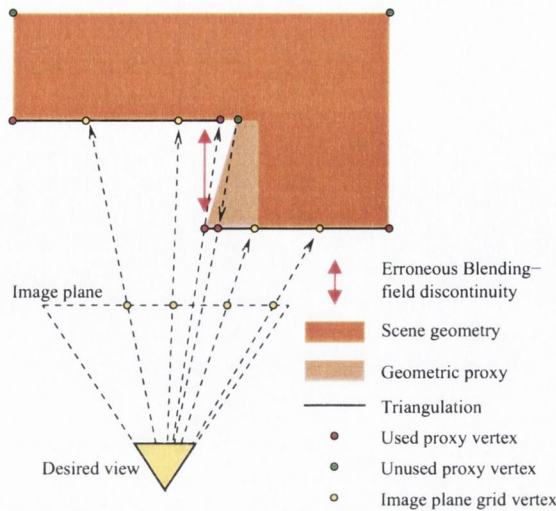


Figure 5.5: The geometric proxy does not match the scene’s geometry. As a consequence, introducing a discontinuity in the triangulation causes a discontinuity of the blending field that does not match the scene, yielding a visual artefact.

The greedy pruning process now outputs a list of all initial sub-views sorted by the estimated order in which they should be removed to minimize perceptual degradation.

5.2 Implementation

We implemented our framework in Haskell [JHH⁺] using HOpenGL, the available OpenGL and Glut binding. There were numerous motivations for this choice.

The first is that our experience with programming in a high level functional language had shown us that it provides great ease of prototyping. This is because such languages are designed to be more “abstraction friendly” than others, which makes it easier to “code as one thinks”, as opposed to contorting one’s thought to fit a rigid imperative philosophy. Haskell’s list comprehension mechanism is a good example of this and here are two lines from the code:

```
camList <- readArray blendableViewsPerVertexArray vertexIndex
let blendableCamerasAssocs = [(i,camerasArray!i) | i <- camList]
```

very intuitively fetches a list of the indices of the cameras whose images can be

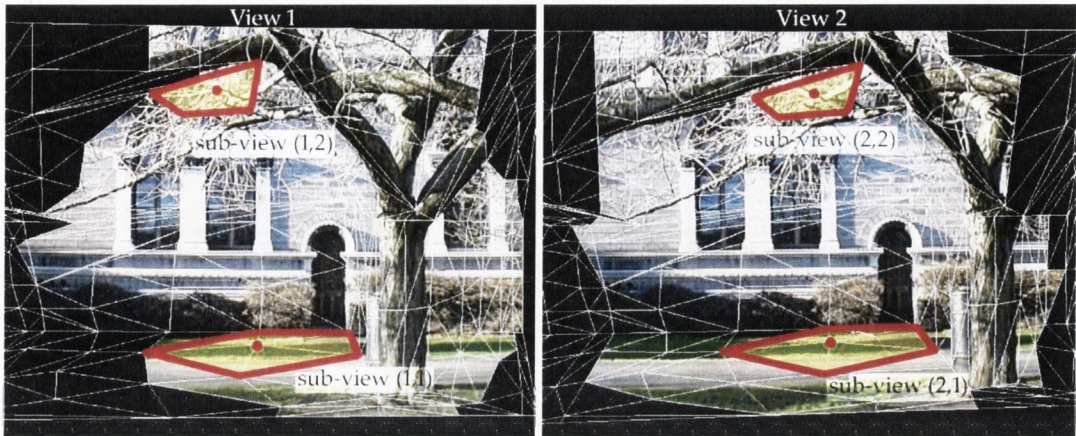


Figure 5.6: Four sub-views, corresponding to the same two geometric proxy vertices seen in two different input views.

blended at the vertex of index `vertexIndex`, then creates a list of associations between those camera indices and the corresponding camera data.

Another reason was debugging time. The presence of a garbage collector removes all potential for memory allocation errors. The strong typing constraints also tend to displace a lot of errors from runtime to compilation. We thought it would be worthwhile to investigate whether Haskell’s lazy evaluation feature would find applications within our framework. There was also a novelty value in developing a non trivial graphics application in Haskell. According to the successive Computer Language Shootout benchmarks, the GHC Haskell compiler is rapidly closing its performance gap compared to C compilers. Finally, Haskell features a well designed foreign function interface that makes it possible to invoke code written in other languages with little effort, so choosing it did not mean forsaking the use of many well-written C libraries.

Here is a brief overview of the code organization, followed by more detailed descriptions of those parts that merit them.

5.2.1 Organization

The application can be run in three different modes:

1. A rendering mode that reads an unstructured lumigraph (using either the original images or them re-packed in an atlas as textures) and displays it, with the possibility of

excluding views or sub-views on the fly if a view (sub-view) selection priority list is provided.

2. A view selection mode that executes the view selection algorithm (for whole views or sub-views) and produces a view (sub-view) selection priority list.
3. A texture atlas generation mode that takes as input a sub-view selection priority list along with a number of sub-views to discard and outputs a re-packed texture atlas.

The general structure of the application follows the Haskell concept of module: collections of functions implementing a specific logical functionality with an interface describing how they can be used from other modules. Figure 5.7 shows how the modules interact.

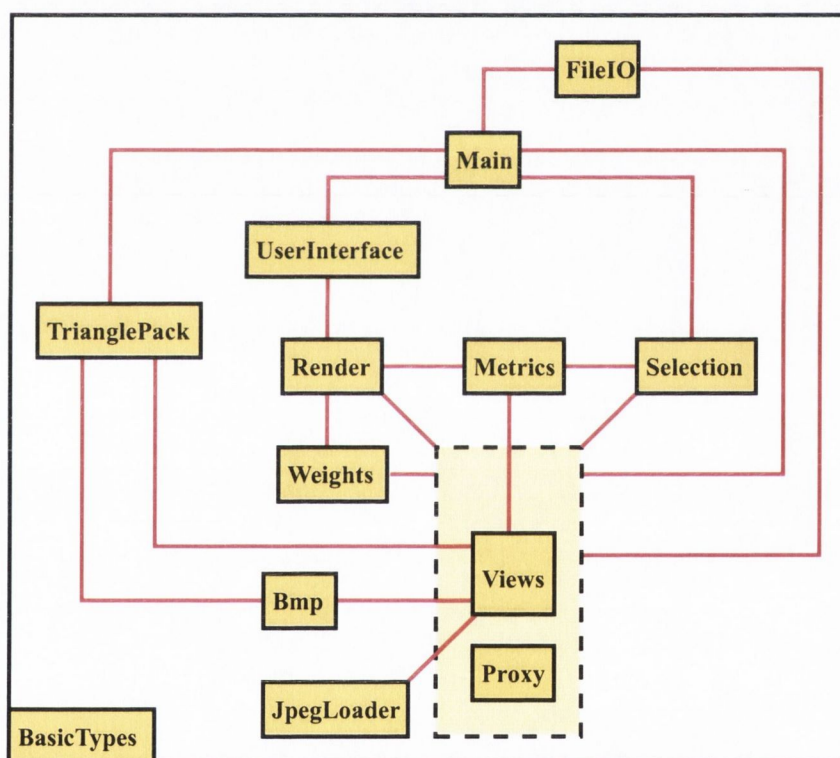


Figure 5.7: Application organization diagram.

The module names are fairly self-explanatory, but short descriptions follow.

- **BasicTypes** contains the declaration of all the types commonly used throughout the program, i.e.:

```
type VertexIndex = Int
type Blendable = Bool
```

- **Main** is where OpenGL and GLUT are initialized, callbacks for the user interface registered, and command line arguments interpreted to select the application mode and set various parameters.
- **Jpegloader** consists of a binding to the JPEG library.
- **Bmp** contains an implementation of a simple .bmp image loader and saver.
- **FileIO** is where the loading and saving of non-picture files is done. It includes parsers for the Alias .obj mesh format used to represent the geometric proxy, and for our unstructured lumigraph format, which simply consists of a base pathname to the pictures followed by a list of picture indices along with the parameters of the corresponding camera. Because of the interaction between the different modes of the application, and because we implemented the view selection algorithm so that it can be interrupted and resumed, a number of internal data structures have to be storable on disk to be recovered. In Haskell, all basic types instantiate the `Show` and `Read` classes, and so do most type combinators, like lists, tuples and arrays. Thus, thanks to the strong typing constraints, it is only necessary to write a single generic function to save (read) any internal structure to (from) disk:

```
saveAnything :: (Show a) => FilePath -> a -> IO ()
saveAnything filename thing =
    writeFile filename $ show thing

recoverAnything :: (Read a) => FilePath -> IO a
recoverAnything filename =
    liftM read $ readFile filename
```

Where `a` is a generic type. In the case of a structure associating a list of blendable views to each vertex of the geometric proxy, it could stand for `Array VertexIndex [CameraIndex]`. The `IO` monad is a language construction that reconciles

calls to functions that generate side effects, such as disk writing or displaying, with Haskell's pure functional behaviour. The `mapM` function is the list application function in the `IO` monad, which applies a function to each element of a list in sequence.

- **UserInterface** defines the behaviour of the Glut keyboard and mouse callbacks for the rendering mode.
- **Render** is where the unstructured lumigraph rendering is done. It contains the display function for both rendering and view selection modes.
- **TrianglePack** implements the texture atlas re-packing.
- **Views** contains functions to translate camera parameters into OpenGL projection matrices and setup projective texture mapping.
- **Proxy** handles preliminary tasks that have to do with the geometric proxy. This includes determining which vertices are visible from what views, and what views are eligible to be blended at each vertex.
- **Weights** computes the blending weights that input views should have at a particular point in space.
- **Metrics** implements GPU versions of SSIM and RMS. It also defines a binding for the fragment program extension of OpenGL, as it did not function properly yet in HOpenGL at the time of implementation, and a binding to Hector Yee's PDIFF implementation.
- **Selection** implements the view selection algorithms and data structures.

5.2.2 Unstructured Lumigraph Renderer

Implementing the unstructured lumigraph rendering algorithm is fairly straightforward. We will now present some details about our implementation.

Blending Weights

As explained earlier, for the aggregate perceptual scores of our view selection approach to make sense, the initial image plane triangulation step of the algorithm has to be bypassed. As

a side-effect, the computation of the blending weights proposed by Buehler *et al.* [BBM⁺01] can be simplified. In the original paper, for a given point P in the represented scene, input views (here index by i) are first sorted according to a penalty function that is the weighted sum of three terms:

$$\text{penalty}(i) = \alpha \text{penalty}_{ang}(i) + \beta \text{penalty}_{res}(i) + \gamma \text{penalty}_{fov}(i) \quad (5.1)$$

The term $\text{penalty}_{ang}(i)$ accounts for the difference of viewing angle between view i and the desired view, while $\text{penalty}_{res}(i)$ accounts for the difference in the resolution with which the scene surface at point P is sampled in view i and the desired view. Finally, $\text{penalty}_{fov}(i)$ represents the penalty ascribed to an input view i that does not see point P . It is different from the previous two terms 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 colour influence it, and it does not depend on the parameters of the desired view.

In our case, the set of points $\{P\}$ (which consists of the vertices of the geometric proxy) is not view-dependent as there is no image plane triangulation. This means that $\text{penalty}_{fov}(i)$ can be pre-computed by projecting the vertices of the geometric proxy into each input view. The original unstructured lumigraph paper [BBM⁺01] is somewhat concise on the computation details of this term. But, because of the discretisation of the blending field, it is more complex than what a cursory reading would suggest. 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. Therefore, if $\text{penalty}_{fov}(i)$ is computed based 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 colour 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 5.8. As a result, we precompute $\text{penalty}_{fov}(i)$ by checking for each vertex not only if it is visible in view i , but if all of its neighbours are visible too.

From there, if `camList` is the list of blendable view indices for the current vertex and `bySecond` is an ordering that orders couples according to their second member by increasing value, we can extract the k best views with their penalties by writing:

```
take k $ sortBy bySecond $ [(i, penalty i) | i <- camList]
```

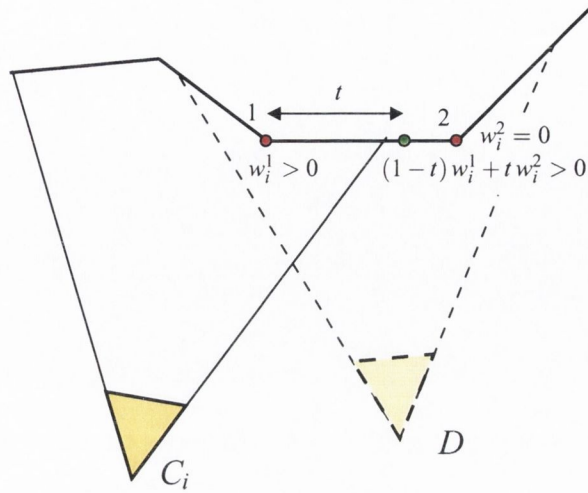



Figure 5.8: The blending weight for view i at the point in green, visible from the desired view D , but not seen by camera C_i , is non-null if the blending weight for view i at point 1 (w_i^1) is non-null.

Blending weights are then computed by following the smooth normalizing formula given in Buehler *et al.*'s paper.

Projective texture mapping

In unstructured lumigraph rendering, the parameterization of the light field is based on the pinhole camera model described in the background section. This means that, for a given picture in the lumigraph and its corresponding camera information, the position of the pixel $\mathbf{p}'_p$ that corresponds to the ray of light passing through the camera's optical centre and a point $\mathbf{p}'_w$ of the represented scene can be expressed as $C\mathbf{p}'_w$ followed by the perspective division, where C is the camera's matrix. OpenGL contains a fast implementation of this model in its vertex transformation pipeline, but its texture mapping mechanism is also designed to accommodate the model through automatic texture coordinate generation.

The texturing pipeline is able to handle 4 independent texture coordinates (named s , t , r and q). Only s , t and r can be used for actual texture look-up, allowing for 3D texturing. Coordinate q on the other hand behaves like the fourth coordinate in the homogeneous notation: s , t and r are divided by it before look-up. This last feature is what makes pro-

jective texture mapping feasible, as it allows for perspective division. In automatic texture coordinate generation mode, one or more of a vertex's texture coordinates are computed on the fly as a function of the vertex's coordinates after all transformations, as opposed to being specified beforehand by the programmer. Transformed vertices are expressed in the viewing camera's coordinate system, which is not desirable as the pinhole camera model expects points to be expressed in the world's coordinate system. However, among the three available texture coordinates generation modes, the `EyeLinear` mode first transforms the vertex back into the world's coordinate system. The rest of the computation is a dot product with a user-specified 4D vector, called `Plane` in OpenGL terminology. The perspective projection can therefore be achieved by specifying the corresponding line of the picture's camera matrix for each texture coordinate, where:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \quad (5.2)$$

The following code sets up projective texturing:

```
texGen S $= Enabled
texGen T $= Enabled
texGen Q $= Enabled

texGenFunc S $= EyeLinear $ Plane c11 c12 c13 c14
texGenFunc T $= EyeLinear $ Plane c21 c22 c23 c24
texGenFunc Q $= EyeLinear $ Plane c41 c42 c43 c44
```

The third coordinate is superfluous as we are looking up pixels from 2D pictures.

Blending

The blending of views over each triangle is implemented using OpenGL's alpha blending mechanism. Its principle is to compute the new colour of a pixel of the framebuffer as a linear function of the existing colour there (the `destination`), the colour of the fragment being drawn over it (the `source`) and the alpha component of the fragment. In our case, the blending weights of the views that contribute to a triangle sum to 1 at each of its vertices,

and it is easy to verify that linear interpolation preserves this property at any point of the triangle. As a result, assuming that the framebuffer starts out black, the weighted colour sum at each pixel of the triangle can be computed by re-drawing the triangle sequentially for each contributing view, setting the alpha component of each vertex to that view's blending weight for it, and defining the blending function as $\text{destination} + \alpha \text{ source}$:

```
blendEquation $= Just FuncAdd
blendFunc $= (SrcAlpha, One)
```

The blending weights computation step is designed so that it returns for each input view the list of triangles that have a vertex for which the view's blending weight is non-null, along with the weights:

```
weightsPerView :: [(ViewIndex, [(TriangleIndex, Blending-
Weights)])]
```

Input views that do not contribute to any triangle in the desired view are absent from the list. This allows us to bind each texture once only and then only if necessary.

If `drawTriangle :: (TriangleIndex, BlendingWeights) -> IO ()` is the function that draws a textured triangle with appropriate alpha components, then we can write (omitting the type declarations for conciseness, as they can be inferred from the type of `weightsPerView`):

```
viewContribution (viewIndex, triangleList) =
  do
    textureBinding Texture2D $= Just viewIndex
    mapM drawTriangle triangleList
```

And finally:

```
drawLumigraph viewList = mapM viewContribution viewList
```

5.2.3 Implementation of the Structural Similarity Index

We implement a modified version of SSIM using OpenGL's fragment shader mechanism. We need to be careful not to take into account the signal outside of a triangle when computing the measure over it. To do 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 normalising, 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 find that this sits well with the fact that the corners are precisely where the ULR algorithm samples the blending field with the most accuracy. In practice, we ensure that the “undesirable” flag is properly set by scissoring the alpha channel to 0 within the bounding rectangle of the triangle, expanded by the radius of the SSIM window, before rendering and comparing it. Note that this flag can be arbitrarily set over areas of the triangle. We explain the benefits of this property in the touchstone view selection section. 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. Our implementation on an ATI Radeon 9800 XT graphics card did not allow for full precision arithmetic on each channel. However, we found that this was not a concern due to the summing of the measure over the triangle.

Figure 5.9 illustrates our implementation of the SSIM in six fragment shading passes, taking advantage of the separability of the weighting window in 1D horizontal and vertical steps.

The whole reference image is initially stored in texture unit 1. Before the triangle to be compared is rendered using the IBR 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 triangle in six fragment shading passes, taking advantage of the separability of the weighting window. We refer to the mathematical formulation of the SSIM given in Chapter 2 (Section 2.3.4) 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

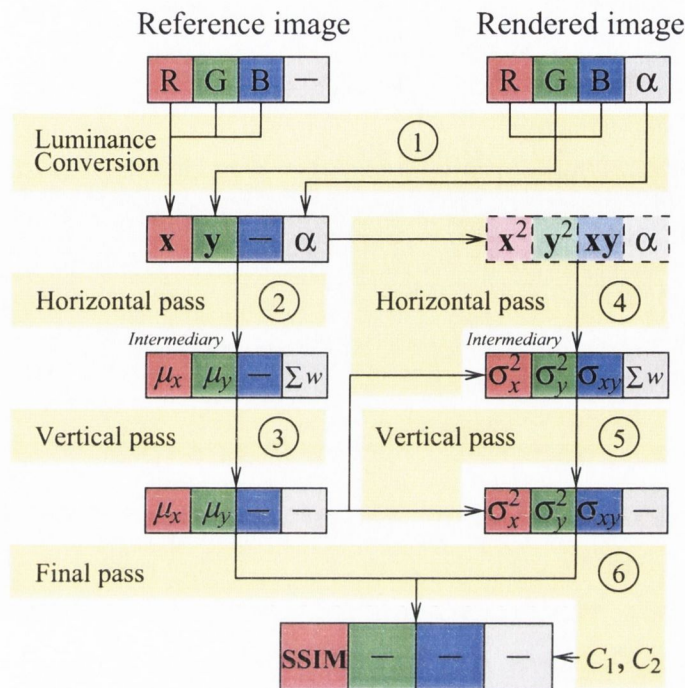


Figure 5.9: The six fragment shader passes used to compute SSIM indices.

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.

5.3 Texture Atlas Generation from Selected Sub-Views

The unstructured lumigraph rendering 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 sub-views, input views (now with holes in them) have to be repacked into a texture atlas to optimise the use of texture memory.

It is interesting to note that, unlike for memory savings, our version of the algorithm automatically takes advantage of view removals in term of rendering speed. This is trivial in the case where a whole view is removed, as it is no longer necessary to compute its penalty value at each vertex. But in fact removed sub-views behave just the same: the view they belong to gets its `Blendable` status set to false over the vertices which it covers. It is therefore skipped when computing view penalties at those vertices.

5.3.1 Triangle Packing

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 [Hal98] 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 rasterising 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 rasterising, 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 5.10.

Since we are taking triangles from many input views, without modification Hale's [Hal98]

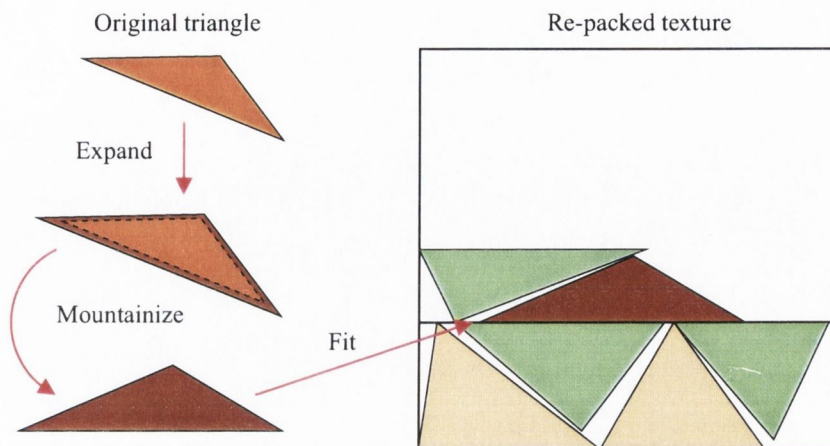


Figure 5.10: Triangle packing.

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. We therefore modify Hale's height sorting with a condition on the proximity of the input views' camera centres. Figure 5.11 shows a texture generated in this way.

5.3.2 Maintaining Perspective Correct Projective Texture Mapping

Repacking the sub-views into an atlas results in the loss of the correspondence between the texture coordinates and the initial camera projection matrices. It is not enough to simply store the texture coordinates of the repacked triangles and use them upon rendering. Indeed, this would bypass the per-fragment perspective division that is performed when automatic texture coordinate generation is used, leading to obvious projection errors.

There are two possibilities to address this problem: one is to compute "sub-camera" matrices for each sub-view and then load them at rendering time for automatic texture coordinate generation. The other is to implement a per-fragment perspective correction shader. The first option necessitates several expensive OpenGL state changes before drawing each triangle: the coefficients of the automatic texture coordinate generation mechanism have to be updated. The resulting adverse impact on performance is unacceptable.

The purpose of the perspective correction shader is to determine which pixel of the triangle the automatic texture coordinate generation mechanism would have fetched when using



Figure 5.11: A packed texture.

the original projection matrix. To achieve this, it computes the barycentric coordinates that the fragment would have had in the original triangle. The proper coordinates for the texture look-up are then obtained by weighting the coordinates of the vertices of the packed triangle in the texture atlas by these barycentric coordinates. The fragment program needs to be given the texture coordinates of the current triangle's vertices in the original view, as well as in the texture atlas. In OpenGL, passing parameters to a fragment program is an expensive state change that leads to a performance hit, comparable to that of updating automatic texture coordinate generation coefficients. Fortunately, the ULR algorithm does not make use of colour or normal information upon rasterizing (except for the alpha channel). This provides us with the space to pass the necessary parameters using efficient instructions that are callable within `glBegin - glEnd` pairs.

Chapter 6

Results and evaluation

In the course of our work, a number of datasets were produced. Some were generated from renderings of synthetic scenes, while others were captured. They were used as test data for user studies that we conducted to evaluate the view selection techniques that we presented in the previous chapter. The case of whole view selection is treated first, followed by sub-view selection. We then provide an analysis of the texture memory savings and rendering speed-ups achieved by re-packing remaining sub-views. Finally, we explain how the principle behind our view selection approach can be applied to automatically tune the user parameters of image-based rendering techniques and provide design details in the case of unstructured lumigraphs.

6.1 Whole view selection

In Chapter 5, we introduced a framework to select input views based on visual fidelity metrics. In order to validate our approach, we conducted a two alternate forced choice (2AFC) experiment comparing its output with that of a regular view selection process that we implemented based on a hierarchical clustering approach. For this user study, we only considered the use of the structural similarity index within our framework. Two synthetic scenes were created to provide controlled stimuli for the experiment.

6.1.1 Comparisons

As can be seen in Chapter 2, there are no techniques described in the literature that provide the same functionality as our method, as existing view selection methods assume that the geometry of the scene is accurately known. Therefore, a comparison with “competing” methods is difficult to engineer. We have considered comparing our technique to manual hand-picking. However, we felt that the outcome would be inconclusive due to the strong dependence on the abilities of the person choosing the views. Our experience indicates that, when the scene exhibits complex and scattered view dependent properties, manual view selection is confusing and results in an ad-hoc regular placement of retained views due to lack of understanding of how choices interact.

A sensible view selection approach is to select input views so that the retained set samples the viewing parameters (position, distance/resolution and parts of the scene covered) as regularly as possible. If p is the desired number of remaining views, a way to achieve this is to group the input views in p clusters, and then retain for each cluster the closest view to its centre. Since it is unclear what the positions of initial cluster centres should be, we use a hierarchical clustering algorithm.

The crucial issue in using hierarchical clustering for view selection is to define a proper distance over view parameters. Camera centre position, viewing direction and field of view should all be taken into account in some way, which is difficult as they exist in different spaces (Euclidean vs directions vs visible object space). In this respect, the ULR blending weights show good potential to solve this issue. However, some work is needed to exploit them as a camera to camera distance measure, which would involve considering the overlap of vertices affected by each view. In this work, we focused on a perceptual approach, as we felt that the actual content of the images should be exploited.

We avoided this problem of unreconcilable viewing parameters by considering only the Euclidean distance between camera centres. The test scenes were designed to ensure that the camera direction did not vary much over the input views. Distance and field of view were kept roughly constant. In this context, a method that would successfully incorporate all viewing parameters would yield similar results to the approach we took. It should be noted that methods based on view entropy, such as those described in [FCOL00] and [VF01], would also select evenly placed views if applied to the geometric proxies of our test scenes (although those techniques are designed to operate on more accurate geometric represen-

tations of the scenes). This is because each view exhibits more or less the same coverage versus sampling resolution ratio over the geometric proxies of our test scenes. The drawback of our choice is that we are unable to evaluate our approach on scenes representative of the full range of situations that the unstructured lumigraph rendering (ULR) algorithm can handle. However, we are optimistic that it will generalise well as it does not make any more assumptions than ULR does.

6.1.2 Test scenes

This user study was conducted close to the beginning of our research into view selection as a milestone to test the potential of our approach. At the time, efficient structure from motion packages were not as easy to procure as they are now. Image-based rendering datasets of real-world scenes were also hard to come by. For expediency, we chose to work on synthetic scenes.

The initial sets of input views consist of 240 views of each scene lying roughly on a plane parallel to the geometric proxy. The extent of the viewing volume and the number of input views were chosen so that view-dependent effects such as parallax, specularities and reflections were captured adequately while keeping the number of views reasonable. Each view consists of a 512×512 picture along with the camera pose used to render it. The geometric proxies were created in a matter of seconds from the available geometries of the scenes. Their level of subdivision was chosen to ensure adequate sampling of the blending field without hurting rendering performance, they have no relationship with the level of geometric complexity of either scenes.

1. **Celtic logo:** This scene consists of two juxtaposed Celtic designs. The logo on the left is a highly detailed extruded mesh obtained through displacement mapping and given a slightly specular finish. The logo on the right is a perfectly diffuse texture mapped plane. The geometric proxy used is a plane made of 256 triangular faces. The contrast between the strong view dependence on the left and its absence on the right is intended to enable us to judge whether our method correctly retains more views that cover the view dependent part. (*cf.* Figure 6.1, bottom)
2. **Pub scene:** Our aim was to reproduce a typical real world scene that one may wish to capture using image based techniques. Thus, we attempted to model geometric details

and lighting phenomena, such as window reflections, as accurately as possible. We took as our starting point a textured model of a city block extracted from the Virtual Dublin project [HO03]. Fine details were added to the model in 3D studio [Aut]. A dataset consisting of views of the Messrs. Maguire pub façade was then produced by running a global illumination simulation. This was done using Splutterfish's Brazil renderer plugin for 3D studio. A high dynamic range panorama of a Dublin sky was used for environment lighting. We created the panorama from a set of photographs taken from the rooftop of a nearby shopping centre. Each portion of sky was photographed with varying levels of exposure, the high dynamic range panorama was assembled using HDR shop [Uni] after having stitched the photographs of same exposure together. The geometric proxy is roughly planar, with 224 faces (*cf.* Figure 6.1, top). Figure 6.2 shows some views of the pub scene. In the real world, this scene could have been captured with an off-the-shelf digital camera held at different heights while walking up and down the street. The presence of well-marked corners would have allowed for camera pose reconstruction using common correspondence based techniques. The proxy would have been created by fitting a subdivided plane to the reconstructed corner positions and editing some vertices to roughly fit jutting areas.

6.1.3 Experiment

We applied both view selection methods to our test scenes: our visual fidelity based approach and the hierarchical clustering one. It took roughly 36 hours for the perceptual algorithm to order the initial list of input views for each scene on a 1.5GHz Pentium III machine. We then pre-selected three target numbers of remaining input views for each scene (*i.e.*, how much of the ordered list for that scene to keep). The pre-selection was done arbitrarily without knowledge of the visual outcome of either selection methods. Numbers were chosen in an *ad hoc* way reflecting the amount of view dependence present in each scene. The numbers were 100, 60 and 20 for the celtic logo scene, corresponding to 41.5%, 25% and 8.5% of the number of initial views, and 180, 120 and 60 for the pub scene, corresponding to 75%, 50% and 25%. The clustering algorithm was able to select views in less than a minute for each given target number. Figure 6.3 shows screenshots of the logo scene rendered with 8.5% remaining views for both methods. At that point, we could verify that the perceptual method was discarding views more cleverly than the regular one in the case of the celtic logo scene

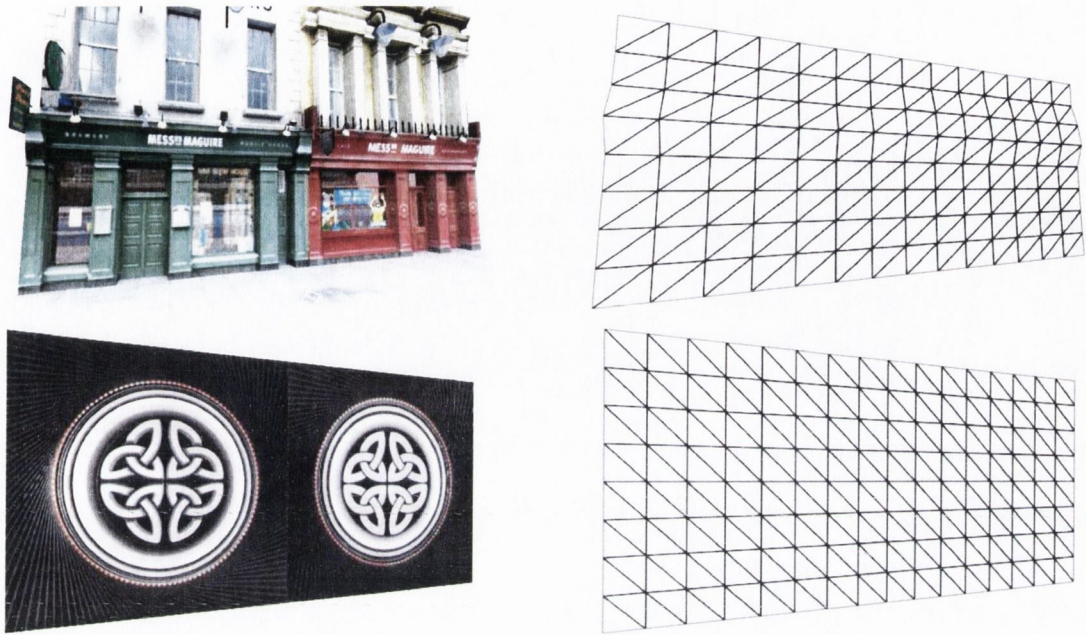


Figure 6.1: General views of the test scenes and their geometric proxy.

by examining the coverage of the remaining views, as illustrated in Figure 6.4. Predictably, coverage results from each technique were not easily distinguishable visually for the pub scene, due to its even distribution of view dependence.

Stimuli: Using each of the 12 sets of selected input views (2 for each method $\times$ 2 for each scene $\times$ 3 for each number of remaining views), we rendered two arbitrarily chosen sweeping sequences for each scene with the ULR algorithm, for a total of 24 stimuli. We produced gold standard sequences by rendering them in the same way as we had rendered the input views in the first place. In order to avoid delays and participant behaviour dependent effects in the presentation, each sequence was rendered at a resolution of 256×256 so that they could be displayed on the same screen.

Method and participants: We chose to implement the experiment around a web based interface because of the convenience it provides to display video files. Since the 2AFC design is entirely deterministic (in that the order of stimulus presentation does not depend on subject behaviour, but is purely random), it could be implemented as a simple PHP/MySQL web



Figure 6.2: Several input views for the pub scene, such as might be captured from the street with a video camera.

application. The database was used in two fashions: to store the responses of the subjects, but also to handle the changes of state of the experiment and in particular hold a randomised experiment plan generated upon connection of each participant.

The display device used was a 21" flatscreen CRT monitor, subtending a horizontal field of view of 36° from the chosen viewing distance. Participants were presented with a series of 36 screens: one per sequence per scene per target number of remaining views times three repetitions. All screens were displayed at 80Hz and a resolution of 1280×1024 and were composed of the gold standard for that sequence at the top and the stimuli to be compared at the bottom left and right. Each video sequence had a resolution of 256×256 , covering roughly 7° of the subjects' horizontal field of view. This small size was chosen to lower the impact of each subject's gaze fixation sequence on the outcomes. All video sequences had a framerate of 25fps.

Figure 6.5 shows the experiment setup. On each screen, the task was to choose which



Figure 6.3: Left: conventional rendering. Middle: unstructured lumigraph rendering using 8.5% of initial input views selected by hierarchical clustering. Right: ULR rendering using the same number of views selected by our perceptual method. Note how most of the depth of the celtic motif is lost in the middle image. The loss is much more conspicuous when the viewer moves around the scene as the parallax effect is damped.

ULR rendered video sequence the viewer found most similar to the gold standard. They were specifically asked to pay particular attention to the feeling of depth conveyed by each sequence. They could view each sequence up to three times in the order of their choosing by clicking on a button on-screen. The screen presentation order was randomized, as well as the respective positions in each screen of the sequence corresponding to the perceptual method and the sequence corresponding to the regular one. A learning screen was shown first to familiarize participants with the setting and task at hand. Fourteen students and staff members from Trinity College Dublin participated in the experiment and all had no *a priori* knowledge of what was being compared.

Results: Results are summarized in Figure 6.6. They show a very significant preference for the perceptual method in an average of 80% of the presentations. Encouragingly, the average preference rate was nearly as high in the case of the pub scene as for the celtic logo. This means that our approach was able to improve visual fidelity to the original even when view dependent phenomena were not spatially gathered (in object space or viewing space). A single factor ANOVA on the differences between scenes gave only marginal significance: $F(1,26) = 3.2$, **P-value** = 0.087.

As could be expected, the preference rate slightly rises with the number of discarded views, as the impact of improperly discarded views is less and less counterbalanced by the

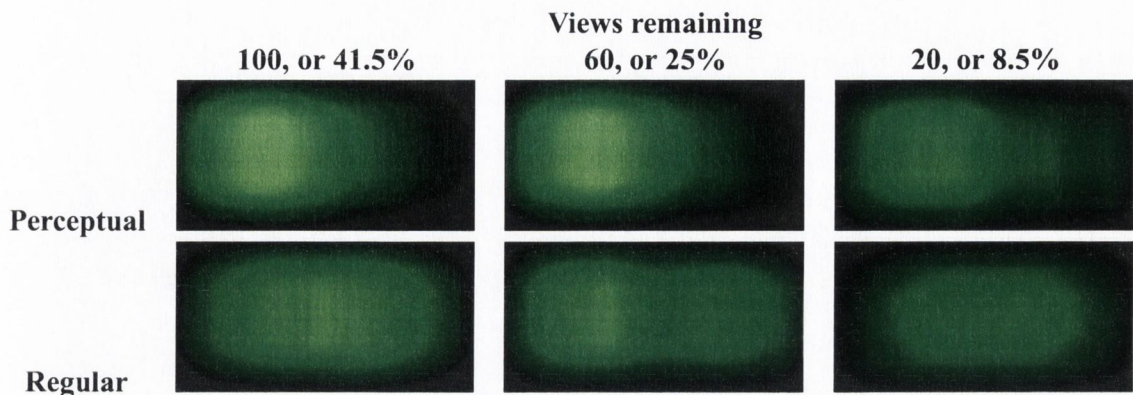


Figure 6.4: Coverage of the celtic logo’s geometric proxy by the remaining input views (brighter shades indicate better coverage). Columns correspond to the number of remaining views. Top row: perceptual approach, bottom row: regular.

contributions of remaining views. However, a single factor ANOVA between the preference ratings of the least, intermediate and most removed views cases showed that this difference falls slightly short of marginal significance: $F(2,39) = 2.4$, **P-value** = 0.107. (Another ANOVA indicated a marginally significant effect of the remaining views number variable when only run on the extreme cases: $F(1,26) = 3.4$, **P-value** = 0.075.)

6.1.4 Conclusions

Our method for the selection of whole views for unstructured lumigraph rendering had shown great potential on our synthetic test scenes. Once it has ordered the initial input views by increasing perceptual importance, an artist can quickly explore renderings of the scene using varying numbers of remaining views until he is satisfied with the compromise between visual quality and resource consumption (memory as well as GPU time).

At the time of this first study, the implementation of our view selection was in its first iteration and did not incorporate the evaluation of the visual fidelity metric at the triangle level and the degradation score caching mechanism described in Chapter 5. As a result, computation times quoted in the next section are more representative than the 36 hours necessary here.

From an artist’s point of view, besides view selection, the main technical difficulty in authoring unstructured image based representations is the problem of accurate registration of



Figure 6.5: Photograph showing the setup of the first experiment.

the input views, and their registration with the geometric proxy. The user study presented in the next section is based on test scenes captured from the real world with current registration techniques.

The success of a greedy approach is dependent on the conditioning of the input system. A bad conditioning corresponds to cases where taking the most profitable decision at a specific step has particularly negative impact on profits made at later steps, leading to much sub-optimal outcomes. It is a consequence of path-dependency of the outcome of individual decisions. In our case, the decisions in question are the choices of views to discard. There is a direct relationship between the extent of the represented scene that a view covers and how path-dependent the outcome of its removal is. To take an example, let us consider the case of the Celtic logo scene introduced earlier. All input views cover roughly the same extent of the scene. The more a view covers the pattern on the right, which was designed to exhibit no view-dependence, the more redundant it will be judged by the visual fidelity metric and the sooner it will be discarded. Some of those early discarded views however contained

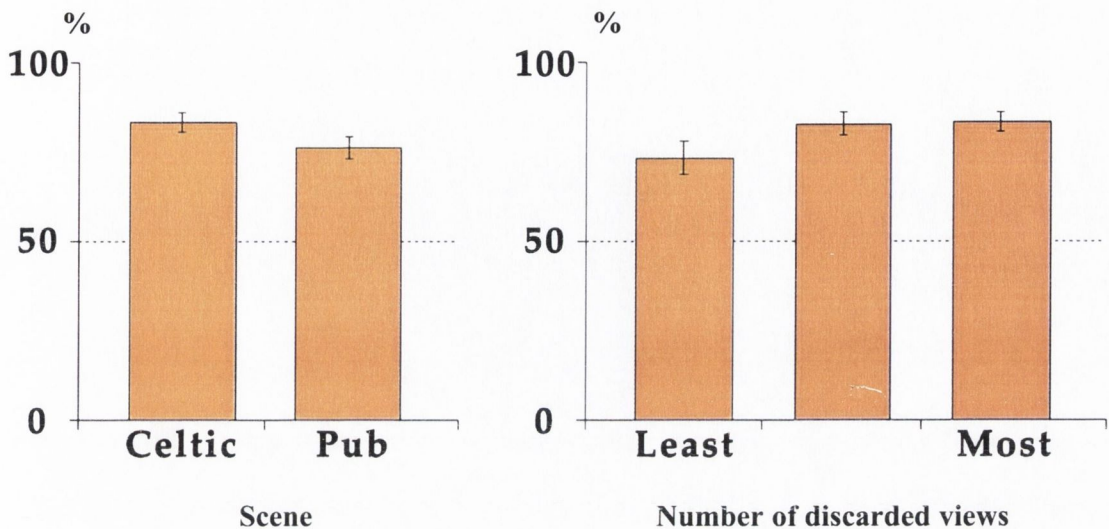


Figure 6.6: Percentage of times that participants preferred the perceptual view selection to the regular view selection. On the left, by scene. On the right, by number of input views removed.

information on part of the view-dependent region of the scene. The loss of this information will influence later removals in ways that could be detrimental to the final outcome. Ideally, we would have wished to only discard the redundant portions of these views and keep their useful regions for later decisions.

Splitting the input views into sub-views helps achieve this, it allows the selection process to better adapt to local variations in view dependence.

6.2 Sub-view selection

The second user study concerns itself with the evaluation of our view selection framework refined to operate at the sub-view level.

We acquired four video sequences of different scenes using a hand-held Canon XL1 PAL DV camcorder. Camera poses were recovered using PFTrack [The]. Rough geometric proxies were created in a few dozen minutes with a standard polygon editing tool, using three 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

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

Table 6.1: Test scenes statistics (note that we do not count sub-views that are not blendable for lack of visibility of the whole triangle ring)

viewing region of interest, unless they fit the scene geometry accurately in the regions where overlapping occurs (See Chapter 5, Section 5.1.4 for an in-depth discussion of this issue). Statistics for each unstructured lumigraph thus created are summarized in Table 6.1. Views of these scenes are shown in Figures 2 to 9 in the appendix.

6.2.1 User study

Design

We first wished to measure how much better our sub-view selection technique is to currently available alternatives. As discussed in the previous section and in Chapter 5, without accurate geometric knowledge of the scene, current alternatives are limited to techniques that ensure a uniform coverage of the geometric proxy by the retained views while sampling the range of viewing parameters evenly. We therefore picked two test scenes where uniform coverage could easily be enforced for any number of discarded sub-views: their initial views were obtained by evenly trucking the camera respectively in front of a building facade (Cf. Appendix 7.2, Figure 5) and in the university library (Cf. Appendix 7.2, Figure 4). 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 sub-view level, for a given number sv of discarded sub-views, we estimated the number v of views to discard by dividing sv by the average number of sub-views contained in each view. Once all the sub-views 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 sub-views than needed had been discarded) a single sub-view from each. To ensure the even distribution of removed (recovered) sub-views, we used a vertex colouring of the geometric proxy and picked sub-views of the same colour, moving on to the next colour when

one was exhausted. The colouring was obtained with a standard graph colouring algorithm.

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 sub-view discarding strategies (three metrics plus *regular*), at which number of discarded sub-views users started to perceive visual degradation compared to the original dataset. We chose a double-random staircase experiment design ([Tre95]). In a staircase experiment, users are presented with a sequence of stimuli at different level 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 (two seconds) rendered video sequences of the test scenes, one of which uses the original dataset, shown one after the other. The number of sub-views 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.

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, as became obvious during a pilot phase of the experiment. For this reason, we produced two sets of smaller videos showing sub-regions of the test scenes. For the *Exterior* scene, the sub-regions consisted of the windows plus the tree branch (*window*), and the door plus the tree trunk (*door*). For the *Library* scene, they consisted of a section of the upper glass barrier (*glass*), and a section of the upper book stacks (*stacks*). The novel camera paths used to produce the videos were created ad-hoc. The same path was used for both sub-regions of a test scene.

The full experiment thus consisted of 32 staircases: 4 sub-view discarding strategies $\times$ 4 test sub-regions $\times$ 2 staircases (one ascending, one descending), all randomly interleaved to counteract learning and expectation effects. Each staircase was limited to 8 reversals. We chose 3200 sub-views as the initial increment (decrement) value of the stimulus level, which was lowered to 1600 upon the third reversal and 800 upon the sixth. A few pilot runs of

the experiments yielded an average duration of 110 minutes. 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 sub-region of each scene. Which session each participant sat through first was randomized. 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.

Implementation and setup

Unlike the 2AFC design, the staircase method is path dependent: the next stimulus shown depends on the earlier choices of the subject. We anticipated that the book-keeping of the states of the 12 staircases would be quite tedious to code robustly using PHP scripts, but we wanted to retain the convenience of the browser interface. We thus decided to experiment with Seaside [Avi], a recent web application development environment built on top of Smalltalk [Ing78]. It allowed us to quickly implement the staircase method in a friendly object oriented framework, with no concern for scripting related hassles such as variable propagation. The implementation was done in Squeak [Squ], an open-source Smalltalk environment.

The display device used was a 24" LCD monitor displaying 1280×1024 pixels at 75Hz, subtending roughly 30° of the horizontal field of view from the chosen viewing distance. This corresponded to horizontal fields of view of roughly between 4° and 5° for each video sequence. All video sequences were displayed at a framerate of 25fps.

Figure 6.7 shows a photograph of a subject taking the experiment.

Results

By fitting a psychometric function to their responses, we estimated the Point of Subjective Equality (PSE) *i.e.* the number of discarded sub-views at which participants had an even chance of reporting some visual degradation. Results are summarized in Figure 6.8. Results for the *door* sub-region 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. The error bars in the chart



Figure 6.7: Photograph showing the setup of the second experiment.

represent standard errors, the fact that they are narrow indicates that participants gave similar responses for a given stimulus. This makes us confident that the results are representative of the broad population. Of course it is possible that the baseline abilities of all the participants were shifted in a similar manner, but over 16 subjects, it is very unlikely. In any case, the comparison between discarded strategies would still hold as there is a single task that involves the same abilities.

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 sub-region 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 sub-region factor, it is natural that a given metric yields different numbers of discarded sub-views at the PSE on different sub-regions, 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 sub-view 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* sub-region ($p = 0.057$). 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* sub-region ($p = 0.052$), providing the exception to the rule that each metric used in our framework yielded statistically different results.

A reading of the chart in Figure 6.8 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* sub-regions, yet strongly outperforms the competition for the *window* sub-region. 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* sub-region, which contains complex patterns of intertwined small branches in the forefront, whose inaccurate fit with the geometric proxy yields strong blurring as sub-views are discarded.

In Figure 6.9 we plot the aggregated psychometric function over all participants for each sub-region. 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* sub-region, 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.

6.2.2 Rendering performance

The sub-views of each unstructured lumigraph were perceptually ordered by our greedy pruning process. Processing times are given in Table 6.2. We then navigated the unstruc-

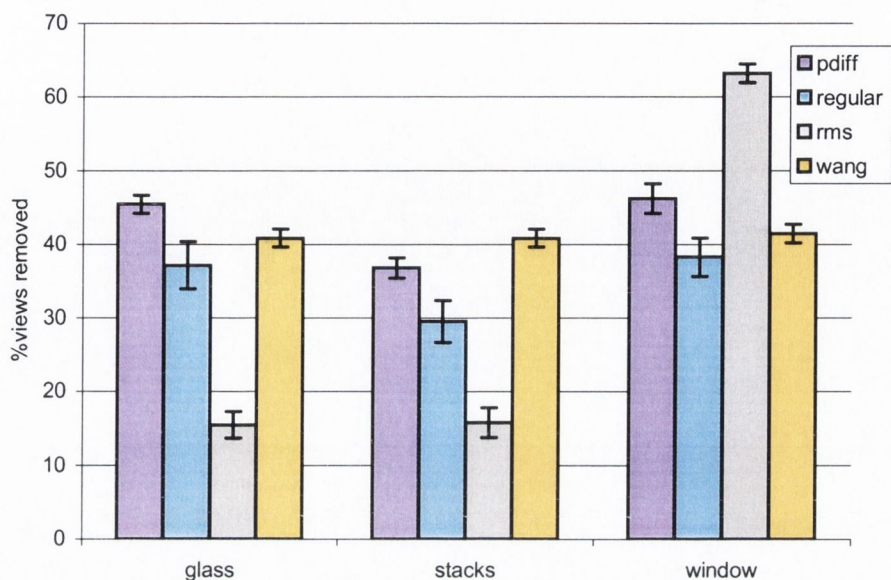


Figure 6.8: Proportion of sub-views discarded for each strategy for each test sub-region at the PSE (standard errors are shown).

tured lumigraphs while changing the number of retained sub-views within the ordered list, allowing us to determine the lowest number before degradations became noticeable. Figure 6.10 shows a novel view rendered at this threshold juxtaposed with an input photograph.

Resulting texture memory footprints and framerates are summarized in Table 6.4. They vary significantly because of the very different 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

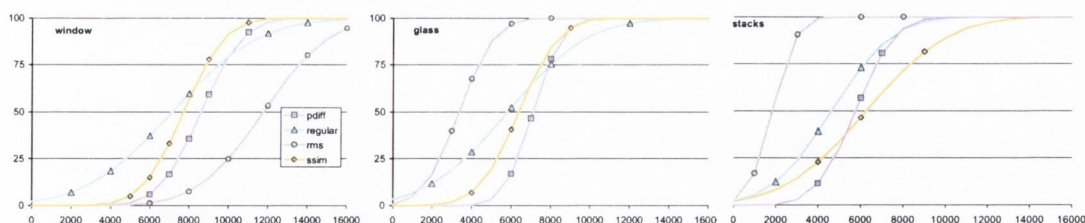


Figure 6.9: Psychometric functions aggregated over the 16 participants for each sub-region. The Y axis represents the probability of a participant reporting some visual degradation. The X axis shows the number of sub-views discarded.

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

Table 6.2: Approximate time taken to order all sub-views by increasing perceptual importance (hours).



Figure 6.10: Exterior scene. Left: frame of digital video footage. Right: a novel view rendered after trimming of the unstructured lumigraph.

of the human operator in making a judgement call on the data/quality compromise - a call that our technique facilitates. 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 framerates quoted in Table 6.4 illustrate this point: as expected, the rendering speed-up correlates with the number of vertices of the geometric proxy. Table 6.5 contains both ratios.

Figure 6.11 illustrates the behaviour of our technique on an input view. In this test scene,

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

Table 6.3: Number of retained sub-views for each test scene.

	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

Table 6.4: 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 framerate due to texture swapping.

the proxy fits the ground and the wall, but is jutting towards the viewer at the tree’s trunk and branches. Our technique correctly discards sub-views (in red) where little view dependence is present, *i.e.* where the proxy is modeling the scene properly and the surface there does not exhibit view dependent phenomena. Comparisons of renderings of the three other scenes made with different numbers of sub-views are shown in Figures 6 to 9 in the appendix.

6.2.3 Discussion

Our refined view selection method for unstructured lumigraph rendering, operating on sub-views, has shown great potential on real-world scenes, thus validating our framework. It yields significantly better results than regular discarding when using either PDIFF or SSIM, and this consistently over sub-regions of varying nature. Once it has ordered the initial sub-views by increasing perceptual importance, an artist can quickly explore renderings of the scene using varying numbers of remaining sub-views until he is satisfied with the compro-

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

Table 6.5: Results expressed in terms of texture memory usage ratio and number of candidate views per vertex ratio.



Figure 6.11: 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 sub-views that were discarded by our technique.

mise between visual quality and resource consumption (memory as well as rendering time).

When it comes to comparing the performance of different metrics within our framework, Yee and Newman's PerceptualDiff appears to win, but the results are less clear-cut. As acknowledged by Wang *et al* in their citation of the research conducted within the Modelfest framework [Mod] 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 background section (2), the closest techniques with which to compare our work deal with interactive rendering from compressed lightfields. Be they based on vector quantization, discrete cosine transform or wavelets, they typically achieve much higher compression ratios than what we obtain, thanks to their much finer granularity.

The most dramatic results are however obtained with stronger requirements on 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 [XG03]. 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 optimized ULR implementation. However, at roughly 7 to 15 fps, 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 structured light field techniques tend to enforce rigidly. Another is its higher suitability for inclusion in a 3D engine.

One last *caveat* is that the compression ratios quoted for structured techniques use the full size of the parameterisation as denominator, irrespective of whether enough light information was captured in the input images to fill it in the first place. As a result, ratios obtained in the compression of unstructured datasets are expressed more conservatively.

The use of our framework raises the question of how to choose the touchstone views. The choice of touchstone views is related to the behavior of the visual fidelity metric: In our experience, the quality of the registrations obtained with commercial tracking software was good enough for the metric to behave well. However, for a scene where some views are not properly registered, any 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 behaviour of the framework for views taken from a neighbouring viewpoint. If on the other hand the badly registered view is excluded from the set of touchstone views, it (ot its sub-views) will be automatically discarded by the framework, as its 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 a touchstone view, in an even distribution.

A limitation of our technique is the use of a view-independent geometric proxy, which as a result has to have unit depth complexity where the geometry of the scene is not precisely known. It is however conceivable to use such a proxy for the purpose of applying our technique, and then to render the resulting lumigraph using view-dependent triangulation. The view independence of the proxy also aggravates a limitation of the ULR algorithm mentioned in Figure 6.1, namely that views that do not cover a triangle entirely cannot be used to texture it. This limitation can however be worked around during authoring by either taking wider

angle pictures of the desired scene or panning the camera. Our framework ensures that the useless visual information around the edges will be culled in the final packing.

6.3 Parameter adjusting

One authoring drawback that all image-based rendering techniques presented in Chapter 2 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.

In Chapter 5, we introduced our aggregate fidelity metric for image-based representation. By exploring the metric scores resulting from different user parameter choices for the rendering algorithm, optimal parameters can be determined.

6.3.1 Case of the unstructured lumigraph

As discussed in Chapter 5 (Section 5.2.2), there are 3 user selected parameters in our implementation of the unstructured lumigraph algorithm. They are: the number k of views that are blended over a given triangle, the weight α given to the penalty that accounts for viewing angle discrepancy, and the weight β given to the penalty that accounts for viewing resolution discrepancy (γ was made irrelevant thanks to the blendability pre-computation). Coefficients α and β 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 and not as an absolute value. Fixing α as 1 and only letting β vary therefore yields the same functionality.

In the same way that we used 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. Evaluating the aggregate metric is not particularly fast. However it can be made faster by using a sparser set of touchstone views. Besides, the parameter space is only 2-dimensional, with k varying discretely over a narrow range of suitable values (from 1 to a reasonable number, i.e. 8). It would therefore be possible to explore parameters as an optimisation problem. In practice, we simply evaluate the aggregate metric over a

predetermined set of possible parameter pairs and go with the one that gives the highest value.

6.3.2 Finer tuning

Just as the finer visual granularity of the unstructured lumigraph algorithm was used to perform view selection at the sub-view level, it can be exploited to fine tune the rendering 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 Chapter 5, Section 5.1.4, Figure 5.6). 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 $(8 \times 20)^3 \approx 4 \cdot 10^6$ 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 $(8 \times 20)^n$ 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.

6.3.3 Results and discussion

We have only applied our framework to one test scene (shown in the appendix, Figure 5) and only to adjust the parameters globally for the whole scene. Figure 6.12 shows a plot of the aggregate metric as a function of k and β for that 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. We are working on producing results on other test scenes as well as applying our local parameter tuning heuristic.

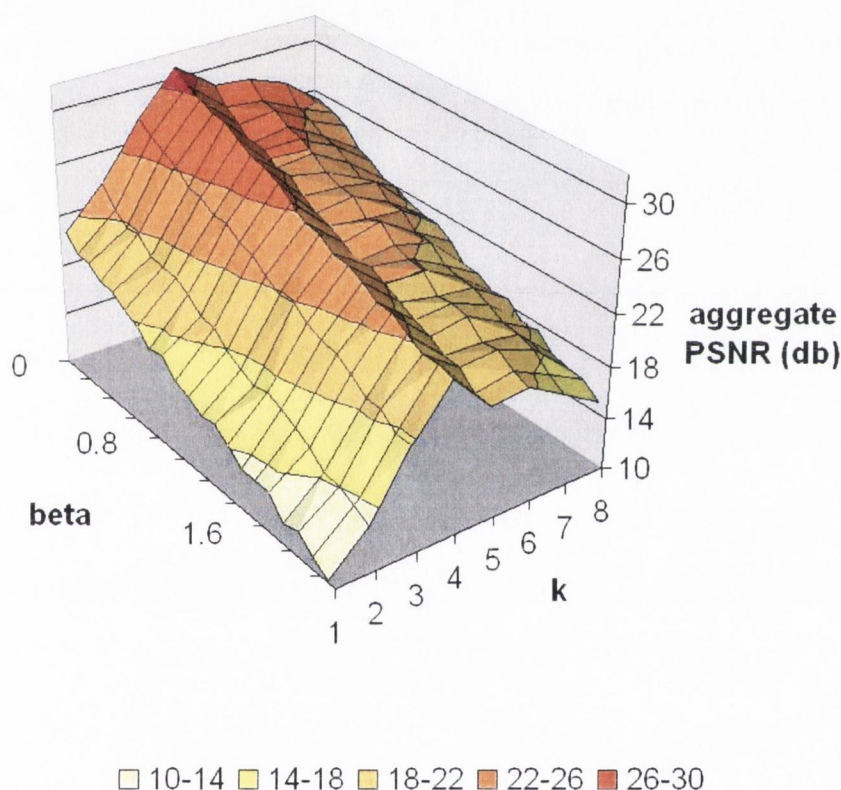


Figure 6.12: Aggregate metric against k and β for the test scene shown in Figure 5.

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.

Chapter 7

Applications, conclusions and future work

7.1 Practical applications of the contributions

To conclude this thesis and illustrate the potential of our contributions, let us return to the imaginary real-world authoring scenario mentioned in the introduction and see where they fit. Recall that it is the case of an artist in charge of designing city blocks for a skate-boarding simulation video game:

1. **Game design choices** In this first step of the pipeline, the artist familiarizes himself with the features of the game that are relevant to his task. For instance, the fact that the project is a skate-boarding game places constraints on the player character's viewpoint: it is roughly at standing eye level and it is limited to the areas that the game designers have made skateable within the city maps.
2. **Representation choice** Here, the artist collaborates with the game programmers to decide on a suitable representation for the city blocks. From the map design, it is possible to anticipate an interval of distances from which each block will be seen by the player. Those distance intervals can guide the choice of representation, as illustrated in Figure 7.1, but it is constrained by software issues. Programmers could be reluctant to implement orthogonal techniques (*e.g.* light field rendering and 3D warping) to what they use for the rest of the game. Thanks to its flexibility, and because it uses rendering

mechanisms directly compatible with traditional 3D engines, an **unstructured lumigraph** like technique is likely to be chosen. At this point, programmers will also give the artist a budget in terms of texture memory and polygon count.

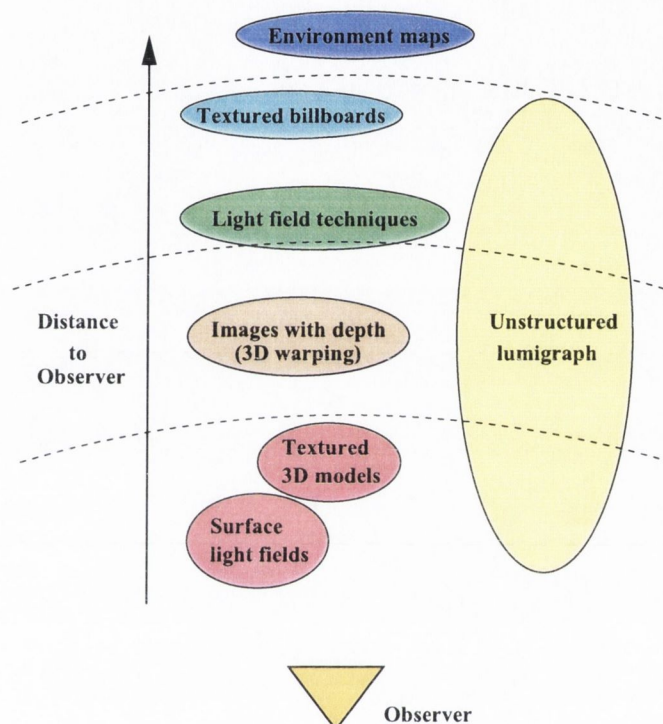


Figure 7.1: Types of suitable image-based representations as a function of distance to the observer.

3. **Capture of relevant building textures** The artist decides which buildings will be created from existing architecture. He then sets out to obtain collections of photographs of them with corresponding camera parameters in the most convenient manner: using a video camera and structure from motion software.
4. **Modelling of relevant buildings** Using the registered photographs as a guide, the artist creates geometric proxies of the buildings. To do this, he can use the **3D image-based modelling system** that we propose. By using all the photographs initially at his disposal, he can use our **method to automatically tune** the parameters of the unstructured lumigraph renderer and get a feel of the minimum level of detail of the geometric proxy necessary to convey the visual appearance of the building faithfully.

5. **City layout and refining of the end result** The artist now has a collection of buildings, some of which are represented by voluminous unstructured lumigraphs. Thanks to our **view selection framework**, he can interactively discard input sub-views until he is satisfied with the compromise between visual quality and memory footprint, with the assurance that the choice of discarded sub-views is clever, as shown in our **user studies**. With our **texture re-packing scheme**, he can be confident that the compression thus achieved will not have a negative impact on rendering speed, as in the case of structured methods, but in fact increase it as illustrated in our **performance analysis**.

He now arranges the buildings to design city blocks according to the city maps. Our framework lets him exploit the city layout to discard sub-views more cleverly in two ways: By removing touchstone views that correspond to impossible or unlikely player viewpoints from the evaluation of the aggregate fidelity metric. And by projecting occluding city blocks onto touchstone views and ignoring the corresponding areas when evaluating the metric. These steps ensure that input sub-views are given high redundancy scores if the visual information they provide mostly affects areas that the player will not see from their viewpoints. These two steps are not equivalent to abruptly culling the sub-views in question, as the **perceptual view selection framework** may determine that they are still useful to render the corresponding areas from neighbouring viewpoints.

Our **system for capturing problematic scenes** did not fit in this scenario, as buildings exhibit strong features that make the use of structure from motion algorithms appropriate for the recovery of camera parameters. Nonetheless, we have demonstrated the usefulness of the tools that we created.

7.2 Future work

In this thesis, we have argued for the authoring superiority of unstructured image-based rendering approaches over structured ones. However, as explained in the background and the evaluation chapters, structured approach have a compression advantage that could tip the scales in their favour when memory consumption is more of a concern than rendering speed. A possibility would be to provide artists with “black box” tools that convert easy to author and render unstructured representations into more compact structured ones and back,

for instance for transmission over a network. A formal comparison of the different aspects of compression techniques would be very worthwhile, but it is a challenge because of the slight variations in requirements and features of the various image-based rendering algorithms that make the choice of test scenarios that are suitable across the board difficult.

In its current implementation, the main drawback of our view selection method is its pre-computation time. Clocking at several hours, it is long enough to necessitate that the artist schedule around it. Given the somewhat brute force approach of the method, it is unlikely that it can be made instantaneous. However, considering that the number of input views can grow considerably with the area of the viewing region that the artist wishes to cover, work should be done on making it more efficient. Another major interest of increased efficiency is that it would make it possible to experiment with elaborate perceptual fidelity metrics. There are several avenues of investigation towards that aim. A statistical approach could be used at each removal step to avoid considering all remaining sub-views, but only a sub-set of them. It is also possible to considerably reduce computation time by playing with the number of touchstone views considered when applying the fidelity metric. Lastly, the approach seems to be easily parallelisable, since fidelity scores whose cached value have become obsolete are computed independently from each other at each iteration. In this respect, our use of functional programming could make parallelisation quite straightforward. We are currently investigating a port to parallel Haskell.

Finally, we would like to further experiment with multi-modal user interfaces using motion capture for computer assisted creation.

Appendix 1

This appendix contains figures showing the test scenes that were created to evaluate our view selection framework. Figures 2 to 5 show a few input views along with the geometric proxy used for unstructured lumigraph rendering for each test scene. Figures corresponding to each test scene are re-quoted in Table 1. Figures 6 to 9 show two novel views rendered with different numbers of retained sub-views for each test scene as explained in Chapter 6, Section 6.2.

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

Table 1: Test scenes statistics (note that we do not count sub-views that are not blendable for lack of visibility of the whole triangle ring)

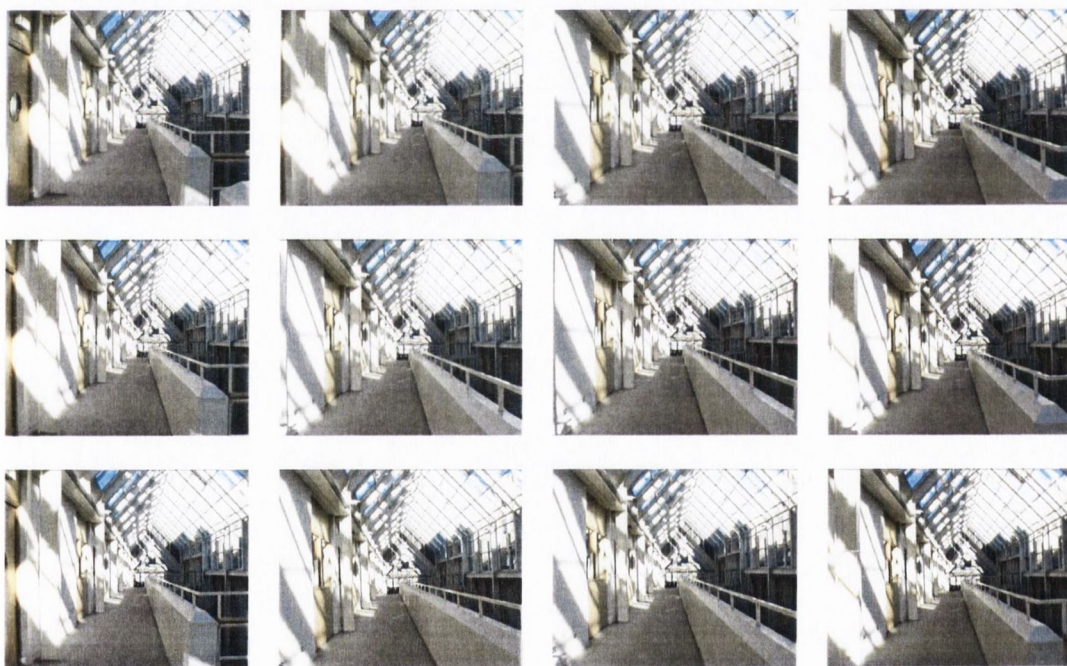
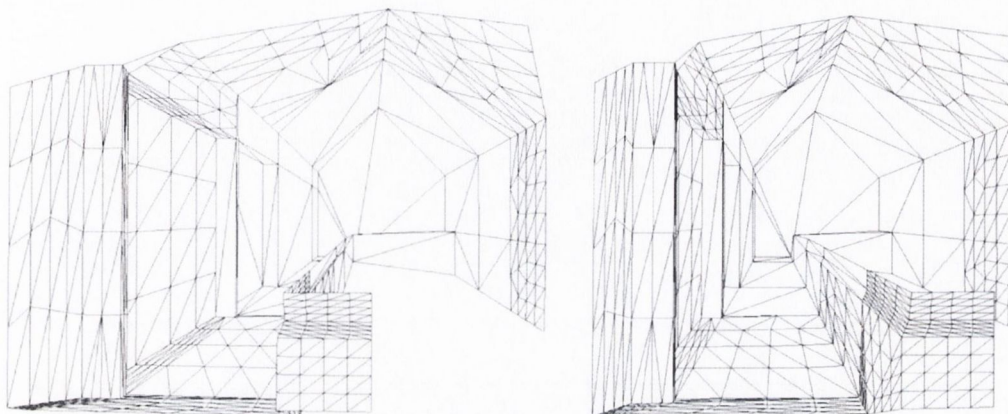


Figure 2: Geometric proxy seen from two angles and several input views for the Mezzanine scene.

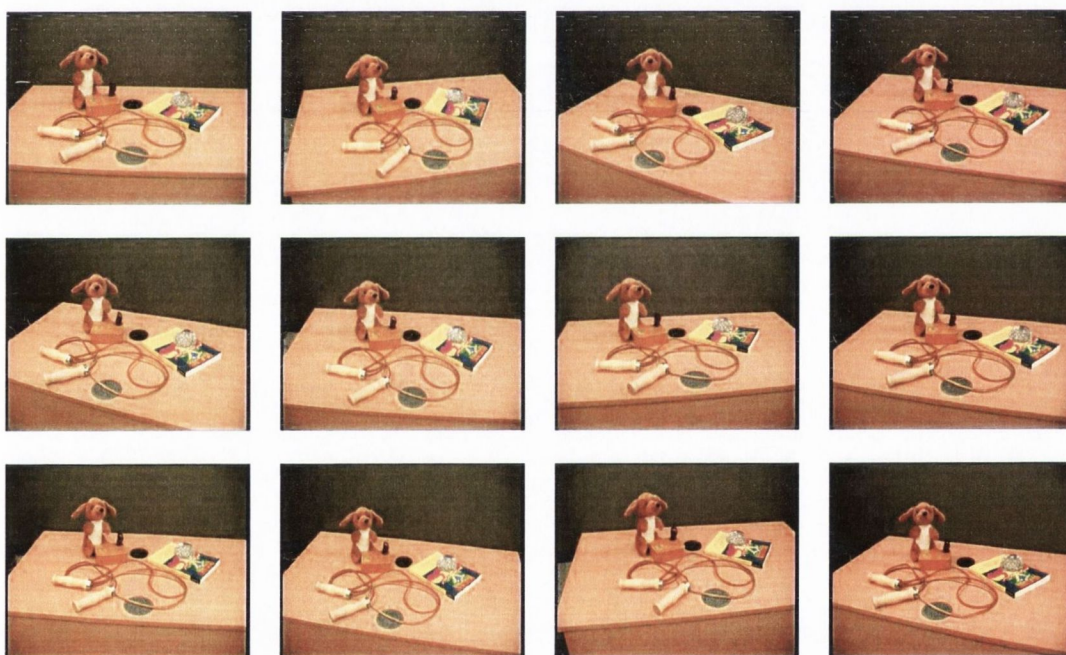
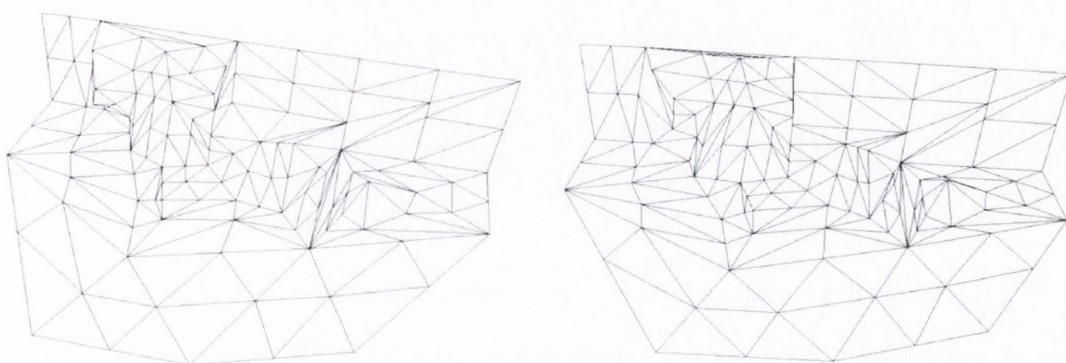


Figure 3: Geometric proxy seen from two angles and several input views for the Objects scene.

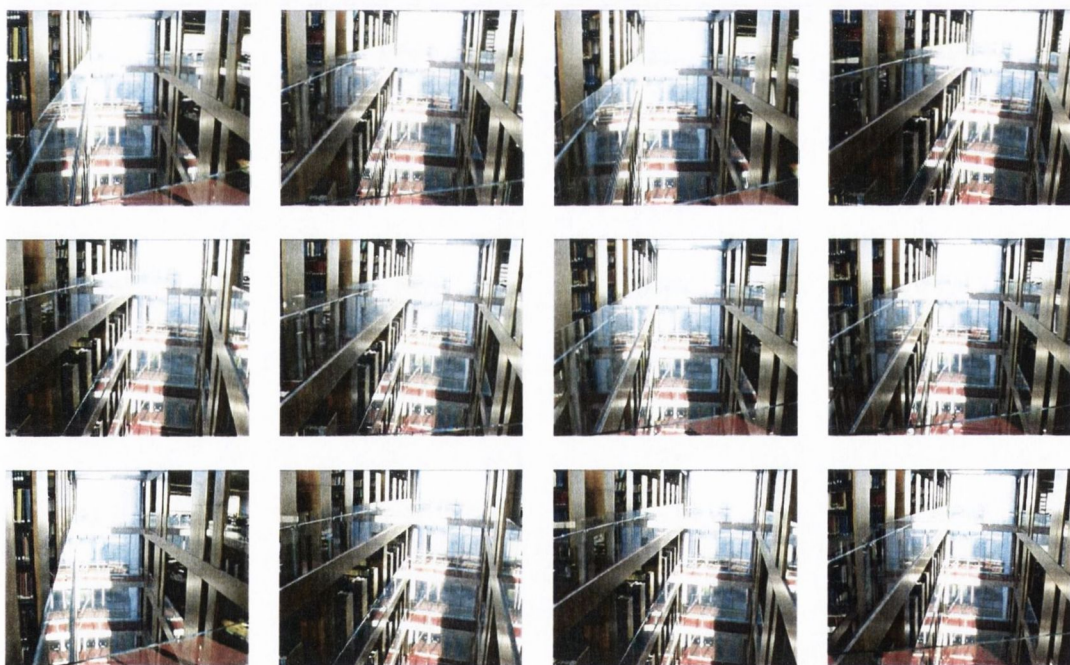
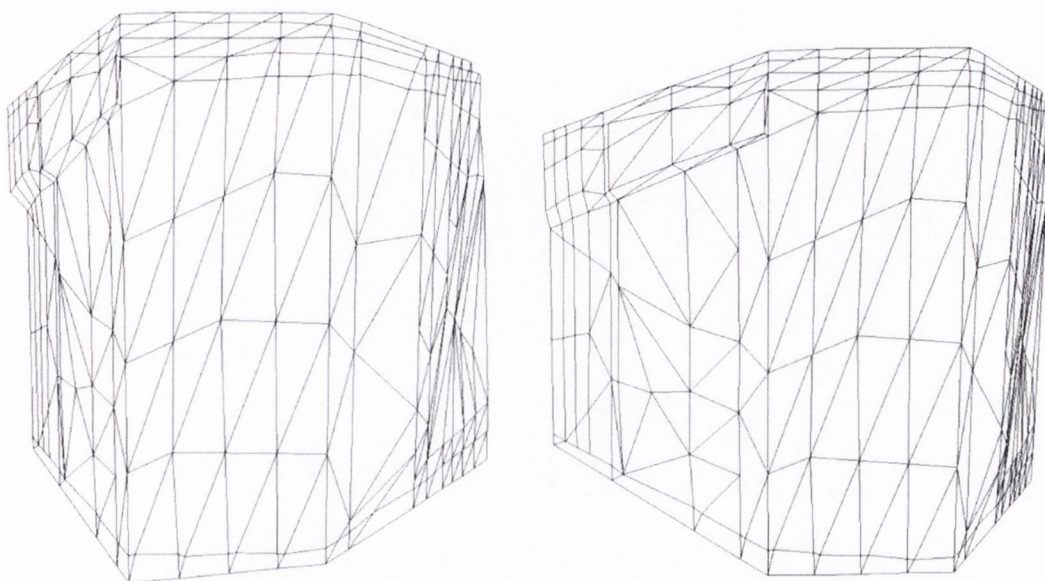


Figure 4: Geometric proxy seen from two angles and several input views for the Library scene.



Figure 5: Geometric proxy seen from two angles and several input views for the Exterior scene.

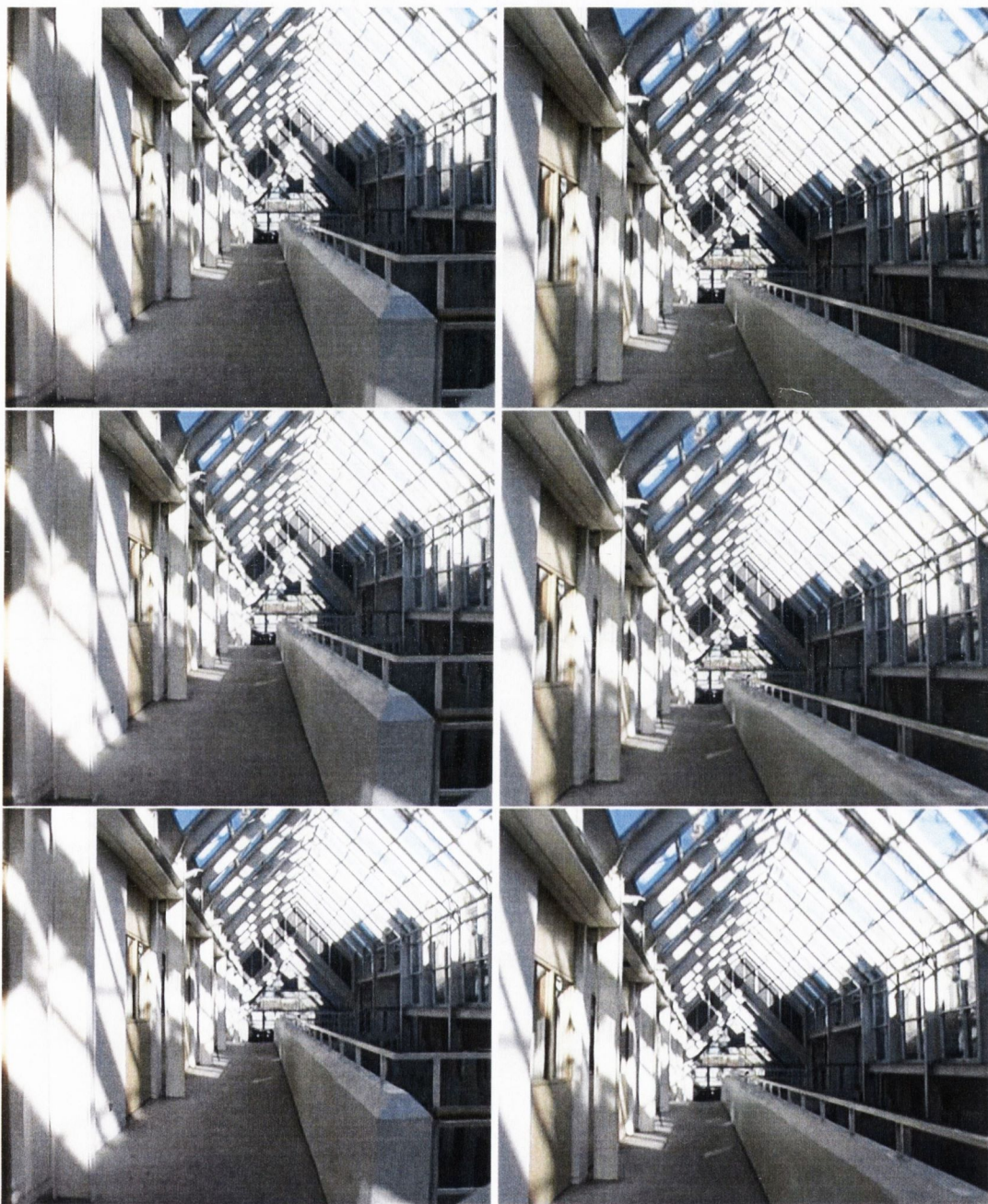


Figure 6: Novel views of the Mezzanine scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 11508 and 13008 sub-views out of 14508 according to Tables 6.1 and 6.3 of Section 6.2

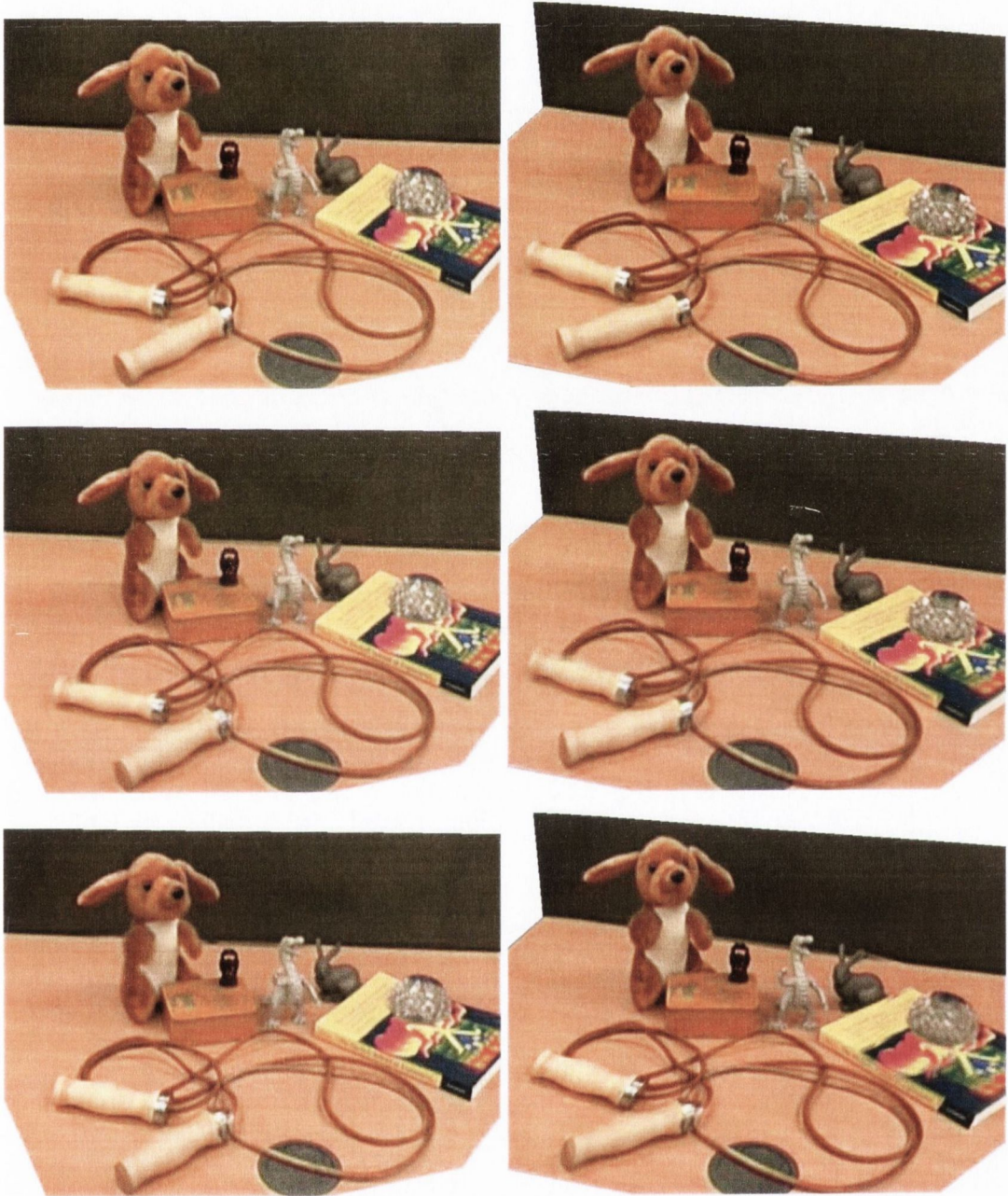


Figure 7: Novel views of the Objects scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 10674 and 15674 sub-views out of 20674 according to Tables 6.1 and 6.3 of Section 6.2



Figure 8: Novel views of the Library scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 4125 and 8125 sub-views out of 15625 according to Tables 6.1 and 6.3 of Section 6.2



Figure 9: Novel views of the Exterior scene. Each column shows a randomly chosen novel view. The first row corresponds to renderings made with the whole input dataset. Row 2 and 3 show renderings obtained after discarding respectively 7638 and 11638 sub-views out of 18638 according to Tables 6.1 and 6.3 of Section 6.2

Bibliography

- [2d3] 2d3. Boujou 4. www.2d3.com.
- [AB91] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, 1991.
- [AGJ83] S. Delman A. Gilchrist and A. Jacobsen. The classification and integration of edges as critical to the perception of reflectance and illumination. In *Perception and psychophysics*, volume 33(5), pages 425–436, 1983.
- [Ano94] Anonymous. An 'artificial spectrum top'. *Nature*, 51, 1894.
- [Aut] Autodesk. 3d studio max. usa.autodesk.com.
- [Avi] Avi Bryant. Seaside. www.seaside.st.
- [BBM⁺01] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Computer Graphics (SIGGRAPH 01)*, pages 425–432, August 2001.
- [Ble] Blender Foundation. Blender. www.blender3d.org.
- [BM95] M. R. Bolin and G. W. Meyer. A frequency based ray tracer. In *SIGGRAPH 95 Conference Proceedings*, pages 409–418, August 1995.
- [BM98] M. R. Bolin and G. W. Meyer. A perceptually based adaptive sampling algorithm. In *SIGGRAPH 98 Conference Proceedings*, pages 299–309, July 1998.
- [BSKS05] Attila Barsi, László Szirmay-Kalos, and Gábor Szijártó. Stochastic glossy global illumination on the gpu. In *SCCG '05: Proceedings of the 21st spring*

conference on Computer graphics, pages 187–193, New York, NY, USA, 2005. ACM Press.

- [CCST00] Jin-Xiang Chai, Shing-Chow Chan, Heung-Yeung Shum, and Xin Tong. Plenoptic sampling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [Che95] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conference Series):29–38, 1995.
- [CHLG05] Greg Coombe, Chad Hantak, Anselmo Lastra, and Radek Grzeszczuk. Online construction of surface light fields. In *Rendering Techniques*, pages 83–90, 2005.
- [CW93] S. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics (SIGGRAPH'93)*, pages 279–288, 1993.
- [CWE94] S. Coren, L. M. Ward, and J. T. Enns. *Sensation and Perception*, 4th ed. Harcourt Brace, New York, 1994.
- [Dal93] Scott Daly. The visible differences predictor: an algorithm for the assessment of image fidelity. In *Digital images and human vision*, pages 179–206, 1993.
- [DBY98] P. Debevec, G. Borshukov, and Y. Yu. Efficient view-dependent image-based rendering with projective texture mapping. In *Rendering Techniques 98 Proc. Eurographics Workshop on Rendering*, 1998.
- [Deb97] Paul E. Debevec. Facade: modeling and rendering architecture from photographs and the campanile model. In *SIGGRAPH '97: ACM SIGGRAPH 97 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '97*, page 254, New York, NY, USA, 1997. ACM Press.
- [Del34] B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

- [Dig] DigiLan, University of Hannover. Voodoo camera tracker. www.digilab.uni-hannover.de.
- [DPF03] Reynald Dumont, F. Pellacini, and James A. Ferwerda. Perceptually-driven decision theory for interactive realistic rendering. *ACM Transactions on Graphics*, 22(2):152–181, 2003.
- [DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996.
- [Fau93] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.
- [FCOL00] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.
- [FdMS02] M. Fiorentino, R. de Amicis, G. Monno, and A. Stork. Spacedesign: a mixed reality workspace for aesthetic industrial design. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 86–318, 2002.
- [Fec38] G. T. Fechner. Ueber eine scheibe zur erzeugung subjectiver farben. *Pogg. Ann. d. Physik u. Chemie*, 121:227–232, 1838.
- [Fec60] G. Fechner. *Elemente der psychophysik*. Breitkopf and Hartel, Leipzig, 1860.
- [Fer01] J. Ferwerda. Hi-fi rendering. In A. McNamara and C. O’Sullivan, editors, *ACM Campfire on Perceptually Adaptive Graphic, Preconference Proceedings*, 2001. <http://isg.cs.tcd.ie/gap/>.
- [Fer03] James A. Ferwerda. Three varieties of realism in computer graphics. In *Proceedings SPIE Human Vision and Electronic Imaging '03*, 2003.
- [Ger39] A. Gershun. The light field. *Journal of Mathematics and Physics*, translated by P. Moon and G. Timoshenko, 18:51–151, 1939. Originally published in Moscow, 1936.

- [GGSC96] S.J. Gortler, R. Grzeszczuk, R. Sweliski, and M.F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conferences Series, SIGGRAPH'96*, pages 43–54, 1996.
- [GJ84] A. Gilchrist and A. Jacobsen. Perception of lightness and illumination in a world of one reflectance. In *Perception*, volume 13, pages 5–19, 1984.
- [GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battale. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM Press.
- [GTLL06] Gaurav Garg, Eino-Ville Talvala, Marc Levoy, and Hendrik P.A. Lensch. Symmetric photography : Exploiting data-sparseness in reflectance fields. In *Proc. 2006 Eurographics Symposium on Rendering*, 2006.
- [Guo98] Baining Guo. Progressive radiance evaluation using directional coherence maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 255–266. ACM Press, 1998.
- [Hal98] Joshua George Hale. *Texture re-mapping for decimated polygonal meshes*. Edinburgh University, 1998.
- [HHS⁺97] John Hancock, E. Hoffman, R. Sullivan, D. Ingimarson, Dirk Langer, and Martial Hebert. High-performance laser range scanner. In *SPIE Proceedings on Intelligent Transportation Systems*, 1997.
- [HLW96] V. Hlavac, A. Leonardis, and T. Werner. Automatic selection of reference views for image-based scene representations. In *Proc. ECCV*, pages 526–535, 1996.
- [HO03] J. Hamill and C. O'Sullivan. Virtual dublin: A framework for real-time urban simulation. In *Journal of WSCG*, volume 11, pages 221–225, 2003.
- [Hub89] David H. Hubel. *Eye, Brain and Vision*. New York: Scientific American Library, 1989.
- [HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

- [Ing78] Daniel H. H. Ingalls. The smalltalk-76 programming system design and implementation. In *Conference Record of the fifth Annual ACM Symposium on Principles of Programming Languages*, 1978.
- [Int] Intel. Open source computer vision library. <http://www.intel.com/technology/computing/opencv/index.htm>.
- [JHH⁺] Simon Peyton Jones, Paul Hudak, John Hughes, , and Philip Wadler. The history of haskell (draft). www.haskell.org.
- [Joh] John W. Eaton and Many others. GNU Octave. www.octave.org.
- [Kru89] L. E. Krueger. Reconciling Fechner and Stevens: Toward a unified psychophysical law. *Behav. Brain Sci*, 12:251–267, 1989.
- [LH96] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conferences Series, SIGGRAPH'96*, pages 31–42, 1996.
- [LPC⁺00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
- [LS00] Zhouchen Lin and Heung-Yeung Shum. On the number of samples needed in light field rendering with constant-depth assumption. In *CVPR*, pages 1588–, 2000.
- [Lub95] J. Lubin. A visual discrimination model for imaging system design and development. In E. Peli, editor, *Vision Models for Target Detection and Recognition*, pages 245–283. World Scientific, 1995.
- [MB95] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics (SIGGRAPH'95)*, pages 39–46, August 1995.
- [McN00] A. McNamara. *Comparing Real and Synthetic Scenes using Judgements of Lightness*. University of Bristol, 2000. PhD Thesis.

- [McN01] A. McNamara. Visual perception in realistic image synthesis. *Computer Graphics Forum*, 20(4), 2001.
- [MEG00] M. Magnor, P. Eisert, and B. Girod. Model-aided coding of multi-viewpoint image data. *ICIP'00*, pages 919–922, 2000.
- [MG99] M. Magnor and B. Girod. Hierarchical coding of light fields with disparity maps. *Proc. IEEE International Conference on Image Processing (ICIP'99)*, 3:334–338, 1999.
- [MG00] M. Magnor and B. Girod. Model-based coding of multi-viewpoint imagery. *Proc. SPIE Visual Communication and Image Processing (VCIP-2000)*, Perth, Australia, 1:14–22, June 2000.
- [Mic92] D. Michaels. *Exploiting Continuity-in-Time in Motion Vision*. Dept. Electrical Eng. and Computer Science, Massachusetts Inst. Technology, 1992. PhD thesis.
- [Mit87] D. P. Mitchell. Generating antialiased images at low sampling densities. In *Computer Graphics, Annual Conference Series*, pages 65–72. ACM SIGGRAPH, 1987.
- [ML92] G. W. Meyer and A. Liu. Color spatial acuity control of a screen subdivision image synthesis algorithm. In B. E. Rogowitz, editor, *Human Vision, Visual Processing and Digital Display III. Proc. SPIE*, volume 1666, pages 387–399, 1992.
- [MMB97] William R. Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3d warping. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 7–ff., New York, NY, USA, 1997. ACM Press.
- [Mod] ModelFest. Modelfest. vision.arc.nasa.gov/modelfest.
- [Mys98] K. Myszkowski. The visible difference predictor: Applications to global illumination problems. In *Proceedings of the Ninth Eurographics Workshop on Rendering*, pages 223–236, June 1998.

- [NLB⁺05] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005–02, Stanford Computer Science, 2005.
- [NMA06] Vincent Nozick, Sylvain Michelin, and Didier Arquès. Real-time plane-sweep with local strategy. In *Journal of WSCG*, volume 14, 2006.
- [OBM00] Manuel M. Oliveira, Gary Bishop, and David McAllister. Relief texture mapping. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 359–368, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [PS89] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. In Jeffrey Lane, editor, *SIGGRAPH '89 Proceedings*, volume 23, pages 281–288, 1989.
- [Reaa] RealViz. Image modeller. www.realviz.com.
- [Reab] RealViz. Matchmover 4.0. www.realviz.com.
- [Ric70] J. P. Richter. *The notebooks of Leonardo da Vinci*, volume 1. New York: Dover, 1970.
- [RPG99] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of ACM SIGGRAPH 1999*, pages 73–82. ACM Press / ACM SIGGRAPH, 1999.
- [RSH02] Rajeev Ramanath, B. W. Snyder, and C. D. Hinks. Image comparison measure for digital still color cameras. In *ICIP (1)*, pages 629–632, 2002.
- [RT06] Ramesh Raskar and Jack Tumblin. Computational photography. In *State of the Art Report of the Eurographics 2006 conference*, 2006.
- [Sar77] D. J. Sarkinson. On the role of the observer and a distortion measure in image transmission. In *IEEE Transactions Communication*, volume 25(11), pages 1251–1267, 1977.

- [SCG⁺05] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R. Marschner, Mark Horowitz, Marc Levoy, and Hendrik P. A. Lensch. Dual photography. *ACM Trans. Graph.*, 24(3):745–755, 2005.
- [SDW02] Daniel Steinwand, Brian Davis, and Nathan Weeks. Geowall: Investigations into low-cost stereo display systems. In *USGS Open File Report*, 2002.
- [SFWG04] William A. Stokes, James A. Ferwerda, Bruce Walter, and Donald P. Greenberg. Perceptual illumination components: A new approach to efficient, high quality global illumination rendering. In *ACM SIGGRAPH conference proceedings*. ACM Press, 2004.
- [SGHS98] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Computer Graphics (SIGGRAPH'98)*, pages 234–242, 1998.
- [SH99] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH'99)*, pages 299–306, 1999.
- [SHS99] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. Adaptive acquisition of lumigraphs from synthetic scenes. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 151–160. The Eurographics Association and Blackwell Publishers, 1999.
- [SKC03] Heung-Yeung Shum, Sing Bing Kang, and Shing-Chow Chan. Survey of image-based representations and compression techniques. *IEEE Trans. Circuits Syst. Video Techn.*, 13(11):1020–1037, 2003.
- [Squ] SqueakFoundation. Squeak smalltalk. www.squeak.org.
- [SS00] Gideon P. Stein and Amnon Shashua. Model-based brightness constraints: On direct estimation of structure and motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):992–1015, 2000.
- [SSY⁺04] Heung-Yeung Shum, Jian Sun, Shuntaro Yamazaki, Yin Li, and Chi-Keung Tang. Pop-up light field: An interactive image-based modeling and rendering system. *ACM Trans. Graph.*, 23(2):143–162, 2004.

- [The] The Pixel Farm. Pftrack. www.thepixelfarm.co.uk.
- [TJNU97] G. A. Thomas, J. Jin, T. Niblett, and C. Urquhart. A versatile camera position measurement system for virtual reality tv production. In *Proceedings of the International Broadcasting Convention '97*, pages 284–289. IEE Conference Publication, 1997.
- [Tra91] D. Travis. *Effective Color Displays*. Academic Press, 1991.
- [Tre95] B. Treutwein. Adaptive psychophysical procedures. *Vision Research*, 35(17):2503–2522, 1995.
- [Tsa92] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Radiometry*, pages 221–244, 1992.
- [Uni] University of Southern California. Hdr shop. gl.ict.usc.edu/HDRShop.
- [VFSH01] P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference (VMV01)*, pages 273–280. IOS Press, Amsterdam, November 2001.
- [VMKK00] Valdimir Volevich, Karol Myszkowski, Andrei Khodulev, and Edward A. Kopylov. Using the visual differences predictor to improve performance of progressive global illumination computation. *ACM Trans. Graph.*, 19(2):122–161, 2000.
- [VQE00] VQEG. Final report from the video quality experts group on the validation of objective models of video quality assessment. Technical report, VQEG, 2000.
- [VWJL04] Vaibhav Vaish, Bennett Wilburn, Neel Joshi, and Marc Levoy. Using plane + parallax for calibrating dense camera arrays. *cvpr*, 01:2–9, 2004.
- [WAA⁺00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 287–296. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

- [War94] G. Ward. The radiance lighting simulation and rendering system. In *Proc. SIGGRAPH*, pages 459–472. ACM Press, 1994.
- [WBL02] Z. Wang, A. C. Bovik, and L. Lu. Why is image quality assessment so difficult? In *Proceedings of the IEEE International Conference on Acoustics, Speech, & Signal Processing*, volume 4, pages 3313–3316, 2002.
- [WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing*, volume 13 no. 4, April 2004.
- [WLLZ00] Yunnan Wu, Lin Luo, Jin Li, and Ya-Qin Zhang. Rendering of 3d-wavelet-compressed concentric mosaic scenery with progressive inverse wavelet synthesis (piws). In *VCIP*, pages 31–42, 2000.
- [XG03] Tong Xin and R.M. Gray. Interactive rendering from compressed light fields. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1080–1091, 2003.
- [XS93] Yalin Xiong and Steven Shafer. Depth from focusing and defocusing. Technical Report CMU-RI-TR-93-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1993.
- [YEBM02] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A real-time distributed light field camera. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 77–86, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [YMG02] Jingyi Yu, Leonard McMillan, and Steven Gortler. Scan light field rendering. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 137, Washington, DC, USA, 2002. IEEE Computer Society.
- [YN04] Yangli Hector Yee and Anna Newman. A perceptual metric for production testing. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 121, New York, NY, USA, 2004. ACM Press.

- [ZCS02] Li Zhang, Brian Curless, and Steven M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.
- [Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [ZL00] Cha Zhang and Jin Li. Compression and rendering of concentric mosaics with reference block codec (rbc). In *VCIP*, pages 43–55, 2000.