Publications for Task 5.2
Deliverable 5.21

Date: 06.07.2015
Grant Agreement number: EU 323567
Project acronym: HARVEST4D
Project title: Harvesting Dynamic 3D Worlds from Commodity Sensor Clouds
Document Information

<table>
<thead>
<tr>
<th>Deliverable number</th>
<th>D5.21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable name</td>
<td>Publications for Task 5.2</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Date</td>
<td>2015-07-06</td>
</tr>
<tr>
<td>WP Number</td>
<td>5</td>
</tr>
<tr>
<td>Lead Beneficiary</td>
<td>TUD</td>
</tr>
<tr>
<td>Nature</td>
<td>R</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Author(s)</td>
<td>TUD, ParisTec (Aroudj, Fuhrmann, Boubekeur)</td>
</tr>
</tbody>
</table>

Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Author</th>
<th>Org.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>07.07.2015</td>
<td>Samir Aroudj</td>
<td>TUD</td>
<td>Initial Version</td>
</tr>
<tr>
<td>0.2</td>
<td>08.07.2015</td>
<td>Simon Fuhrmann</td>
<td>TUD</td>
<td>Final Version</td>
</tr>
</tbody>
</table>

Statement of originality

This deliverable, although of public dissemination level, may at the time of delivery still contain original unpublished work (e.g., accepted papers that are not public yet, or papers under revision). Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.
# TABLE OF CONTENTS

1  Executive Summary .................................................................................................................................................. 1  
1.1  Introduction .......................................................................................................................................................... 1  
1.2  Publications ......................................................................................................................................................... 1  
2  Description of Publications ............................................................................................................................................ 2  
2.1  Overview ............................................................................................................................................................. 2  
2.2  MVE – A Multi-View Reconstruction Environment ................................................................................................. 3  
2.3  Floating Scale Surface Reconstruction .................................................................................................................. 4  
2.4  Covariance Trees for 2D and 3D Processing ............................................................................................................. 5  
2.5  Wavelet Point-based Global Illumination ................................................................................................................. 6  
3  Other Results ............................................................................................................................................................... 7  
4  Appendix ..................................................................................................................................................................... 7  

1 EXECUTIVE SUMMARY

1.1 INTRODUCTION

This deliverable describes the publications that resulted from Task 5.2 and how they fit into the work plan of the project.

The objective of Task 5.2 is to realize data structures and algorithms based on the multi-scale scene data theories devised in Task 5.1. The outcomes encompass reconstruction, suitable representation, manipulation and visualization of multi-scale graphical data.

There are four publications which are mainly attributed to Task 5.2. They can be found in the appendix of this deliverable. Additionally, there are five publications which are related to this deliverable. These are only mentioned but not discussed here since they stronger belong to other deliverables. The latter, related papers are available on the Harvest4D webpage or in other deliverables.

1.2 PUBLICATIONS

The four main publications of task 5.2 can be found in the appendix:

- Simon Fuhrmann, Fabian Langguth and Michael Goesele
  *MVE – A Multi-View Reconstruction Environment*

- Simon Fuhrmann and Michael Goesele
  *Floating Scale Surface Reconstruction.*

- Thierry Guillemot, Andrès Almansa and Tamy Boubekeur
  *Covariance Trees for 2D and 3D Processing.*
  In: Computer Vision and Pattern Recognition (CVPR, Oral), 2014.

- Beibei Wang, Xiangxu Meng and Tamy Boubekeur
  *Wavelet Point-based Global Illumination.*

The following related publications can be found on the webpage or in other deliverables:

- Ronny Klowsky and Michael Goesele
  *Wavelet-based Surface Reconstruction from Multi-Scale Sample Points.*
2 DESCRIPTION OF PUBLICATIONS

2.1 OVERVIEW

Multi-scale as well as hierarchical algorithms and data-structures for capturing, processing and rendering were devised for this work package for two reasons. First, almost all geometry acquisition devices produce samples which represent surface areas with a particular extent and thus have an inherent and variable scale value. Second, many visual applications include handling of large datasets with varying level of detail requiring usage of hierarchical spatial data structures.

In particular, we have introduced the Multi-View Environment (MVE), the Floating Scale Surface Reconstruction algorithm (FSSR), the Covariance Tree for 2D and 3D processing and a new wavelet-based hierarchical radiance cache for point-based global illumination.

- Stephane Calderon and Tamy Boubekeur
  *Point Morphology.*

- Beibei Wang, Jing Huang, Bert Buchholz, Xiangxu Meng and Tamy Boubekeur
  *Factorized Point-Based Global Illumination.*

- Murat Arikan, Reinhold Preiner, Claus Scheblauer, Stefan Jeschke, Michael Wimmer
  *Large-Scale Point Cloud Visualization through Localized Textured Surface Reconstruction.*

- Murat Arikan, Reinhold Preiner, Michael Wimmer
  *Multi-Depth-Map Raytracing for Efficient Large-Scene Reconstruction.*

- Hélène Legrand and Tamy Boubekeur
  *Morton Integrals for High Speed Geometry Simplification.*
2.2 MVE – A MULTI-VIEW RECONSTRUCTION ENVIRONMENT

![Diagram](image)

Figure 1 [Fuhrmann et al. 2014]: The MVE reconstruction pipeline main steps are estimation of cameras from input images (Structure from Motion), reconstruction of depth maps (Multi-View Stereo) and surface reconstruction (Floating Scale Surface Reconstruction).

The publication is about MVE, a complete multi-view reconstruction system which creates a colored surface triangle mesh from multiple photos. The software system implements the following main reconstruction steps. First, a Structure-from-Motion algorithm is applied to recover sparse 3D scene points and camera parameters of the input images such as camera orientations and positions. Second, a robust Multi-View Stereo algorithm is applied to compute a depth map for each input image given the estimated camera parameters. Each depth map is then converted into a 3D point cloud and merged into a single, very dense point cloud of the scene. Third, the resulting point cloud is the input to our FSSR algorithm to reconstruct a colored surface mesh.

Importantly and in contrast to most image-based geometry reconstruction approaches, MVE is focused on reconstructions from input images that show the scene at varying scale. This means that photos can be taken freely with an emphasis on more interesting regions in the scene, which is important, e.g., in cultural heritage. As a result, MVE supports large datasets consisting of both highly detailed regions of interest and coarser, less interesting parts of the scene.
2.3 FLOATING SCALE SURFACE RECONSTRUCTION

Floating Scale Surface Reconstruction (FSSR) is the final step in the MVE pipeline for multi-scale surface reconstruction. It uses a flexible, local and scalable approach, which stands in contrast to the previously published frequency-space decomposition proposed in [Klowsky et al. 2013] from deliverable 5.11.

FSSR takes as input a set of oriented samples with an additional scale value and an optional color and confidence estimate. As a very core part of FSSR, each sample point is converted into a local basis and weighting functions based on sample scale and normal. The per-sample scale value models the approximate physical extent of a sample, as samples are not infinitely small points. The weighted combination of all basis functions then defines the implicit function represented in a sparse octree data structure. Finally, a triangle mesh extracted from the implicit function via an hierarchical marching cubes algorithm.

This approach has multiple advantages. First, redundancy of sample points can be properly exploited by merging samples with compatible scale in order to remove noise while preventing detail loss. Second, the automatic scale-based approach does not require setting parameters by the user. Third, the method is local and thus scalable to large datasets because the basis functions are compactly supported.

Figure 2 [Fuhrmann and Goesele 2014]: The top row shows a multi-scale reconstruction of the Elisabeth dataset with color (left), shading (middle) and false coloring of the scale (right). The second row shows example input images.
2.4 COVARIANCE TREES FOR 2D AND 3D PROCESSING

The Covariance Tree collaboratively models high-dimensional filters by parameterizing multi-scale learned anisotropic Gaussians in a binary space partition tree. The tree is built over the domain of the function, e.g., the image space. The result of a query to the tree takes the form of a Gaussian mixture approximating the data attached to a particular region of the domain. Beyond 2D filtering and hole filling, we propose applications to 3D point-set processing, including an efficient implementation of our previous Non-Local Point Set Surfaces scheme.

![Covariance Tree](image)

Figure 3 [Guillemot et al. 2014]: Non-local Bayes surface reconstruction from a 3D point cloud using a Covariance Tree.

Gaussian Mixture Models (GMM) have become one of the major tools in modern statistical image processing and allowed performance breakthroughs in patch-based image denoising as well as restoration problems. Nevertheless, their adoption level was kept relatively low because of the computational costs involved in learning such models on large image databases. With Covariance Trees, we provide a flexible and generic tool for dealing with such models without the computational penalty or parameter tuning difficulties associated to a naive implementation of GMM-based image restoration tasks. We do so by organizing the data manifold in a hierarchical multi-scale structure -- the Covariance Tree (or CovTree) -- that can be queried at various scale levels around any point in feature-space. We start by explaining how to construct a Covariance Tree from a subset of the input data, how to enrich its statistics from a larger set in a streaming process and how to query it efficiently at configurable scale. We then demonstrate its usefulness on several applications such as nonlocal image filtering, data-driven denoising, reconstruction from random samples and surface modeling from unorganized 3D points sets. This work has led to a publication at IEEE CVPR 2014 (Oral).
2.5 WAVELET POINT-BASED GLOBAL ILLUMINATION

Figure 4 [Wang et al. 2015]: Compared to point-based global illumination (left), our hierarchical wavelet radiance cache (middle) can capture indirect non-diffuse reflections almost as accurately as Monte Carlo solutions (right) for only a fraction of the rendering time.

Point-Based Global Illumination (PBGI) is a popular rendering method in special effects and motion picture productions. This algorithm provides a diffuse global illumination solution by caching radiance in a meshless hierarchical data structure during a pre-process while solving for visibility over this cache at rendering time. This is done using a micro buffer for each receiver whereas the micro buffers are localized depth and color buffers inspired from real time rendering environments. As a result, noise-free ambient occlusion, indirect soft shadows and color bleeding effects are computed efficiently for high-resolution image output in a temporally coherent fashion. We propose an evolution of this method to address the case of non-diffuse inter-reflections and refractions. While the original PBGI algorithm models radiance using spherical harmonics, we propose to use wavelets parameterized on the direction space to better localize the radiance representation in the presence of highly directional reflectance. We also propose a new importance-driven adaptive micro buffer model to accurately capture incoming radiance at a point. Furthermore, we evaluate outgoing radiance using a fast wavelet radiance product and limit the induced larger memory footprint by hierarchically encoding the wavelets in the PBGI tree. Ultimately, our algorithm can handle non-Lambertian BSDFs in the light transport simulation and thus reproduce caustics and multiple reflections or refractions bounces. The achieved quality is similar to bidirectional path tracing in a large number of cases whereas only a fraction of its computation time is required. Our approach is simple to implement and easy to integrate into an existing PBGI framework while providing an intuitive control on the approximation error. We evaluate it on a collection of example scenes. This work has led to a publication in the journal Computer Graphics Forum (proceedings of EGSR 2015).
### 3 OTHER RESULTS

A number of other publications that appear in other deliverable make use of multi-scale scene representations. They are discussed here only briefly.

Wavelet-based Surface Reconstruction from Multi-Scale Sample Points [Klowski and Goesele 2013] is a precursory work to the floating-scale surface reconstruction and discussed in D5.11.

Point Morphology [Calderon and Boubekeur 2014] is based on a hierarchical Gaussian Mixture representation. The hierarchy is used as level-of-detail mechanism. In a similar way, Factorized Point-Based Global Illumination [Wang et al. 2013] approximates the scene with point sampling and represents the scene in a spatial hierarchy, where the radiance of inner nodes is approximated using spherical harmonics.

The papers on mapping images to geometry [Arikan et al. 2014, Arikan et al. 2015] also in essence allow a multi-scale scene representation. The fact that the geometry resolution is determined by the image assigned to an area on the surface, and that this image is chosen in an optimal way, allows representing the surface at multiple scales that correspond to the available color resolution. This is similar in spirit to FSSR, but uses a different mechanism.


### 4 APPENDIX

The following pages contain all the publications listed in Section 1.2, which mainly belong to this deliverable. Other publications, which are referenced here, can be found on the public Harvest4D web page or in other deliverables.
MVE – A Multi-View Reconstruction Environment

Simon Fuhrmann    Fabian Langguth    Michael Goesele
TU Darmstadt, Germany

Figure 1: Our multi-view reconstruction pipeline. Starting from input images, structure-from-motion (SfM) techniques are used to reconstruct camera parameters and a sparse set of points. Depth maps are computed for every image using multi-view stereo (MVS). Finally, a colored mesh is extracted from the union of all depth maps using a surface reconstruction approach (FSSR).

Abstract
We present MVE, the Multi-View Environment. MVE is an end-to-end multi-view geometry reconstruction software which takes photos of a scene as input and produces a surface triangle mesh as result. The system covers a structure-from-motion algorithm, multi-view stereo reconstruction, generation of extremely dense point clouds, and reconstruction of surfaces from point clouds. In contrast to most image-based geometry reconstruction approaches, our system is focused on reconstruction of multi-scale scenes, an important aspect in many areas such as cultural heritage. It allows to reconstruct large datasets containing some detailed regions with much higher resolution than the rest of the scene. Our system provides a graphical user interface for structure-from-motion reconstruction, visual inspection of images, depth maps, and rendering of scenes and meshes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1 Introduction

Acquiring 3D geometric data from natural and man-made objects or scenes is a fundamental field of research in computer vision and graphics. 3D digitization is relevant for designers, the entertainment industry, and for the preservation as well as digital distribution of cultural heritage objects and sites. In this paper, we introduce MVE, the Multi-View Environment, a new, free software solution for low-cost geometry acquisition from images. The system takes as input a set of photos and provides the algorithmic steps necessary to obtain a high-quality surface mesh of the captured object as final output. This includes structure-from-motion (SfM), multi-view stereo (MVS) and surface reconstruction.

Geometric acquisition approaches are broadly classified into active and passive scanning. Active scanning technologies for 3D data acquisition exist in various flavors. Time of flight and structured light scanners are known to produce geometry with remarkable detail and accuracy. But these systems require special hardware and the elaborate capture setup is expensive. Real-time stereo systems such as the Kinect primarily exist for the purpose of gaming, but are often used for real-time geometry acquisition. These systems are based on structured light which is emitted into the scene.
They are often of moderate quality and limited to indoor settings because of inference with sunlight. Finally, there is some concern that active system may damage objects of cultural value due to intense light emission.

Passive scanning systems do not emit light, are purely based on natural illumination, and will not physically harm the subject matter. The main advantage of these systems is the cheap capture setup which does not require special hardware: A consumer-grade camera (or just a smartphone) is usually enough to capture datasets. These systems are based on finding visual correspondences in the input images, thus the geometry is usually less complete, and scenes are limited to static, well textured surfaces. The inexpensive demands on the capture setup, however, come at the cost of much more elaborate computer software to process the unstructured input. The standard pipeline for geometry reconstruction from images involves three major algorithmic steps:

- Structure-from-Motion (SfM) reconstructs the extrinsic camera parameters (position and orientation) and the camera calibration data (focal length and radial distortion) by finding sparse but stable correspondences between images. A sparse point-based 3D representation of the subject is created as a byproduct of camera reconstruction.
- Multi-View Stereo (MVS), which reconstructs dense 3D geometry by finding visual correspondences in the images using the estimated camera parameters. These correspondences are triangulated yielding 3D information.
- Surface Reconstruction, which takes as input a dense point cloud or individual depth maps, and produces a globally consistent surface mesh.

It is not surprising that software solutions for end-to-end passive geometry reconstruction are rare. The reason lies in the technical complexity and the effort required to create such integrated tools. Many projects cover parts of the pipeline, such as VisualSfM [Wu13] or Bundler [SSS06] for structure-from-motion, PMVS [FP10] for multi-view stereo, and Poisson Surface Reconstruction [KH13] for mesh reconstruction. A few commercial software projects offer an end-to-end pipeline covering SfM, MVS, Surface Reconstruction and Texturing. This includes Arc3D, Agisoft Photoscan and Acute3D Smart3DCapture. We offer a complete pipeline as open source software system free for personal use and research purposes.

Our system gracefully handles many kinds of scenes, such as closed objects or open outdoor scenes. It avoids, however, to fill holes in regions with insufficient data for a reliable reconstruction. This may leave gaps in models but does not introduce artificial geometry, common to many global reconstruction approaches. Our software puts a special emphasis on multi-resolution datasets which contain both detailed and less detailed regions, and it has been shown that inferior results are produced if the multi-resolution nature of the input data is not considered properly [MKG11, FG11, FG14].

In the remainder of the paper, we will first give an overview of our system (Section 2). The practical applicability of our system is demonstrated in a hands-on guide in Section 3. We then show reconstruction results on several datasets in Section 4 and demonstrate the superiority of our pipeline in comparison to alternative state of the art software. We briefly describe our software framework in Section 5 and finally conclude in Section 6.

2 System Overview

Our system consists of three main steps, namely structure-from-motion (SfM) which reconstructs the parameters of the cameras, multi-view stereo (MVS) for establishing dense visual correspondences, and a final meshing step which merges the MVS geometry into a globally consistent, colored mesh. In the following, we give a concise overview of the process, using the dataset of Anton’s Memorial as an example for a cultural heritage artifact, see Figure 1. For a more detailed explanation of the approaches, we refer the interested reader Szeliski’s textbook [Sze10].

2.1 Structure-from-Motion

Structure-from-motion is one of the crowning achievements of photogrammetry and computer vision which started its roots in [AZB94, PKG98] and opened up to a wider audience in [PGV’03, SSS06]. In essence, SfM reconstructs the parameters of cameras solely from sparse correspondences in an otherwise unstructured image collection. The recovered camera parameters consist of the extrinsic calibration (i.e. the orientation and position of the camera), and the intrinsic calibration (i.e. the focal length and radial distortion of the lens). Although, at least in theory, the focal length can be fully recovered from the images for non-degenerate configurations, this process can be unstable. However, a good initial guess for the focal length is usually sufficient and can be optimized further.

Feature detection: First, machine-recognizable features are detected in the input images (Figure 2, left) and matched in order to establish sparse correspondences between images. Differences in the images require invariance of the features with respect to certain transformations, such as image scale, rotation, noise and illumination changes. Our system implements and jointly uses both SIFT [Low04] and SURF [BETVG08] features which are amongst the top performing features in literature.

Feature matching: The detected features are matched between pairs of images (Figure 2, right). Because corresponding points in two images are subject to the epipolar constraints of a perspective camera model [LP95], enforcing these constraints removes many false correspondences. The pairwise matching results are then combined and expanded over several views, yielding feature tracks. Each track corresponds to a single 3D point after SfM. Depending on the size
of the scene, matching can take a long time, because every image is matched to all other images, resulting in a quadratic algorithm. As an expedient, the state after matching (containing the feature detections and the pairwise matching) can be saved to file and reloaded later in case the remaining procedure is to be repeated under different parameters. This state is called the prebundle.

We also investigated in accelerating the matching time of our system. Common approaches include matching fewer features per image, reducing the number of pairs in a pre-processing step, or accelerating the matching itself using parallelism. We use a practical combination of the approaches: By matching a few low-resolution features, one can identify image pairs that potentially do not match, and reject the candidates before full-resolution matching is performed. Although low-resolution matching rejects some good image pairs, we could not observe any loss of quality in the reconstruction. It has been shown by Wu [Wu13] that this can considerably accelerate the matching time.

Incremental reconstruction: The relative pose of a good initial image pair is estimated, and all feature tracks visible in both images are triangulated. What follows is the incremental SfM, where suitable next views are incrementally added to the reconstruction, until all reconstructable views are part of the scene (Figure 3). Lens distortion parameters are estimated during reconstruction. The performance of subsequent algorithms is considerably improved by removing the distortion from the original images.

Not many software solutions for SfM have been published, probably because the theoretic concepts and algorithmic details are involved. Freely available software for this purpose includes Bundler [SSS06] and VisualSfM [Wu13].

2.2 Multi-View Stereo

Once the camera parameters are known, dense geometry reconstruction is performed. MVS algorithms exist in various flavors [SCD+06]. Some approaches work with volumetric representations [KBC12] and usually do not scale well to large datasets. Others reconstruct global point clouds, e.g. the popular PMVS implementation of Furukawa et al. [FP10]. Scalability issues further motivated work that clusters the scene into smaller manageable pieces [FCSS10]. Although PMVS is widely used, we aim at creating much denser point clouds for mesh reconstruction in order to preserve more details in the final result. A third line of work directly reconstructs global meshes [VLPK12] and couples MVS and surface reconstruction approaches in a mesh evolution framework.

We use the Multi-View Stereo for Community Photo Collections approach by Goesele et al. [GSC+07] which reconstructs a depth map for every view (Figure 4). Although depth map based approaches produce lots of redundancy because many views are overlapping and see similar parts of the scene, it effortlessly scales to large scenes as only a small set of neighboring views is required for reconstruction. In a way, this can be seen as an out-of-core approach to MVS. Another advantage of depth maps as intermediate representation is that the geometry is parameterized in its natural domain, and per-view data (such as color) is directly available from the images. The excessive redundancy in the depth maps can be a burden; not so much in terms of storage but processing power required for depth maps computation. On the positive side, this approach has proven to be capable of producing highly detailed geometry, and to overcome the noise in the individual depth maps [FG11, FG14].

2.3 Geometry Reconstruction

Merging the individual depth maps into a single, globally consistent representation is a challenging problem. The input photos are usually subject to large variations in viewing parameters. For example, some photos show a broad overview of the scene while others show small surface details. The depth maps inherit these multi-scale properties which leads to vastly different sampling rates of the observed surfaces.

Many approaches for depth map fusion have been proposed. The pioneering work by Curless and Levoy [CL96] renders locally supported signed distance fields (SDF) of the depth maps into a volumetric representation. Overlapping SDFs are averaged, which effectively reduces noise, but
also quickly eliminates geometric details if depth maps with different resolution are merged. To this end, Fuhrmann and Goesele [FG11] present a solution based on a hierarchical SDF which avoids averaging geometry at different resolutions. We use the follow-up work by Fuhrmann and Goesele [FG14]. They present a point-based reconstruction approach (Floating Scale Surface Reconstruction, FSSR), which additionally takes per-sample scale values as input. In contrast to point-based approaches that do not use scale, such as Poisson Surface Reconstruction [KH13], the method is able to automatically adapt the interpolation and approximation behavior depending on sample scale and redundancy without explicit parameter settings. An important aspect of FSSR is that it does not interpolate regions with insufficient geometric data. Instead, it leaves these regions empty which is useful for incomplete or open (outdoor) scenes. This stands in contrast to many global approaches that often hallucinate geometry, requiring manual cleanup.

In order to generate the input samples for FSSR, all depth maps are triangulated and colored using the input image. The connectivity information is used to compute a normal for each vertex. Additionally, the lengths of all edges emanating from a vertex are averaged and used as scale value for the vertex. The union of all vertices from all depth maps is then used as input to FSSR. An hierarchical, signed implicit function is constructed from the samples and the final surface (Figure 5) is extracted as the zero-level set of the implicit function using a hierarchical variant of the Marching Cubes algorithm [KKDH07].

3 Reconstruction Guide

We now demonstrate in a practical walkthrough of a reconstruction session starting with photo acquisition all the way to the final geometry. Specifically, we point out best practices and explain how to avoid common pitfalls.

Capturing photos: A dataset reconstructs best if a few simple rules are observed. First, in order to successfully reconstruct a surface region, it must be seen from at least five views. This is a requirement of the MVS algorithm to reliably triangulate any 3D position. Photos should thus be taken with a good amount of overlap. Usually, unless the dataset becomes really large, more photos will not hurt quality, but there is a tradeoff between quality and performance. As a rule of thumb, taking twice as many photos as one might think is a good idea. In order for triangulation to work, parallax is required. The camera should be re-positioned for every photo. (This is exactly opposite to how panoramas are captured, where parallax in the images must be avoided.) This is also important for SfM: Triangulating a feature track with insufficient parallax results in a small triangulation angle and a poorly conditioned 3D position. Figure 6 shows some input images of our exemplary dataset.

Creating a scene: A view is a container that contains per-viewport data (such as images, depth maps and other data). A scene is a collection of views, which make up the dataset. A new scene is created using either the graphical interface of our software, UMVE, or the command line tool makescene. Technically, the scene appears as a directory in the file system (with the name of the dataset). It contains another directory views/ with all views stored as files with the extension .mve. Creating a new scene will solely create the views/ directory for now. Importing photos will create a .mve file for every photo. This process will also import meta information from the images (EXIF tags), which is required to get a focal length estimate for every photo. If EXIF tags are not
available, a default focal length will be assumed. This, however, can lead to SfM failures if the default value is a bad guess for the actual focal length. Figure 7 shows our graphical interface UMVE after importing images into a new scene.

**Figure 7:** UMVE after creating a new scene from images. The left pane contains a list of views. A view appears in red if no camera parameters have (yet) been reconstructed. The central part is the view inspector where the individual images of the selected view can be inspected. The right-hand side contains various contextual operations, such as UI elements to start the MVS reconstruction.

**SfM reconstruction:** The SfM reconstruction can be configured and started using UMVE, or the command line tool sfmrecon. The UI guides through feature detection, pairwise matching and incremental SfM. What follows is the SfM reconstruction starting from an initial pair, and incrementally adding views to the reconstruction. Finally, the original images are undistorted and stored in the views for the next step. Figure 8 shows a rendering of the SfM reconstruction with the sparse point cloud and the camera frusta. Note how dense the frusta are spaced around the object to achieve a good reconstruction.

**Figure 8:** UMVE rendering the SfM reconstruction. The central element is the 3D scene inspector. The right-hand side offers various rendering options.

**MVS reconstruction:** Given images with camera parameters, dense geometry is reconstructed using MVS. This can be done with either UMVE or the command line tool dmrecon. The most important parameter is the resolution level at which depth maps are reconstructed. A level of 0, or $L_0$, reconstructs at the original image size, $L_1$ corresponds to half the size (quarter the number of pixels), and so one. Looking at the resolution of recent digital cameras, a full-size $L_0$ reconstruction is rarely useful as finding dense correspondences gets more difficult, often leading to sparser depth maps at much higher computational cost. Using smaller images (we often use $L_2$), the process is faster and depth maps become more complete. See Figure 9 for a depth map computed at $L_2$.

**Figure 9:** UMVE displaying the reconstructed depth map. The mapping of depth values to intensities can be controlled using the histogram in the upper right corner. Purple pixels correspond the unreconstructed depth values.

**Surface reconstruction:** The scene2pset tool combines all depth maps in a single, large point cloud. At this stage, a scale value is attached to every point which indicates the actual size of the surface region the point has been measured from. This additional information enables many beneficial properties using the FSSR surface reconstruction approach [FG14]. Then, the FSSR tools compute a multi-scale volumetric representation from the points (which does not require setting any explicit parameters) and a final surface mesh is extracted. The mesh can appear cluttered due to unreliable regions and isolated components caused by inaccurate measurements. The mesh is thus cleaned by deleting small isolated components, and removing unreliable regions from the surface. See Figure 10 for the final reconstruction.

**Figure 10:** UMVE rendering the reconstructed surface. The reconstructed vegetation has been manually cropped from the geometry with a bounding box in order to isolate the statue.

© The Eurographics Association 2014.
4 Reconstruction Results

In the following, we show results on a few datasets we acquired over time. We report reconstruction times for all datasets in Table 1. A complete reconstruction usually takes several hours, depending on the size of the dataset and the available computing resources.

**Der Hass:** The first dataset is called *Der Hass* and contains 79 images of a massive stone sculpture, see Figure 11. This is a relatively compact dataset with uniform scale as the images have the same resolution and are evenly spaced around the object. Notice that although the individual depth maps contain many small holes, the final geometry is quite complete. Here, redundancy is key as all of our algorithms are completely local and no explicit hole filling is performed.

**Notre Dame:** Next, we reconstruct the façade of *Notre Dame* in Paris from 131 images downloaded from the Internet. We demonstrate that our pipeline is well suited even for Internet images: The features we use are invariant to many artifacts in the images, such as changing illumination. The MVS algorithm [GSC*07] uses a color scale to compensate for changing image appearance and is well suited for community photo collections. The surface reconstruction [FG14] handles the unstructured viewpoints well; it will, however, not produce a particularly good model colorization. This is because the original images have non-uniform appearance and color values are computed as per-vertex averages. In Figure 12, e.g., the portal appears slightly brighter as it has been reconstructed from brighter images.

**Citywall:** We conclude our demonstration with the *Citywall* dataset in Figure 13. The 363 input images depict an old historic wall with a fountain. This dataset demonstrates the multi-scale abilities of our system. While most of the views show an overview of the wall, some photos cover small details of the fountain. These details are preserved during reconstruction yielding a truly multi-resolution output mesh.

4.1 Runtime Performance

In Table 1, we present timings for all datasets in this paper. The reconstructions have been computed on an Intel Xeon Dual CPU system with 6 × 2.53GHz per CPU. Usually 4GB of main memory are sufficient for the smaller datasets. For large datasets, we recommend at least 8 GB of main memory (such as for the *Citywall* dataset, where surface reconstruction is quite demanding). Since most parts of the pipeline are parallelized, multiple CPUs will considerably improve the computation time. Currently we do not perform computations on the GPU as only few steps of our pipeline would benefit from GPU acceleration.

All datasets listed here use exhaustive matching, however, we investigated in reducing the time for feature matching. In order to quickly reject candidates that are unlikely to match, we first match a few hundred low-resolution features instead of the full set of features. We evaluated this approach on the *Citywall*, the largest of our datasets. As can be seen in Table 1, this reduces the matching time by about 2/3, rejecting about 2/3 of all potential image pairs.
4.2 Limitations

A practical limitation in the presented system is the memory consumption in some parts of the pipeline. For example, surface reconstruction keeps all points in memory for the evaluation of the implicit function. Since our algorithm is purely local, we plan to implement out-of-core solutions to further scale the approach. Datasets with many images pose a bottleneck in runtime performance of SfM. This is because every image is matched to all other images, resulting in a quadratic algorithm in the number of images. Although low-resolution matching reduces runtime performance, our system is not limited to, but suitable for a few hundred images.

A general limitation of multi-view reconstruction approaches is the lack of texture on the geometry. Since stereo algorithms rely on variations in the images in order to find correspondences, weakly textured surfaces are hard to reconstruct. This is demonstrated in Figure 14. Although most of the depth map has been reconstructed, multi-view stereo fails on the textureless forehead in the relief.

5 Software

The principles behind our software development make our code base a versatile and unique resource for practitioners (use it) and for developers/researchers (extend it). One notable example of an extension (also relevant in the context of cultural heritage) is the texturing approach by Waechter et al. [WMG14]. The core functionality of MVE is available as a small set of easy to use, cross-platform libraries. These libraries solely build upon libjpeg, libtiff and libpng for reading and writing image formats, and do not require any other external dependencies. We strive for a user-friendly API and to keep the code size at a maintainable minimum. The correctness of many components in our code is backed by unit tests. Our GUI application requires (aside from our own libraries) the widely used QT framework for the user interface. We ship with our libraries a few command line applications for the entire pipeline to support computation on server machines without a graphical interface. MVE is tested and operational under Linux, MacOS and Windows. The source code is available from our website†.

6 Conclusion

In this paper we presented MVE, the Multi-View Environment, a free and open 3D reconstruction application, relevant to the cultural heritage community. It is versatile and can operate on a broad range of datasets. This includes the ability to handle quite uncontrolled photos and is thus suitable for reconstruction amateurs. Our focus on multi-scale data enables to put an emphasis on interesting parts in larger scenes with close-up photos. We believe that the effort and expert knowledge that went into MVE is an important contribution to the community.

† http://www.gris.informatik.tu-darmstadt.de/projects/multiview-environment/
Acknowledgements

Part of the research leading to these results has received funding from the European Commission’s FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D) and ICT-611089 (CR-PLAY), as well as the DFG Emmy Noether fellowship GO 1752/3-1.

References


© The Eurographics Association 2014
Floating Scale Surface Reconstruction

Simon Fuhrmann       Michael Goesele
TU Darmstadt, Germany

Abstract

Any sampled point acquired from a real-world geometric object or scene represents a finite surface area and not just a single surface point. Samples therefore have an inherent scale, very valuable information that has been crucial for high quality reconstructions. We introduce a new method for surface reconstruction from oriented, scale-enabled sample points which operate on large, redundant and potentially noisy point sets. The approach draws upon a simple yet efficient mathematical formulation to construct an implicit function as the sum of compactly supported basis functions. The implicit function has spatially continuous “floating” scale and can be readily evaluated without any preprocessing. The final surface is extracted as the zero-level set of the implicit function. One of the key properties of the approach is that it is virtually parameter-free even for complex, mixed-scale datasets. In addition, our method is easy to implement, scalable and does not require any global operations. We evaluate our method on a wide range of datasets for which it compares favorably to popular classic and current methods.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: Surface Reconstruction

1 Introduction

Surface reconstruction from sampled data is a long-standing and extensively studied topic in computer graphics. Consequently, there exists a broad and diverse range of methods with various strengths and weaknesses. One well-known example is VRIP [Curless and Levoy 1996], an efficient and scalable method able to create high quality models. Due to these properties, it was extensively used in the context of the Digital Michelangelo project [Levoy et al. 2000] to merge the captured range images. Since then many new techniques were developed that, e.g., use more advanced mathematical concepts, are able to smoothly interpolate holes, or employ hierarchical techniques. These approaches come, however, often at the cost of limited efficiency, scalability or certain quality issues. Moreover, they frequently treat reconstruction as completely separate from the actual sample acquisition process.

Our goal in this paper is to present a method that is able to efficiently reconstruct high quality meshes from acquired sample data even for large and noisy datasets using a virtually parameter-free method. Examples of such reconstructions from hundreds of millions of samples are the Fountain dataset (Figure 1) and the full-sized David statue (Figure 12) from the Digital Michelangelo project [Levoy et al. 2000]. Following on earlier work, we attach a scale value to each sample which provides valuable information about the surface area each sample was acquired from.

The sample scale can in general be easily derived from the acquisition process (e.g., from the sample footprint in a structured light scan or the patch size in a multi-view stereo algorithm). This definition of scale that has been used in prior work [Muecke et al. 2011; Fuhrmann and Goesele 2011]. Knowing scale allows us to reliably identify redundancy in the samples and avoid intermingling data captured at different scales (such as in multi-view stereo depth maps reconstructed from images at various distances to the geometry, as shown in Figure 1). Without scale information, datasets containing non-uniform redundancy, sample resolution or noise characteristics will, in general, lead to poor reconstructions. Many methods do adapt the reconstruction resolution to the input data in some way. These decisions, however, are often based on the density of the input data. Figure 2 shows a common case that demonstrates why density and scale are not always related: An increased sample den-
Conceptually, our method is based on reconstructing an implicit function $F$ from the input samples. $F$ has spatially continuous scale (floating scale), i.e., the scale at which surface details are represented by $F$ varies continuously as defined by the scale of the input samples. We then define a discrete, scale-adaptive sampling of $F$ and extract an isosurface corresponding to the zero-level set of $F$. The implicit function $F$ is constructed as the sum of compactly supported basis functions. But unlike, e.g., Radial Basis Functions [Carr et al. 2001] or Smooth Signed Distance Reconstruction [Calakli and Taubin 2011] our method does not require the solution of a global problem, is computationally tractable, and the implicit function can, given the samples, readily be evaluated. The compact support leads to an approach that reconstructs open meshes and leaves holes in regions where data is too sparse for a reliable reconstruction. This is useful for scenes which cannot be completely captured, such as outdoor scenes. This stands in contrast to methods such as Kazhdan et al. [2006], which perform excellent hole-filling but often hallucinate geometry in incomplete regions, requiring manual intervention.

Our contributions are:

- The reconstruction of a continuous, signed implicit function with spatially continuous scale (floating scale) using a simple mathematical formulation,
- a virtually parameter-free approach that selects the appropriate reconstruction scale and automatically adapts the interpolation and approximation behavior depending on the redundancy in the data,
- no costly aggregation of samples in a pre-processing step so that the implicit function can, given the input samples, readily and rapidly be evaluated, and
- an efficient and scalable method that does not require any global operations (such as applying graph cuts or solving large systems of equations).

In the remainder of this paper we first review related work (Section 2). We then formally introduce our surface reconstruction approach (Section 3) and perform experiments on synthetic and real-world data (Section 4). Next, we describe the isosurface extraction (Section 5) and evaluate our approach (Section 6). We finally discuss the limitations of our approach (Section 7) and conclude with an outlook on future work (Section 8).

2 Related Work

We give an overview of closely related surface reconstruction algorithms with a focus on how they handle scale, whether and which parameters they require, and to what extent they use costly global optimizations to reconstruct the final mesh.

Volumetric Range Image Processing (VRIP) [Curless and Levoy 1996] averages surfaces (regardless of scale) in a regular grid using a volumetric approach based on the signed distance function. Averaging a high resolution and a low resolution surface yields an average surface quickly blurring the high resolution information. Our method is similar in that it also uses the weighted average of locally estimated functions to define the implicit surface compactly around the input data. While VRIP’s implicit function is approximately a signed distance function, the interpretation of our function is more abstract and values do not represent distances. In contrast to VRIP, (Screened) Poisson Surface Reconstruction [Kazhdan et al. 2006; Kazhdan and Hoppe 2013] uses the density of the samples as indicator for scale. Thus a denser set of samples is assumed to originate from a surface sampled at a higher resolution. However, the sampling rate is not necessarily related to the sample resolution, and an increased sampling rate may simply be caused by data redundancy (see Figure 2). As a consequence, Poisson Surface Reconstruction starts fitting to the sample noise and hallucinates geometric detail. Mesh Zippering [Turk and Levoy 1994] selects a triangulated depth map for each surface region, eroding redundant triangles. It is worth noting that such an approach works with meshes at pixel resolution and is thus, at least in theory, able to select high resolution surface parts and could avoid averaging with low resolution surfaces. In practice Mesh Zippering is fragile and fails in the presence of noise and outliers.

Using basis functions for surface reconstruction is a common approach, e.g., for rendering of atomic structures [Blinn 1982] or in the area of mesh-free particle-based simulation [Yu and Turk 2013]. A scalar field is defined as the sum of radially symmetric or anisotropic basis functions, possibly with finite support, and triangulated or rendered at a fixed isovalue. Radial Basis Functions (RBFs) have been used for surface reconstruction from (oriented) point clouds [Turk and O’Brien 1999] but their work is limited to small problems and closed surfaces. Another inherent difficulty lies in defining off-surface constraints to avoid the trivial solution. Although advances made RBFs much more tractable to real world data in terms of size and handling of noise [Carr et al. 2001], RBF fitting is global in nature and a large linear system of equations must be solved to obtain the parameters of the basis functions. Similarly, Calakli and Taubin [2011] present a variational approach to recon-
struct a smooth signed distance function which requires the global solution of a linear system of equations.

A local approach is presented by Ohtake et al. [2003] who fit local shape functions to oriented points and employ weighting functions to blend together the local representations. The approach requires parameters such as the support radius for fitting the local shape functions and an error threshold that controls the refinement of the hierarchical decomposition. All of these parameters, as well as the choice of the local shape functions, depend on the density, redundancy and noise characteristics of the input samples. Their approach is “multi-scale” in the sense that features are reconstructed at different resolution, however, multi-scale input samples are not considered. The method is related to ours in that it constructs the implicit function as a weighted sum of local functions. In contrast, their functions fit multiple points using local shape priors over an octree hierarchy, whereas our functions are defined on a per-sample basis. Shen et al. [2004] present an approach based on an implicit moving least-squares formulation. One key distinction to [Ohtake et al. 2003] is that not only point constraints are considered when fitting the input data: Integrated constraints are used over the polygons which allows the method to either interpolate or approximate polygonal data.

Mixed-Scale: Although there exists a wealth of surface reconstruction literature, few authors consider samples at different scales as input. Integrating scale in the reconstruction process allows us to identify and use redundancy to suppress noise, and to distinguish between high and low resolution samples. Given sufficient high resolution information, any amount of additional low resolution information should not degrade the high resolution reconstruction.

Muecke et al. [2011] splat Gaussians for every input sample into a grid to produce a 3D confidence map. They use normalized Gaussians so that every sample contributes the same confidence but, depending on the scale of the sample, distribute the confidence over differently sized regions. The final surface is extracted as the maximum confidence cut through a graph defined by the grid. The downside of this approach is the unsignedness of the map, and the exact maximum of the function cannot be obtained by interpolation. The global graph cut optimization is also a limiting factor. We draw inspiration from this approach in that we also use basis functions whose size change with the sample scale. In contrast, our implicit function is signed, the zero level-set can be triangulated with voxel accuracy, and we do not require any global optimization.

Fuhrmann and Goesele [2011] present a multi-scale depth map fusion method. The distance field of triangulated depth maps is rendered into a hierarchical signed distance field and, in contrast to VRIP, only surfaces at compatible scales are averaged. Low resolution information is discarded in regions with sufficient high resolution information. The final surface is extracted as the zero level-set of the implicit function. Although our work is inspired by the same basic idea of reconstructing multi-resolution data, the approaches are quite different. Where Fuhrmann and Goesele [2011] assume triangulated depth maps with known sensor positions as input, we rely on oriented, scale-enabled surface samples. Instead of a discretized representation of the implicit function both spatially and in scale, our implicit function can be evaluated anywhere without interpolation in scale and space, solely from the input samples. Thus scale selection becomes more flexible and is not limited to neighboring octree levels. Like VRIP, Fuhrmann and Goesele [2011] cannot extract surfaces in regions without data. Our implicit function extends beyond the input samples to some degree, which enables us to fill small holes and obtain more complete reconstructions. Finally, our iso-surface extraction does not require a global Delaunay tetrahedralization, and is thus more efficient and produces meshes with fewer output triangles.

3 Floating Scale Implicit Function

In this section we describe the choice of our implicit function. We assume that N input samples are given and equipped with a position \( p_i \in \mathbb{R}^3 \), a normal \( n_i \in \mathbb{R}^3 \), \( \|n_i\| = 1 \), and a scale value \( s_i \in \mathbb{R} \). Optional attributes are the sample’s confidence \( c_i \in \mathbb{R} \) and a color \( C_i \in \mathbb{R}^3 \). We will treat color reconstruction only as subordinate aspect of our work.

In the first step an implicit function \( F(x) : \mathbb{R}^3 \rightarrow \mathbb{R} \) is defined as the weighted sum of basis functions \( f_i \). Every sample in the input set contributes a single basis function which is parameterized by the sample’s position and normal, as well as its scale value. This step does not require any preprocessing and \( F \) can readily be evaluated. The final surface is then given as the zero-set of \( F \). In order to make the approach computationally tractable, the basis function weights \( w_i \) are compactly supported such that only a small subset of all samples need to be evaluated to reconstruct \( F \) at a position \( x \in \mathbb{R}^3 \). Due to the compact support of the basis functions, the set \( \{ x \mid F(x) = 0 \} \) essentially defines a surface everywhere beyond the support of the samples. We therefore only consider the zero-level set inside the support where the weight function \( W \) is strictly positive, i.e.

\[
\{ x \mid F(x) = 0 \land W(x) > 0 \}. \tag{1}
\]

3.1 Implicit Function

Like many approaches in literature, we reconstruct a signed implicit function which is positive in front of and negative behind the surface (similar to a signed distance function). This function \( F \) is defined as a weighted sum of basis functions:

\[
F(x) = \sum_i c_i w_i(x) f_i(x) \quad \text{and} \quad W(x) = \sum_i c_i w_i(x) \tag{2}
\]

Function \( f_i \) and weight \( w_i \) are parameterized by the \( i \)-th sample position \( p_i \), normal \( n_i \), and scale \( s_i \). The optional confidence \( c_i \) essentially scales the weight function and can easily be omitted by setting it uniformly to \( c_i = 1 \). In the following, without loss of generality, we define \( f_i \) and \( w_i \) as a one parameter family of functions depending only on the scale \( s_i \) of the sample. The position \( p_i \) and normal \( n_i \) are considered by translating and rotating the input coordinate \( x \):

\[
x_i = R_i \cdot (x - p_i) \tag{3}
\]

with rotation matrix \( R_i = R(n_i) \) such that \( x \) is transformed into the local coordinate system (LCS) of sample \( i \). The LCS is defined such that the sample’s position is located in the origin and the normal coincides with the positive \( x \)-axis. Because the normal defines the LCS only up to a one dimensional ambiguity it is important that the basis and weight functions \( f_i \) and \( w_i \) are defined in a rotation invariant manner, such that the reconstruction is invariant to the choice of the LCS orthogonal to the normal. Given a rigid transformation \( T \) and reconstruction operator \( R \) acting on a point set \( P \), this property ensures that \( T(R(P)) = R(T(P)) \).

3.2 Basis Function

Similar to Muecke et al. [2011], we use basis functions that, for every sample, contribute the same “confidence”, or volume, to the implicit function. Depending on the scale of a sample, the volume is distributed over differently sized regions. As basis function \( f \) we use the derivative of the Gaussian \( f_0 \) in the direction of the normal with \( \sigma = s_i \) set to the scale of the sample. (We flip the sign of the function because it is defined to be positive in front of the surface,
i.e. in the direction of the positive x-axis.) Normalized Gaussians \( f_y, f_z \) are used orthogonal to the normal in y and z direction.

\[
f_x(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \quad f_y(x) = f_z(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{z^2}{2\sigma^2}} \quad (4)
\]

Figure 3 illustrates the function components in 1D. This yields the basis function

\[
f(x_i) = f_x(x)f_y(y)f_z(z) = \frac{x}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}(x^2+y^2+z^2)} \quad (5)
\]

The function is rotation invariant around the normal because \( f_x, f_y \) can be rewritten in terms of the distance \( \sqrt{y^2+z^2} \) to the normal. The integral of the function’s absolute value is 1 and thus every basis function contributes the same volume to the implicit function:

\[
\iiint |f(x_i)| dx_i = \int |f_x(x)| dx \int f_y(y) dy \int f_z(z) dz = 1 \quad (6)
\]

Since \( f_y \) and \( f_z \) are normalized Gaussians, their integrals are 1 by definition. We integrate the absolute function \( |f_x| \) because the point-symmetric parts cancel each other out. \( |f_x| \) does not require explicit normalization and \( \int |f_x| = 1 \). Figure 4 (left) illustrates the function in 2D.

### 3.3 Weighting Function

In the following we design a polynomial weighting function \( w \) that has compact support, falls smoothly off to zero and gives more weight to the regions in front of the surface. The justification behind this is related to free space constraints and occlusions as discussed by Curless and Levoy [1996] and Vrubel et al. [2009]: If a sample has been observed, the existence of a surface between the observer and the sample is not possible. Behind the sample, however, we cannot be sure of the existence of a surface and want to reduce the weight quickly. We observe that \( f(x_i) \) has negligible influence beyond \( 3\sigma \), and thus we chose \( 3\sigma \) as the point beyond which the weighting function vanishes. The weighting function

\[
w(x_i) = w_x(x) \cdot w_y(y) \cdot w_z(z) \quad (7)
\]

is composed of a non-symmetric component in x-direction

\[
w_x(x) = \begin{cases} \frac{1}{\sigma^3} + \frac{1}{\sigma^2} + 1 & x \in [-3\sigma, 0) \\ \frac{1}{\sigma^3} - \frac{2}{\sigma^2} + 1 & x \in [0, 3\sigma) \\ 0 & \text{otherwise} \end{cases} \quad (8)
\]

and a rotation invariant component in y- and z-direction

\[
w_{yz}(r) = \begin{cases} \frac{2}{\sigma^3} \frac{r^3}{r^3} - \frac{1}{3\sigma^2} + 1 & r < 3\sigma \\ 0 & \text{otherwise} \end{cases} \quad (9)
\]

\[
w(r) = \sqrt{y^2+z^2} \quad (10)
\]

Note that the function \( w_{x,y} \) is the positive domain of \( w_x \) where \( x \) is replaced with the distance to the normal \( r \). The individual 1D components of the weighting function are illustrated in Figure 3, and a 2D illustration is shown in Figure 4 (right).

### 4 Analysis in 2D

There are many possible choices for both basis and weighting function. We chose the Gaussian family of functions as basis function which empirically provides excellent approximation and extrapolation behavior. We will now demonstrate these properties of the implicit function on simple synthetic 2D datasets as well as real world data. In Section 6 we discuss alternative choices for both the basis function and weighting function.

We visualize the implicit function using a color mapping where positive values are colored green and negative values are blue. Bright colors correspond to small values of \( F \) (near and also far from the isosurface) and darker colors to large values, such that the isosurface is directly visible in the images. A gray color is used outside of the support, where \( W \) is zero. Samples are indicated in red.

### 4.1 Synthetic Data

When designing a reconstruction algorithm, we are concerned with the interpolation, extrapolation and approximation characteristics of the reconstruction operator. On the one hand the isosurface should pass through the input samples, in particular if the points are sparse and accurate. The implicit function should gracefully fill the gaps between the samples using smooth extrapolation. On the other hand, if many redundant and noisy point samples are available, it should approximate the samples and average out the noise instead of over-fitting the data. The presented formulation of the implicit function automatically adapts to the data as demonstrated by the following 2D experiments.

We provide 2D point samples (of a curve) with normals. In the first experiment the scale is computed for each sample as the average distance to the two nearest neighbors of the sample. We then multiply the computed scale with several factors, see Figure 5. With a small factor, the function interpolates the samples but extrapolation becomes less smooth. With an increasing factor, the reconstruction will approximate the points and provide a smoother extrapolation, but a less accurate interpolation.

In Figure 6 we consider the sampling of a function with added noise to positions and normals. The noise is uniform and about 5% of the bounding box of the samples. As we increase redundancy (adding more noisy samples while keeping the scale of the samples constant), the technique starts to increasingly approximate the data, reducing the noise, until the reconstruction converges towards the original function.
5.1 Octree Generation

In order to avoid aliasing when sampling the implicit function or evaluating the function too far from the isosurface, we set bounds on the voxel spacing according to the samples’ scale values. Recall that sample \( i \) has scale value \( s_i \), and the radius of the sample’s support is \( 3s_i \). We impose

\[
S_\ell \leq s_i < S_{\ell-1} \Leftrightarrow S_\ell \leq s_i < 2S_\ell
\]

(11)

where \( \ell \) is the octree level at which the sample will be inserted, and \( S_\ell \) is the side length of an octree node at level \( \ell \) (i.e., the voxel spacing). This forces a sample to be inserted into an octree node with a side length \( S_\ell \) of at most \( s_i \) but usually smaller:

\[
\frac{1}{2} s_i < S_\ell \leq s_i.
\]

(12)

We start with an empty octree without nodes. The first sample \( i \) is inserted in a newly created root node with a side length of \( s_i \) and centered around the sample’s position \( p_i \). When inserting subsequent samples, three cases can occur:

1. The new sample is outside the octree. In this case the octree is iteratively expanded in the direction of the new sample until the new sample is inside the octree. The sample is then inserted using cases 2 or 3.
2. The new sample’s scale is larger than the scale of the root node. Again, octree expansion is used to create new, larger root nodes until the root has a scale according to (12).
3. The new sample’s scale is smaller than the scale of the root node. In this case the tree is traversed, possibly creating new nodes, until a node with a scale according to (12) is reached.

Once a node is determined, the sample is inserted into that node.

5.2 Evaluating the Implicit Function

After inserting all samples in the octree, the octree is prepared for evaluation of the implicit function. We enforce that nodes can be classified into either inner nodes or leaves. Inner nodes have all eight children, and leaves have no children. The current octree, however, has mixed nodes where only some of the children are allocated. We make the octree regular by allocating the remaining unallocated children of nodes which are not leaves. This creates new leaves and eliminates mixed nodes.
A list of voxels (points at which the implicit function is evaluated) is created by iterating all leaf nodes. Each leaf node generates eight voxels in the corners of the node. This is a primal sampling as opposed to a dual sampling where voxels are positioned in the center of the node. Since neighboring leaf nodes share common voxels, every voxel is identified with a unique ID and inserted into a unique set. The implicit function is then evaluated at the voxel positions.

In order to evaluate the implicit function at position \( x \), we design an efficient query on the octree that selects only samples which influence the implicit function at \( x \). The octree is recursively traversed and for every node a check is performed if the node can possibly contain a sample which influences \( x \). From Equation (11) we know that node \( N \) contains samples with a scale of at most \( 2S_N \), where \( S_N \) is the side-length of \( N \). Thus, \( x \) cannot be influenced by any sample in \( N \) if

\[
\|x - \text{center}(N)\| - \sqrt{3}S_N/2 > 3 \cdot 2S_N.
\]  

The left side of the inequality is the worst case (smallest) distance from \( x \) to any point in the node, and the right side is the largest possible influence radius of a sample in \( N \), i.e., 3 times the largest sample scale \( 2S_N \). If the inequality holds, the node can be skipped without descending into child nodes. Otherwise, all samples \( i \) in the node are considered if \( \|x - p_i\| < 3s_i \). The implicit function \( F(x) \) can then be evaluated according to Equation (2) using all selected samples that influence \( x \).

**Scale Selection**: Limiting the number of samples for evaluating the implicit function will have two effects: It speeds up the algorithm, but more importantly, it can actually improve the quality of the reconstruction. On the one hand, the error to the ground truth geometry is decreased by exploiting redundancy to account for the sample noise. On the other hand, the surface error is increased by mixing samples with different scales: As the formation of a sample usually happens through some kind of integration process over a surface area, every sample corresponds to a low-pass filtered version of the original surface depending on the scale of the sample [Klowsky et al. 2012]. Mixing high and low resolution samples will thus have the effect of degrading the isosurface towards a low-pass filtered geometry. This is demonstrated with the synthetic experiment in Figure 8 (see also the supplemental material).

Our approach to this problem is based on the idea of balancing the positive effect of redundancy (Figure 6) with the negative effect of mixing high and low resolution samples (Figure 8). These two properties are orthogonal to each other: Noise reduction improves precision along the surface normal whereas low resolution samples have an impact along the tangent of the surface. Making a trade-off between the two is not straightforward. Fuhrmann and Goesele [2011] discard low resolution samples by locally selecting the highest supported resolution from the discretized scale-space representation. Similarly, we also discard low resolution samples. We do, however, not discretize scale and can therefore choose a continuous cut-off scale using the following heuristic.

To evaluate the implicit function at voxel \( x \), consider the set of samples whose (compact) support overlaps with \( x \). We now determine a cut-off scale value \( s_{\text{max}} \) and only consider samples \( i \) with \( s_i > s_{\text{max}} \) to reconstruct the implicit function at \( x \). Conceptually, we define \( s_{\text{max}} = s_g \cdot f_{\text{noise}} \). Where \( s_g \) is a reference scale and \( f_{\text{noise}} \) can be chosen according to the noise properties of the data. In our implementation, the reference scale \( s_g \) is chosen as a robust 10th percentile of those scale values affecting \( x \). (Finding the 10th percentile is a linear operation and does not require sorting all samples.) We set \( f_{\text{noise}} = 2 \) in all of our experiments.

### 5.3 Isosurface Extraction

At this point, the samples are no longer required and the isosurface can be extracted from the implicit function defined at the octree voxels. This is, however, more complicated than with a regular grid. In the regular case, each cube can be processed individually using Marching Cubes [Lorensen and Cline 1987] and the result is guaranteed to be watertight. In the case of an octree, however, different decisions are made on either side of a cube face (because of depth disparity in the octree), which leaves cracks in the surface. We use the isosurface extraction algorithm proposed by Kazhdan et al. [2007] which yields a crack-free and highly adaptive mesh directly from the octree hierarchy.

The resulting surface contains many degenerated triangles, which is typical for Marching Cubes-like algorithms. To obtain a well-behaved mesh we apply a simple cleanup procedure, see Figure 9. We first identify needle triangles, which are erased by collapsing the short edge. A check that the normals of adjacent triangles do not change too much prevents topological artifacts. Afterwards cap triangles are removed by collapsing vertices with only three adjacent faces. A final pass of needle removal is performed as new needles may be created by the previous operation. This simple procedure usually reduces the number of triangles in the mesh by about 40%.

### 5.4 Color Reconstruction

We use a simple approach to evaluate a second implicit function that yields a color value for every position \( x \). The implicit function has form (2) but uses simpler basis functions. \( f_i \) is replaced with the constant sample color \( C_i \) and the weight function \( w_i \) is replaced with a narrow 3D Gaussian with \( \sigma = 3s_i \cdot s_g \). Here, \( s_g \) is chosen so small to avoid blurring the color and to obtain a crisp texture. Although this weighting function does not have compact support, the weight evaluated at \( \pm 3s_i \) away from the sample is in the order of \( 10^{-10} \), and thus negligible.
6 Results

We perform a thorough evaluation of our approach on three types of datasets. In Section 6.1 we compare our results on controlled data with Mesh Zippering [Turk and Levoy 1994] and VRIP [Curless and Levoy 1996]. We use the Middlebury benchmark in Section 6.2 to rank our reconstruction on multi-view stereo data. Finally, in Section 6.3, we show the performance of our algorithm on mixed-scale data. For all datasets we also compare with the quasi-standard reconstruction algorithm, (Screened) Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe 2013]. Instead of comparing to an exhaustive number of algorithms, we limit ourselves to PSR as one representative algorithm that uses point density to estimate per-sample scale in the reconstruction process. Extensive comparison of PSR with other algorithms has been performed by Kazhdan and Hoppe [2013].

6.1 Range Scanner Data

The availability of both range data and final reconstructions in the Stanford Scanning Repository [2013] allow us to qualitatively compare our reconstructions with those from the website performed with Mesh Zippering [Turk and Levoy 1994] and VRIP [Curless and Levoy 1996]. We obtained the input point sets to our system by aligning the range data using transformations provided by the Stanford Scanning Repository. This yields one mesh per range scan in the global coordinate system. Normals are computed for every vertex from the adjacent triangles. The per-vertex scale value is set to the average length of all edges emanating from the vertex. Connectivity information of the range scans is discarded afterwards.

Stanford Models: Figure 10 compares several reconstructions from the Stanford Scanning Repository [2013] with our own reconstructions. The Stanford Bunny dataset contains 10 range scans, the Dragon 71 and the Armadillo a total of 97 range scans. Our algorithm is able to make use of the redundancy in the data without blurring the result, which reveals details unavailable in the Mesh Zippering and VRIP reconstructions. The surfaces created with PSR look visually very close to our reconstruction, so we omit a visual comparison here. Instead, we provide a quantitative evaluation in Table 1. For this evaluation we split the input point set and use 90% of the samples for reconstruction, and the remaining 10% of the samples to evaluate the RMS error and mean distance to the reconstructed surface. Our method shows performance on par with PSR on these datasets.

<table>
<thead>
<tr>
<th>Stanford Datasets</th>
<th>Bunny</th>
<th>Dragon</th>
<th>Armadillo</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS (PSR)</td>
<td>1.419789</td>
<td>2.950294</td>
<td>5.527238</td>
</tr>
<tr>
<td>RMS (ours)</td>
<td>1.394920</td>
<td>2.930433</td>
<td>5.439365</td>
</tr>
<tr>
<td>Mean (PSR)</td>
<td>0.970039</td>
<td>1.560911</td>
<td>1.706402</td>
</tr>
<tr>
<td>Mean (ours)</td>
<td>0.911296</td>
<td>1.512578</td>
<td>1.676785</td>
</tr>
</tbody>
</table>

Table 1: Quantitative evaluation on Stanford datasets. 90% of the samples are used for reconstruction, the remaining 10% for evaluating the mean and RMS distance to the reconstructed surface. The measurements are in units of $10^{-4}$.

Incomplete Data: We now demonstrate the behavior of our method on data with holes and boundaries. Due to the local nature of the basis functions, the implicit function is undefined beyond the support of the samples. Although the implicit function is able to close small gaps in the sampling of the surface, it does not close larger holes. Figure 11 illustrates this behavior on a single range scan of the Stanford Bunny.

Michelangelo’s David: To showcase the scalability of our approach, we reconstruct the Michelangelo’s David provided by the Stanford 3D Scanning Repository, see Figure 12. The dataset is a VRIP reconstruction of non-rigidly aligned range scans and contains a total of 472 million input samples. Although our reconstruction required a considerable amount of memory (114 GB RAM) and processing time (4 hours on a machine with 8 AMD Opteron Quad-Core processors), we were not able to process the data with PSR within a memory limit of 250 GB at any octree level larger than 11. We succeeded in running Streaming PSR [Bolitho et al. 2007] at a level of 14, which took about a day, but still resulted in a very low resolution output mesh.

6.2 Multi-View Stereo Data

Next, we evaluate our approach on multi-view stereo (MVS) data. We produce the input samples to our algorithm in the following way: A depth map is computed for every input image using the freely available MVS implementation of Goesele et al. [2007]. Similar to the range scanner data, scale is computed for every vertex
in the triangulated depth maps as the average length of all edges emanating from that vertex. But in contrast to the scanner data, every pixel in the depth map actually corresponds to a surface region larger than the pixel: Every depth value is the result of a photometric consistency optimization on patches of a certain extent, which has a (low-pass) filtering effect on the reconstructed surface [Klowsky et al. 2012]. We used a patch size of 5x5 pixels and, empirically, found that multiplying the scale with 2.5 (i.e. the “radius” of the patch) yields good results. Finally, the union of all vertices from all depth maps is used as the input point set.

The Temple Full dataset from the Middlebury benchmark [Seitz et al. 2006] contains 312 images. All MVS depth maps yield a total of 23 million input samples. Our reconstruction is available as Fuhrmann-SG14 on the Middlebury evaluation page for quantitative comparison. (Note that the final geometry does not only depend on our reconstruction technique, but also on the MVS algorithm). We visually compare our result with PSR at an octree depth of 10 in Figure 13. The PSR reconstruction looks slightly sharper around the edges but also has some geometric artifacts. In contrast to PSR, our algorithm does not require any parameter tuning.

6.3 Multi-Scale MVS Data

Our algorithm gracefully handles both clean and uniform scale datasets, but excels in handling multi-resolution datasets. In the following we perform an evaluation on a multi-scale multi-view stereo dataset where images are taken at various distances to the subject. This yields depth maps with vastly different sampling rates of the surface. In contrast to algorithms using point density, our algorithm produces sharp geometry even in the presence of many low resolution samples, and smooth results in low resolution regions. We use PSR as representative algorithm to demonstrate the shortcomings of traditional methods on multi-scale data. We then compare our results with other multi-scale approaches.

We first register the input images using a Structure-from-Motion software. Similar to the Temple Full dataset, we reconstruct dense depth maps using the MVS implementation by Goesele et al. [2007] and use the samples of all depth maps as the input to our algorithm. (More comparisons can be found in the supplemental material that accompanies the paper.)

Elisabeth Dataset: Due to technical limitations (memory consumption and processing time) with PSR, we prepared a smaller dataset called Elisabeth to perform the comparison. The dataset contains high resolution regions with detailed carvings and reliefs, as well as regions captured at a much lower resolution, see Figure 14. Although PSR at level 9 produces a smooth result in the low resolution region, it cannot reconstruct the high resolution details. PSR at level 11 reconstructs the fine details but produces a poor result in low resolution regions: It cannot reliably detect redundancy and, due to the too large octree level, reconstructs the noise in the data. A visual comparison of the reconstruction can be found in Figure 15.

Fountain Dataset: We now compare our algorithm on an MVS dataset with a much larger extent. We captured 384 photos of an old fountain yielding a total of 196 million input samples (about half the size of the David dataset). While most of the scene is captured in lower resolution, one of the two lion heads is captured with many close-up photos. Figure 16 shows some input images as well as an overview of the whole reconstruction spanning more than two orders of magnitude differences in scale. Figure 17 shows some geometric details on the fountain.
We compare our reconstruction with two other mixed-scale approaches, namely the work by Muecke et al. [2011] (SurfMRS) and Fuhrmann and Goesele [2011] (DMFusion) in Figure 18. Due to excessive use of memory with SurfMRS on the full point set, we cropped and reconstructed only the detailed region around the fountain for the comparison. Many details are lost in the SurfMRS reconstruction because the graph cut optimization often cuts through details, such as the teeth and the spout at the mouth. While DMFusion leaves small holes in the surface, our algorithm is able to deliver a watertight result. Although all algorithms managed to properly distinguish between low and high resolution regions, our algorithm achieves a more detailed yet smoother reconstruction.

### 6.4 Alternative Basis and Weighting Functions

In the following we present alternative basis and the weighting functions. In particular, we replace our basis function with signed distance ramps similar to VRIP [Curless and Levoy 1996], and we evaluate the radially symmetric B-spline used in the work of Ohtake et al. [2003] as weighting function.

**Basis Function:** An approximate signed distance function for sample \( p_i \) with surface normal \( n_i \) is given by

\[
f_i(x) = \langle x - p_i, n_i \rangle.
\]

(14)

Because this function does not attenuate orthogonal to the surface normal, it results in a smoother implicit function but less accurate sample interpolation. The integral of the function is unbounded with a constant slope in the direction of the normal, which results in large values if evaluated far away from the surface. This aspect makes the function less useful in multi-scale scenarios because low-resolution samples tend to dominate the implicit function and degrade geometric details. This is demonstrated in Figure 19, which shows a high-resolution region of a large multi-scale dataset.

**Weighting Function:** While we advocate the use of a weighting function with non-symmetric behavior in the direction of the
normal, simpler choices are possible. Ohtake et al. [2003] use the compactly supported, radially symmetric, quadratic B-spline $B(\frac{3}{\sigma} \|x\|^3 + 1.5)$ with radius $3\sigma$ and centered around the origin. For most datasets this weighting function produces very comparable results. However, similar to Vrubel et al. [2009], the non-symmetric weighting function supresses more artifacts caused by noise and outliers, as demonstrated on the Temple dataset in Figure 20.

6.5 Runtime Performance

In this section we report runtime and memory performance of our system. Table 2 lists datasets with the number of input samples, and the time required for the reconstruction. The reconstruction time is split into sampling the implicit function, which consumes most of the time, and isosurface extraction. We also report the peak memory usage of the system, which is measured as the maximum resident memory size of the process. All benchmarks are performed on a Intel Xeon Dual CPU system with 6 × 2.53GHz cores per CPU. The reported wall time for evaluating the implicit function uses all cores. Isosurface extraction, however, is limited to a single core.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Number of Samples</th>
<th>Recon. Time</th>
<th>Peak Memory</th>
<th>Output Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>362 K</td>
<td>30s + 9s</td>
<td>320 MB</td>
<td>277 K</td>
</tr>
<tr>
<td>Dragon</td>
<td>2.3 M</td>
<td>83s + 17s</td>
<td>603 MB</td>
<td>455 K</td>
</tr>
<tr>
<td>Armadillo</td>
<td>2.4 M</td>
<td>63s + 13s</td>
<td>553 MB</td>
<td>293 K</td>
</tr>
<tr>
<td>David</td>
<td>472 M</td>
<td>247m + 38m</td>
<td>114 GB</td>
<td>81.9 M</td>
</tr>
<tr>
<td>Temple</td>
<td>22.8 M</td>
<td>5m + 5s</td>
<td>1.96 GB</td>
<td>176 K</td>
</tr>
<tr>
<td>Elisabeth</td>
<td>39.3 M</td>
<td>19m + 1m</td>
<td>4.39 GB</td>
<td>2.3 M</td>
</tr>
<tr>
<td>Fountain</td>
<td>196 M</td>
<td>178m + 6m</td>
<td>19.9 GB</td>
<td>10.2 M</td>
</tr>
</tbody>
</table>

Table 2: Runtime performance for various datasets. The timings are broken down into implicit function evaluation and surface extraction. The peak memory is measured as the maximum resident memory size of the process.

7 Discussion

Our approach requires normals and scale information for every input sample. Several approaches for estimating normals have been proposed, e.g. by Hoppe et al. [1992] and Dey et al. [2005]. Scale values, however, cannot reliably be inferred without information about the formation of the samples. In the special (but unlikely) case, where the sample density is globally related to the scale of samples (which is assumed in many methods), the scale values can be computed from the sample spacing, for example using the average distance to the $k$ nearest neighbors. If this is not the case, the estimation of scale will fail, and surface reconstruction can produce undesirable results. In particular, the algorithm looses the ability to exploit redundancy for noise reduction and thus reconstructs high frequency noise, as demonstrated in Figure 2.

Although our implementation scales well to huge datasets and the runtime performance is competitive with state-of-the-art methods, sampling the implicit function is a time-consuming step because the Gaussians which we use as basis functions are expensive to evaluate. Even though increasing redundancy does not considerably increase memory consumption, it does increase computation time. The reason is that for every sampling point $x$, more samples influence $x$ and more basis functions need to be evaluated.

Due to the local nature of our algorithm, the implicit function is not defined beyond the support of the samples. Although our approach is able to close small gaps in the surface sampling, it cannot close larger holes and leaves these regions empty. This is suitable for open scenes or geometric objects which are only partially captured. On the other hand, this behavior stands in contrast to many global approaches which perform excellent hole-filling but often hallucinate low resolution geometry in incomplete regions, requiring manual intervention.
8 Conclusion

We presented a point-based surface reconstruction method that considers the scale of every sample and enables an essentially parameter-free algorithm. It can handle both very redundant and noisy as well as controlled datasets without any parameter tuning. This flexibility comes at the price of providing a scale value for every input sample, which is typically easily obtained. The method has been shown to compute highly detailed geometry, gracefully degrades given imperfect input data such as noisy points and normals, outliers, large holes or varying point density. The mathematical concept behind the approach is very simple and will likely inspire more research in this direction. For example, studying the impact of various basis functions on the reconstruction properties can lead to new reconstruction operators.

We believe that the approach is particularly well suited for out-of-core implementation and distributed reconstruction because of the local nature of our formulation. We would like to investigate this direction in future work. This opens the door for high-quality city-scale surface reconstruction projects, impossible with current state-of-the-art approaches.

Acknowledgements

Part of the research leading to these results has received funding from the FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D), ICT-611089 (CR-PLAY) and the DFG Emmy Noether fellowship GO 1752/3-1.

References


Covariance Trees for 2D and 3D Processing

Thierry Guillemot Andrés Almansa Tamy Boubekeur
Télécom ParisTech - CNRS – LTCI – Institut Mines Télécom - Paris, France
{guillemot,almansa,boubekeur}@telecom-paristech.fr

Abstract

Gaussian Mixture Models have become one of the major tools in modern statistical image processing, and allowed performance breakthroughs in patch-based image denoising and restoration problems. Nevertheless, their adoption level was kept relatively low because of the computational cost associated to learning such models on large image databases. This work provides a flexible and generic tool for dealing with such models without the computational penalty or parameter tuning difficulties associated to a naïve implementation of GMM-based image restoration tasks. It does so by organising the data manifold in a hierarchical multiscale structure (the Covariance Tree) that can be queried at various scale levels around any point in feature-space. We start by explaining how to construct a Covariance Tree from a subset of the input data, how to enrich its statistics from a larger set in a streaming process, and how to query it efficiently, at any scale. We then demonstrate its usefulness on several applications, including non-local image filtering, data-driven denoising, reconstruction from random samples and surface modeling from unorganized 3D points sets.

1. Introduction

Statistical Priors for Image Restoration witnessed two quantum leaps in the last decade: The first one, Non Local Means (NLM) [5, 4, 13], allowed to go beyond local regularization priors like TV, by observing that non-local patch-based priors enable to much better capture the self-similar structure of natural image textures. The second one, including Non Local Bayes (NLB) [9], and Piecewise Linear Estimator (PLE) [18], perfected this idea by means of a more detailed description of the manifold containing natural image patches, which turns out to be piecewise regular and low-dimensional with respect to the high-dimensional embedding patch-space.

Actual implementations of these ideas require algorithmic accelerations and model simplifications. Thus, in NLB [9] the manifold is assumed to be locally linear, and approximated by an anisotropic Gaussian model, based on the kNNs in feature space. Ideally the manifold should be estimated over the clean feature but in the absence of better information, it is approximated iteratively from the noisy features. This choice is potentially inaccurate and requires a lot of computation since a local Gaussian Model has to be learnt around each single point in the image.

Another approach has been made popular in image processing by PLE [18] which is closely related to Structured Sparsity as introduced in [12]. In this approach the problem is simplified by modelling the patch manifold as a Gaussian Mixture Model (GMM) that is fitted to the unknown restored patches via their noisy measurements and an iterative Expectation Maximisation (EM) algorithm. In this case the number of Gaussians in the model is fixed in advance to no more than two dozens, in order to keep computational complexity under control, and to ensure that initialization heuristics are sufficient to guide the EM procedure to a good local minimum of the non-convex objective function.

The purpose of this work is to provide a generic data structure which can be used to estimate the patch manifold from a big database of clean patches. Our approach can be seen as a hybrid between PLE and NLB, in the sense that it is based on a GMM (like PLE) that is locally defined in the feature space (like NLB). Contrarily to NLB there is no need to re-learn the model around each patch. Hence the number of gaussians is no longer limited to a few dozens thanks to a hierarchical data-structure that allows to: (a) quickly insert a new patch to enrich the learnt model and (b) quickly query the learnt model parameters that are pertinent around a given patch.

Experimental evidence shows that our scheme closely matches the restoration quality of top-notch state-of-the-art image restoration methods like PLE or NLB. We demonstrate that various 2D and 3D problems –usually formulated in terms of Bayesian a posteriori expectation (EAP)– can be reformulated as a posterior likelihood maximization (MAP) which can be solved by our scheme. Then we provide a few examples showing how our CovTree has an advantage when applied to several problems such as image denoising, image reconstruction, point set surfaces reconstruction.
2. Background

Notation used in the paper Let us consider a spatial domain \( S \subset \mathbb{R}^d \) and let \( \{ p_1, \ldots, p_N \} \) be a point set of \( N \) samples. To fix ideas let us consider a mapping \( f : S \rightarrow \mathbb{R} \) associating each sample \( p_i \in S \) to a value \( f_i \) of a range domain \( \mathcal{R} \subset \mathbb{R}^d \). For example, this representation includes RGB images associating to each pixel \( p_i = (x_i, y_i) \) a color value \( f_i = (r_i, g_i, b_i) \). 3D point sets defined by their spatial positions and normals \( p_i = f_i = (x_i, y_i, z_i, nx_i, ny_i, nz_i)^T \), a more complex representation used to compute bilateral filters concatenating spatial and range coordinates by \( p_i = (x_i, y_i, r_i, g_i, b_i)^T \) of an image to a color value \( f_i = (r_i, g_i, b_i)^T \) or a non local representation by replacing \( p_i \) and \( f_i \) by image patches.

High-Dimensional Filtering Linear filters such as the bilateral filter [3, 15, 17] or the non-local means filter [5] can be computed as a weighted average of values in the range domain \( \mathcal{R} \), where the weights measure the dissimilarity between points in a spatial domain \( S \), by means of e.g. a Gaussian kernel \( \phi_S \) with diagonal covariance matrix \( \Sigma \). Thus, for any \( q \in S \) the filtered signal \( f \) is defined by:

\[
\hat{f}(q) = \frac{\sum_{p_i \in S} \phi_S(p_i - q)f_i}{\sum_{p_i \in S} \phi_S(p_i - q)} \quad (1)
\]

For most applications, a naïve implementation of Eq. 1 requires a quadratic complexity and needs to be accelerated. The main idea of most acceleration techniques is to perform filtering by a linear interpolation of values computed by downsampling \( S \). For bilateral filtering, Paris and Durand [14] introduce a tesselation of \( S \) into hypercubes using a regular grid defined into the spatial 5d domain. Nevertheless, such a grid defines a lot of unecessary cells yielding the use of this approach difficult for higher dimensional applications.

Adams et al. [2] use a \( kd \)-tree dividing \( S \) into hyperrectangles depending on signal variations thus avoiding all the empty cells defined by the regular grid. Then they [1] tesselate \( S \) using simple simplicies. The filter’s response is computed by performing multi-linear or barycentric interpolation. They consider that the signal is a linear manifold and as the dimension \( d_S \) of \( S \) increases, the number of necessary cells to enclose the signal explodes.

Gastal and Oliveira [7] defines a correct downsampling of \( S \) by using non-linear manifolds. They iteratively separate samples from different populations into different clusters using recursive low-pass filters to define adaptive manifolds. Thus, the computation is performed only where needed and the filter response is computed in linear time.

Collaborative Filter When \( p_i = f_i \) is a set of patches corrupted by a Gaussian noise of variance \( \sigma_n^2 \), Eq 1 can be reformulated – in the case of NLM – in terms of Bayesian a posteriori expectation (EAP). Lebrun et al. [9] propose the Non Local Bayes (NLB) filter by replacing the EAP by a posterior likelihood maximization (MAP). Thus, in addition to a non-local mean \( \mu \), each patch is associated to a covariance matrix describing the variability of the patch group. For any noisy patch \( \tilde{p} \), an optimum \( \hat{p} \) is computed by:

\[
\hat{p} = \mu_p + \Sigma_p[\Sigma_p + \sigma_n^2I_d]^{-1}(\tilde{p} - \mu_p) \quad (2)
\]

Ideally the covariance matrix \( \Sigma_p \) corresponds to clean patches variability but, in practice, it is computed from a noisy covariance \( \hat{\Sigma}_p \) describing variability of noisy patches. Such a filter is called “collaborative” because we process all the patches group at the same time by filtering operation \( \Sigma_p = \Sigma_p - \sigma^2 I_d \). Nevertheless, this approach requires a lot of computation since a local Gaussian model has to be learnt twice around each patch in the images to ensure the correctness of \( \Sigma_p \). In practice, the estimation step is approximated by a kNN search in the feature space.

Related to the work about Structure Sparsity [12], Yu et al. [18] introduce the Piecewise Linear Estimator (PLE) that defines a general framework for solving inverse problems in imaging such as inpainting, zooming, or deblurring. The manifold of patches is approximated by a Gaussian Mixture Model (GMM) which is fitted to the unknown restored patches via MAP using an iterative Expectation Maximisation (EM) algorithm. This approach can be seen as a faster implementation of NB which strongly discretizes the manifold of patches. In this context finding a good initialization and a correct number of classes for the GMM is a quite tricky problem.

The aim of our work is to use the latest progress concerning high dimensional filtering to generalize ideas introduced by collaborative filtering. We propose the Covariance Tree, a data structure able to learn points distributions from data and to provide, for any query location and scale, an anisotropic Gaussian corresponding to the local learned distribution. In particular, the Covariance Tree provides several key benefits such as:

1. on-the-fly learning which allows, given an initial structure, to progressively refine the precision of the learned model by streaming additional data points, for a constant memory budget, accounting for a potentially high amount of data points while controlling the size of the tree. This is a key aspect of our work to recover texture details;
Figure 1. Pipeline: To fix ideas, we illustrate the different steps of our algorithm when the two sets \( \{p_i\} = \{f_i\} \). Our CovTree is based on tree main steps: (a) from a sampled data set, we build a binary tree structure from a space partitioning of \( S \) based on \( \{p_i\} \) up to a cell size \( \sigma_q \). (b) each node learns the local statistical distribution modeled by an anisotropic kernel by streaming data points through the tree by considering the \( p_i \) and summing a weighted contribution of \( f_i \) to all traversed nodes kernel, (c) for any query point \( q \in S \) and scale \( \sigma_q \in \mathbb{R} \), our CovTree models the local distribution of learned data at \( q \) at the scale \( \sigma_q \) by providing a multivariate Gaussian distribution defined from a mean \( \mu \) and a covariance matrix \( \Sigma \).

2. fast local distribution estimate, at different scales, without recomputing the data structure;

3. genericity, allowing to solve a number of 2D and 3D processing problems by instantiating our structure with specific spatial and range domains, which includes Non-Local Bayes filtering, data-driven image denoising, image holes completion and 3D Non-Local Point Set Surface reconstruction.

3. The Covariance Tree

Assume that we want to restore a data point \( f \in \mathcal{R} \) that is either noisy or incomplete in some way. We also have access to \( q \in S \) which is related to \( f \) in the following manner: the prior distribution of \( f \) given \( q \) can be modelled as a multivariate Gaussian \( \mathcal{N}(\mu_q, \Sigma_q) \) with parameters varying smoothly as a function of \( q \).

If we know the mapping \( q \mapsto (\mu_q, \Sigma_q) \), and the degradation model (given as the conditional probability of the degraded or incomplete \( f \) given the clean \( f \)) then we can use standard Bayesian techniques (such as MAP or EAP) to estimate \( f \) from the degraded pair \( (q, f) \). In general such a mapping is unknown, but we can estimate it from a large database of examples \( \{(p_i, f_i)\} \in \mathcal{R} \times \bar{S} \subset \mathcal{R} \times S \), by averaging a sufficiently large number of neighbours:

\[
\begin{align*}
\mu_{q,\sigma_q} &= \sum_{p_i \in \mathcal{R}} \phi_{\sigma_q}(||p_i - q||) f_i \\
\Sigma_{q,\sigma_q} &= \sum_{p_i \in \mathcal{R}} \phi_{\sigma_q}(||p_i - q||) f_i f_i^T
\end{align*}
\]

(3)

where \( f_i = (f_i - \mu_{q,\sigma_q}) \). A small scale parameter \( \sigma_q \) provides a better localized prior, but larger values are often required to provide a statistically significant estimation of the local Gaussian distribution from the given examples, especially when the dimensions \( d_S \) and \( d_R \) are large, as is the case in non-local patch-based filtering or restoration for instance.

Unfortunately, high-dimensional restoration problems require large example databases and a brute-force approach to the computation of Eq. (3) at each query point \( q \) is untractable. We tackle this problem by introducing an hierarchical data structure which summarizes and indexes the database at multiple scales, with a limited amount of memory.

This structure is equipped with a fast query mechanism providing an approximation to (3), for any query point \( q \) and for a large range of scales \( \sigma_q \). Our basic idea is to model the database as an hierarchical set of anisotropic multivariate Gaussians approximating smoothly and progressively the distribution of \( \{f_i\} \) in the database. At each level of the structure, the kernels are formed by the mean and the covariance of the local distribution. Consequently, we name our structure Covariance Tree (or cov-tree).

More precisely, a cov-tree is a Binary Space Partition Tree [6] (bsp-tree) carrying anisotropic Gaussians learned from data on its nodes. It defines a rotation-invariant tessellation of \( S \) into space cells, as well as an hierarchical

\footnote{Note that if this assumption holds, then the data \( f_i \) will be more compactly represented by multivariate Gaussians than by simpler isotropic ones (see Figure 2). Indeed, by using isotropic Gaussian we consider that \( f \) is propagating the same way in every direction of the space defining smaller and more numerous cells and decreasing the quality of the learning.}
3.1. Building the tree

Instead of using a kd-tree [2], which fails at capturing anisotropy accurately (see Fig. 3(a)), we give to our cov-tree a bsp structure based on the direction of maximum variation of the input points. Let us consider a tree node \( \eta \) and its associated set of points \( \{ p_j \} \). We compute and store in \( \eta \) a splitting plane \( \{ \eta_d, \eta_c \} \), with its normal \( \eta_d \) defined as the normalized eigenvector associated to the largest eigenvalue of the covariance of \( \{ p_j \} \) and its center \( \eta_c \) defined as the average of the \( p_j \). The cell radius is given by \( \eta_c = \max \| p_j - \eta_c \| \). We then subdivide \( \{ p_j \} \) into two distinct sets based on their signed distance to the plane \( \{ \eta_d - \eta_c \} \). Finally, we construct the two children of \( \eta \), by summing a weighted contribution of \( f \) to all traversed nodes’ kernels (Sec. 3.2);

\[ \mu_\eta := \mu_\eta + w_i \xi_i \]
\[ \Sigma_\eta := \Sigma_\eta + w_i \xi_i^2 \xi_i \]

Each point we stream during the learning step increases the precision of our cov-tree without adding additional nodes (i.e., constant memory cost). For a large database, the tree nodes’ kernels are typically learned over the full data while the tree’s structure is built from a subset only. Interestingly enough, the dataset used for the building step can be different from the learning one.

3.2. Learning local distributions

Once the tree structure is initialized, we can compute the statistics of its cells by streaming pairs of training data \( \{ p_i, f_i \} \) through it. Starting from the root node, a pair is classified top-down using \( p_i \) and enriches each traversed node \( \eta \) by summing \( f_i \) to its local distribution using a weight \( w_i \) defined from a Gaussian \( \phi_\eta \), centered at \( \eta_c \):

\[ \mu_\eta := \mu_\eta + w_i f_i \]
\[ \Sigma_\eta := \Sigma_\eta + w_i f_i^2 f_i \]

The result of the successive steps of building, learning and querying approximates the estimation of covariance matrices with a Gaussian kernel of size \( \sqrt{2\sigma_q} \).

3.3. Querying local distributions

Once fully built, the tree can be queried using any \( q \in S \), providing an anisotropic Gaussian describing the distribution of the learned values around \( q \) at scale \( \sigma_q \). To do so, we first collect a set of tree nodes in the vicinity of \( q \), at different scales, by traversing the tree top-down, gathering every node \( \eta \) intersecting the \( [q, \sigma_q] \) ball and being either a leaf or verifying \( \eta_c > \sigma_q \). Traversing the tree to a finer precision level \( \eta_q << \sigma_q \) would increases the precision, but increases significantly the computational cost (see Fig 4). Second,
Figure 4. Our fast query limits the number of cov-tree nodes gathered (in green) to reconstruct a local anisotropic distribution. (a) When the requested scale $\sigma_q$ is large, only the top nodes are retained. (b) As $\sigma_q$ gets smaller, the gathering shaft gets thinner, collecting deeper nodes mostly.

we estimate the distribution at $q$ as a weighted combination of the gathered nodes’ distributions, using weights $w_i$ defined from a Gaussian $\phi_{\sigma_q}$ centered on $q$:

$$
\begin{align*}
\hat{\mu} &= \frac{1}{\hat{w}} \sum_i w_i \mu_i \\
\hat{\Sigma} &= \frac{1}{\hat{w}^2} \sum_i w_i \Sigma_i - \hat{w} \hat{\mu}^T \hat{\mu}
\end{align*}
$$

(5)

Where $\hat{w}$ and $\hat{w}^2$ are two normalization values.

### 3.4. Complexity Analysis

We recall that $S$ is the $d_S$-dimensional spatial domain and $R$ is the $d_R$-dimensional range domain. Each of the $N_b$ points used during the building step appears only once in the $K_b$ cov-tree clusters, resulting in a building cost of $O(d_S N_b d_R \log(K_b))$. Using $N_l$ points to learn the nodes’ statistics, classifying them has a cost of $O(N_l d_S \log(K_l))$ and learning variations in all nodes has a cost of $O(N_l \log(K_l))$, resulting in a learning cost of $O(N_l \log(K_l)(d_S + d_R^2))$. When querying $N_q$ points, gathering the lists of $K_q$ contributing nodes has a cost of $O(N_q d_S K_q)$; the additional anisotropic Gaussian estimation ($O(N_q K_q d_R^2)$) leads to a total querying cost of $O(N_q K_q (d_S + d_R^2))$. Last, the memory cost of our cov-tree is $O(N_b K_b d_R^2)$ and remains constant during the online steps (learning and querying), allowing to re-learn and/or reuse numerous times a cov-tree precomputed once.

### 4. Applications

We provide a few example applications of our cov-tree to solve inverse problems in 2D imaging and 3D rendering. Whenever possible the results are compared with state of the art techniques for the same problems.

The performance numbers reported in this paper were measured on a 2.4 GHz Intel Xeon processor with 12 GB of memory and 8 cores, but running a non-optimized C++
5.5
10
128 256 512 1024
Time (s)

Figure 5. Computation time (in seconds) of $5 \cdot 10^5$ queries excluding the building step which is run only once) of our cov-tree at different scales $\sigma_q$. Time is measured by specifically constraining the use of a single core.

code\(^3\) which most often takes advantage of only one core.

4.1. Non Local Bayes Filtering

Non-local Bayes denoising [9]\(^4\) is a bayesian MAP estimation from noisy image patches $\hat{p}$.\(^5\) The prior multivariate Gaussian model for the clean patch is estimated from a neighborhood $N_p$ of noisy patches, so the estimated covariance matrix $\Sigma_p$ is also corrupted by noise. When we combine the denoising of this covariance matrix with the MAP estimation in equation (2), we obtain the estimated (denoised) patch as

$$\hat{p} = \mu_p + (\Sigma_p - \sigma_n^2 I_d) \Sigma_p^{-1} (\hat{p} - \mu_p)$$

(6)

The neighborhood $N_p$ is not only restricted to patches $\hat{p}_i$ that are close to $\hat{p}$ in the feature space ($R = \mathbb{R}^{3n^2}$ for $n \times n$ patches of color images), but also to those associated to nearby pixels $(x_i, y_i)$. This restriction makes the search for similar patches tractable in very large images, since the search region does not grow with image size. Using the cov-tree, however, we can afford such large search regions without any significant performance penalty.

We propose three variants of NLB denoising which only differ in the choice of the neighborhood $N(\hat{p})$:

Global Search. The spatial domain $S = R = \mathbb{R}^{3n^2}$ is the same as the feature space, so the neighborhood $N_p$ only depends on $\hat{p}$.

Local Search. In order to reproduce the neighborhood of the original NLB we augment the spatial domain with the pixel coordinates $(x_i, y_i)$, so $S = R^{d_S}, d_S = 3n^2 + 2$.

Compressed Local Search. As an adaptation of [16] to NLB, we can reduce the spatial domain $S$ to the first

\[^{3}\]The querying process can be parallelized by using one thread for each query. A similar solution can be used for the learning process but care must be taken to avoid concurrent writing to the same cell. Parallelizing the building step is more involved but has less impact since this is the least time-consuming part of the pipeline.

\[^{4}\]for simplicity we only consider NLB’s first stage here

\[^{5}\]where noise is zero-mean Gaussian with variance $\sigma^2$

Figure 6. PSNR (with respect to ground-truth) of NLB denoising using our cov-tree with PCA dimensionality reduction in the spatial domain. We measured the PSNR\(_i\)(\(d_S\)) on 20 test images $i$ for different values of the spatial dimension $d_S$. The mean curve $\text{PSNR}(d_S)$ is represented in red and in gray the standard deviations of the centered curves $\text{PSNR}_i(d_S) - \text{PSNR}_i$. Although the absolute levels of PSNR present a large variation (between 29 and 34 dB) among all images, the shape of the $\text{PSNR}_i(d_S)$ curve as a function of $d_S$ is the same for all images and shows a peak around $d_S = 4$ in agreement with the findings of [16] for NLM.

6 principal components in a global PCA analysis of all noisy patches in the image.

Figure 7 presents the denoising results of the original NLB compared to our local variant based on the cov-tree. Our local approach (e) produces better results than the original NLB (d). We can explain it by our use of the approximate Gaussian kernel $\phi$ as a weighting function. In the original NLB, means and covariance matrices are estimated by averaging the $k$ nearest neighbors with a constant weight. Consequently, covariance matrices can be more strongly affected by outliers.

As shown in Figure 6, PCA dimensionality reduction over the $S$ domain not only accelerates the search (as expected), but also produces better performances. The latter can be attributed to the denoising effect of dimensionality reduction, which improves the relevance of the computed neighborhoods $N(\hat{p})$.

Figure 5 shows that the time required to solve a query in the cov-tree is only mildly affected by the neighborhood size $\sigma_q$ of the query. Indeed, as shown in Figure 4, coarse-sized queries explore a lot of nodes in breadth but stop the search at the top of the tree while a fine-sized query explores fewer nodes in breadth but explores the tree in depth. Consequently, the number of nodes (and hence the computational complexity) for each radius size is equivalent, except for medium sized queries which involve the largest number of nodes.

4.2. Data Driven Image Denoising

In [11] the use of huge image databases is advocated as a way to learn the prior underlying natural image patches. Their procedure (called shotgun NLM in [10]) was shown to serve as a way to estimate the fundamental limits of nonlocal image denoising methods, but no attempt was done to make the computation time actually tractable for real appli-
Figure 7. Non-local filtering: We use our CovTree to learn $7 \times 7$ RGB patches extracted from the noisy image (b). The original patch dimensionality (147-D) is reduced to 6-D using PCA as suggested for NLM by [16]. When compared to an accelerated implementation of NLM [7] (c) or the original NLB (d), our filter (e) better recovers features thanks to the use of an approximate Gaussian kernel $\phi$.

Figure 8. Data driven image denoising: We use our CovTree to learn from about $10^8$ clean $7 \times 7$ RGB patches extracted from an image database (first line) to denoise an image corrupted by a noise of standard deviation of 0.1 (b) by exploiting the prior underlying natural images. Compared to the original image (a), our data driven filter (e) better preserves features than using the original NLB filter (c) or our single-image CovTree-NLB filter (d).

A similar idea was proposed in [19] for a PLE-like algorithm with 200 Gaussians, thus requiring many days to learn the prior on a largedatabase. In this section we apply the same idea to implement a “shotgun NLB” denoising algorithm, but using the cov-tree to make both learning and restoration computationally tractable with large learning databases.

Indeed, we propose to use an external noiseless image database instead of noisy patches to denoise a noisy image. This idea has two main benefits: we can increase the number of learned patches (without a significant penalty in computation time or memory use) and data is not degraded by noise, thus increasing the precision of the estimated anisotropic Gaussian.

In practice, we build the tree at a scale $\sigma_b = \sigma_n$ by considering the colored noised patches (without the pixel coordinate) to fix the hierarchy of spatial domain cells. Then, we learn the corresponding range-domain Gaussian models from the database of noiseless colored patches. Finally, the covariance matrix $\Sigma_{\tilde{p}}$ and the mean vector $\mu_{\tilde{p}}$ are estimated from a noisy patch $\tilde{p}$ with $\sigma_q = \sigma_n$. In contrast to the previous section, the estimated anisotropic gaussians $\mathcal{N}(\mu_{\tilde{p}}, \Sigma_{\tilde{p}})$ are noiseless, consequently, we applied equation (2) directly, without need for denoising the covariance matrix.

Figure 8 shows our result of denoising an image of a façade using a database of noiseless images of similar façades in the same city (but not the same façade). As expected, the database-driven denoising performs better. More extensive experimentation is needed to check if this reconstruction is actually close to the fundamental limits announced in [11].

Our database contains about $10^8$ patches and the learning phase takes about 5 hours and 8GB of RAM to hold the data-structure. This is several orders of magnitude faster than the times reported in [11, 19], while our database is also larger. Reconstruction takes about 5 minutes.

4.3. Reconstruction from random samples

One of the most visually striking applications of Bayesian MAP estimation with a Gaussian mixture prior model for image patches, is the reconstruction of an image, from a small random subset of its pixels (20% in our case), as showcased in [18] among others. Let’s consider a patch $q$ sampled by a known random sampling operator $\tilde{q} = S q$. As before, we consider that image patches are locally modeled as an anisotropic Gaussian distribution, with mean $\mu_{\tilde{q}}$ and covariance $\Sigma_{\tilde{q}}$ estimated from the local dictionary. We assume that $\tilde{q}_0$ is an initial estimate of the complete patch (obtained by a cubic interpolation over the Delaunay triangulation). This patch serves as a query to extract a local statistical prior for $q$ from the cov-tree. Then by Bayesian
Figure 9. Data driven reconstruction from sparse samples: We obtain from an original image (a) a masked image by extracting randomly 20% of the original pixels (b). By learning about $10^6$ patches with our CovTree, we reconstruct an image (d) by applying Eq. (7) with a coarse-to-fine query scales, and starting from a cubic interpolation (c).

$$q_{i+1} = \left(\Sigma_{q_i} S^H S + \frac{\sigma_i^2}{2} I_d\right)^{-1} \left(\Sigma_{q_i} S^H q_i + \frac{\sigma_i^2}{2} \mu_{q_i}\right)$$

Combining all reconstructed patches by aggregation, we obtain a first reconstruction $I_1$ from the interpolated image $I_0$. The patches $q_i$ in this first reconstruction ($i = 1$) can in turn be used as an initialisation/query to obtain more accurate prior and reconstruction $q_{i+1}$. We iterate the process with until $q_i = q_{i+1}$ by using a coarse-to-fine query scale $\sigma_i = \alpha^i \sigma_0$ where $\alpha \in [0, 1]$ is a scale factor. Figure 9 presents the result of our approach over an image were only 20% random samples were retained. In our experiments $\alpha = 0.8$ and $\sigma_0 = 0.65$.

5. Conclusion and perspectives

We proposed a novel data structure and associated algorithms that allow to deal with a continuous family of local multivariate Gaussian models: both efficient learning over a large database and fast querying are supported. The extracted model varies continuously over a range domain $R$ when the query point varies over a (possibly different) spatial domain $S$, and different scale levels can be specified by the query resulting in various degrees of spatial locality of the extracted statistical model.

The relevance of such a data structure is motivated by image restoration problems via Bayesian MAP with local Gaussian priors on the image patches. Such models became increasingly successful in image processing during the last 5 years, because they are very close to an accurate statistical model of natural images. However, progress in this area has been slow because we lack the required tools for efficiently handling the massive amounts of data that need to be fed to learn these models in order to approach optimal results. The cov-tree is an attempt to fill this gap.

Given the genericity of the cov-tree, its applicability reaches potentially far beyond image restoration. In the supplementary material we include an application that improves an NLM-like algorithm for denoising and interpolation of 3D point clouds [8]. In this case a data structure is definitely required to index and query a set of patches even for point cloud of $10^6$ points. The reason is that common acceleration and search techniques that are used by non-local methods in 2D image processing do not apply to irregular 3D point sets over a surface that has no implicit parameterization.

Acknowledgements This work has been partially funded by the EC under contracts FP7-287723 REVERIE, FP7-323567 HARVEST4D, by the French government under ANR iSpace&Time, FUI CEDCA and CNES R&T 128435 projects.

References

Wavelet Point-Based Global Illumination

Beibei Wang† † † Xiangxu Meng† Tamy Boubekeur∗

†Telecom ParisTech - CNRS - Institut Mines-Telecom  ∗Shandong University

Figure 1: Compared to traditional PBGI (a), our approach (b) models accurately non-diffuse indirect lighting effects and appears as an efficient substitute to bidirectional path tracing (c), trading a moderate image degradation for up to a 10x speed-up in our experiments.

Abstract

Point-Based Global Illumination (PBGI) is a popular rendering method in special effects and motion picture productions. This algorithm provides a diffuse global illumination solution by caching radiance in a mesh-less hierarchical data structure during a pre-process, while solving for visibility over this cache, at rendering time and for each receiver, using microbuffers, which are localized depth and color buffers inspired from real time rendering environments. As a result, noise free ambient occlusion, indirect soft shadows and color bleeding effects are computed efficiently for high resolution image output and in a temporally coherent fashion. We propose an evolution of this method to address the case of non-diffuse inter-reflections and refractions. While the original PBGI algorithm models radiance using spherical harmonics, we propose to use wavelets parameterized on the direction space to better localize the radiance representation in the presence of highly directional reflectance. We also propose a new importance-driven adaptive microbuffer model to capture accurately incoming radiance at a point. Furthermore, we evaluate outgoing radiance using a fast wavelet radiance product and contain the induced larger memory footprint by encoding hierarchically the wavelets in the PBGI tree. As a result, our algorithm can handle non-lambertian BSDF in the light transport simulation, reproducing caustics and multiple reflections/refractions bounces with a similar quality to bidirectional path tracing in a large number of cases and for only a fraction of its computation time. Our approach is simple to implement and easy to integrate into any existing PBGI framework, with an intuitive control on the approximation error. We evaluate it on a collection of example scenes.

1. Introduction

Over the last decade, global illumination (GI) has become a standard requirement for almost any industrial computer graphics production, from movie special effects to motion pictures and TV shows. Among the vast repository of rendering algorithms that support (at least a subset of) GI effects, Point-Based Global Illumination (PBGI) is certainly one of the most widely used solution. The basic idea of PBGI is to decorrelate the scene complexity from the GI computation by substituting a shaded point cloud to the scene when computing any indirect lighting effect, only keeping the polygonal representation for direct visibility from the camera. This algorithm is particularly efficient at simulating a large number of diffuse effects which are visually appealing in GI, including ambient occlusion, indirect soft shadows and color bleeding. The algorithm itself can be easily adapted to compute multiple light bounces, subsurface scattering and more general volumetric effects while staying fast, embarrassingly parallel and finely tunable by artists. However, although final receiver surface samples may be shaded using any BRDF, the indirect lighting effects captured by standard PBGI are restricted to diffuse inter-reflections and do not allow to re-
produce caustics for instance, implying alternative GI solutions (e.g., Monte Carlo Ray Tracing, Photon Mapping) to tackle these cases.

PBGI At Glance The PBGI rendering algorithm [Chr08] can be summarized in two main stages: radiance caching and many-views rasterization. During the first stage, the virtual scene is sampled into a dense point cloud, each point is shaded from the primary light emitters and a hierarchical data structure (e.g., BVH) is generated to store them, propagating the leaf/point radiance estimates to the inner nodes in a bottom-up fashion. The resulting PBGI tree can therefore be seen as an hierarchical “spatial cache” of the scene’s radiance. In the second stage, the indirect illumination of each receiver (e.g., unprojected image pixel) is evaluated by localizing a color+depth hemispherical buffer, called microbuffer (or MB), at the receiver position and filling it by rasterizing the PBGI tree onto it. This rasterization resembles the hierarchical Z-buffer algorithm, only substituting the PBGI tree to the actual scene and filling a specific MB for each single receiver. The mesh-less nature of the PBGI tree eases the extraction of a receiver-dependent adaptive level-of-detail (LoD) of the scene as a “cut” in the tree, reducing drastically the computational footprint of the many rasterizations required to fill the MBs of all the pixels in a high resolution image. The final receiver radiance (e.g., pixel color) is obtained by convolving the filled MB color component with the receiver reflectance distribution.

Limitations One key feature of PBGI is to reproduce noise-free diffuse GI effects efficiently by modeling radiance in the PBGI tree nodes using spherical harmonics (SH). Unfortunately, even using a larger number of coefficients, SH are not able to capture high frequencies efficiently, which precludes non-diffuse reflections or refractions in our case. Consequently, caustics stemming from metals, plastics, glass and other reflective or refractive materials are not handled with classical PBGI frameworks. Even when ignoring the performance issue induced by the larger number of SH coefficients, ringing artifacts quickly appear and the second step of the algorithm remains flawed: as the receiver color is evaluated by convoluting its BRDF with a discretized incoming radiance (i.e., the MB), the case of glossy to nearly specular reflections cannot cope with the typical low resolution of the MB and again, the intrinsic speed of PBGI vanishes when increasing the MB size, causing additional artifacts as well.

Overview In this paper, we introduce Wavelet PBGI (or WPBGI) to address the problem of non-diffuse indirect lighting effects efficiently (Sec. 3).

First, we propose a new outgoing radiance model which captures directional reflections using Haar wavelets warped over the space of directions (Sec. 3.1). Our model characterizes efficiently, at each node of the PBGI tree, the localized regions on the (hemis)phere which exhibits high frequency outgoing radiance. In order to drastically shrink the significantly larger memory footprint induced by such an approach, we propose to store the radiance wavelets hierarchically, expressing these nodes attributes w.r.t. to their (average) parent one.

Second, when rasterizing the tree for a particular MB at rendering time, our wavelet hierarchy allows to compute adaptive importance-driven cuts with very fine structures (e.g., specular spots) next to coarser ones. Consequently, we introduce an adaptive MB model which drives the LoD extraction and exhibits a spatially varying resolution based on the classical geometric factors (distance, incidence angle) as well as on appearance parameters, such as the estimated incoming radiance and the BSDF glossiness (Sec. 3.2). We additionally extend this MB model to the full spherical domain to support refractions.

Third, we propose a multi-bounce indirect lighting solution, exploiting the hierarchical structure in a form of generalized rasterization, where the PBGI tree is splatted onto itself to emulate the light propagation (Sec. 3.3). Additionally, we use a fast wavelet product to quickly evaluate outgoing radiance from the adaptive incoming one and the BSDF.

As a result, our improved PBGI algorithm allows to render images with caustics effects and many flavors of indirect non-diffuse effects, up to close to perfectly specular as well as refractive materials, for only a fraction of the cost of a bidirectional path tracing solution, with a similar quality (Sec. 4).

2. Previous Work

PBGI. PBGI was first proposed by Christensen [Chr08] to evaluate diffuse light transport by substituting a mesh-less hierarchy to the scene and using a z-buffer inspired approach to solve for visibility at each receiver/pixel. The notion of point-based substitutes was actually introduced by Bunnell [Bun05] for real time ambient occlusion and indirect illumination. Ritschel et al. [REG09] proposed warped microbuffer instead of cube ones to allow for importance sampling based on the receiver BRDF, with an efficient GPU implementation. Holländer et al. [HREB11] further improved fine-grained parallelism of the adaptive cut computation. A number of approaches have been proposed to improve the cut computation, including importance-driven point projection based on an initial clustering [MW11], cut picking algorithm for HDR imaging [Tab12], and tree-cut/microbuffers factorization based on spatial coherence [WHB13]. The PBGI memory issue has been tackled with an out-of-core framework for PBGI, providing a cache-coherent tree construction and traversal [KTO11], and with an in-core solution which quantizes all tree nodes against a small set of representatives, learned on-the-fly [BB12].

Wavelets for Rendering. Wavelets have been extensively used to represent light transport functions [GSCH93,
CSSD95, KTHS06], to approximate environment maps [NRH03], to evaluate, through importance sampling, products of complex functions [CJAMJ05], such as measured BRDFs and distant lighting and to model the multiple components of a rendering pipeline, including lighting, reflectance and visibility [NRH04, SM06]. In combination with other techniques [LSSS04], such as clustered principal component analysis for instance, further compression can be achieved. In our work, we use the off-the-shelf Haar basis, which is easy to implement and met our expectations at capturing high frequencies. However, the wavelets we describe in this paper could be further explored using alternative basis e.g., tuned for the spherical domain [SS95]. Yet, performances may suffer beyond simple basis.

Other Related Work. (Hemi)Spherical harmonics construct low-frequency functions efficiently and have been used to store irradiance environment maps [RH01], transfer and lighting functions [SKS02], incoming [KGBP05] and outgoing radiance [Chr08]. However, for high frequency BRDF or lighting, they require a large number of coefficients and suffer from "ringing" artifacts. Spherical Gaussians have also been widely used to represent light transport functions [TS06, GKM06, LWDH10] and BRDFs [WRG*09, XSD*13]. Lehtinen et al. [LZT*08] proposed a hierarchical mesh-less basis to represent diffuse light transport. Alternatively, Keller et al. [Kcl97] proposed to approximate indirect lighting using many virtual point lights. Walter et al. [WFA*05] improved this approach by organizing them into a binary tree and defining lightcuts, which are similar in essence to the cuts used in PBGI.

3. Wavelet PBGI

In this section, we describe the WPBGI components that differ from the traditional PBGI algorithm and refer to the notations listed in Tab. 1. An overview of the WPBGI work flow is given in Fig. 2. We start by recalling the radiance equation with direct lighting:

\[ L_o(x, \omega_i) = \int_{\Omega_2} L_i(x, \omega_i) \rho(x, \omega_i, \omega_o) V(x, \omega_o)(\omega_o \cdot n) d\omega_o. \]  

(1)

3.1. Wavelet Radiance Model

To properly capture \( L_o(x, \omega_i) \), the radiance distribution of non-diffuse surfaces, we need a compact representation that can describe high frequency variations efficiently, spending only few numbers to model large, slowly varying regions. Wavelets do have this property. Therefore, we propose to model the radiance distribution of each PBGI sample/tree leaf using Haar wavelets. To do so, at caching time, the distribution is estimated for all samples and expressed as a collection of two-dimensional functions \( L_o(j) \), where \( j \) is the index of the point. Each of these functions is projected onto the Haar basis \( \Psi = \{ \psi_i \} \) by writing it as a series of expansions:

\[ L_o(j) = \sum c_i \psi_i, \]

with \( C = \{ c_i \} \) representing the vector of coefficients in the basis. To evaluate the coefficients, \( L_o(j) \) is sampled in a cube map yielding six small (typically 32 × 32) images for each point sample. We then Haar-transform each image and store the resulting coefficients in a tree structure similarly to Sun et al. [SM06]; the coefficients of the mother scaling basis function are located at the root and each node contains three detail coefficients and children indices. When the detail coefficients of a node and of all its children fall below a threshold, the node is discarded, which ultimately leads to a sparse representation. This method also allows on-demand point-wise reconstruction (i.e., compressed radiance estimation at the point for a given direction) and avoids decompressing the entire wavelet for each single direction query. This compressed radiance \( \tilde{L}_o(j) \) is reconstructed as follows:

\[ \tilde{L}_o(j) = C_i \psi_i. \]

Hierarchical Wavelet Coding

With our radiance model in hand, we can now build the PBGI tree in a bottom-up fashion, as a complete binary bounding sphere hierarchy, with its leaves being formed by the points of the initial scene sampling and its internal nodes providing economic substitutes for their related sub-trees, including average position/normal, bounding radius and radiance distribution. We observe that Haar wavelets support linear operators and choose earlier to express our wavelet radiance models in the

<table>
<thead>
<tr>
<th>Table 1: Notations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x, n )</td>
</tr>
<tr>
<td>( L_o, L_i )</td>
</tr>
<tr>
<td>( \rho )</td>
</tr>
<tr>
<td>( \omega_i, \omega_o )</td>
</tr>
<tr>
<td>( V )</td>
</tr>
<tr>
<td>( T_2, T_4 )</td>
</tr>
<tr>
<td>( U_i )</td>
</tr>
<tr>
<td>( D )</td>
</tr>
<tr>
<td>( W )</td>
</tr>
<tr>
<td>( L_{o,j}, F_j )</td>
</tr>
<tr>
<td>( \tilde{L}_{o,j}, \tilde{F}_j )</td>
</tr>
<tr>
<td>( \Psi )</td>
</tr>
<tr>
<td>( C_i )</td>
</tr>
<tr>
<td>( G_{i,j}, H_{i,j} )</td>
</tr>
<tr>
<td>( j_l, j_r )</td>
</tr>
<tr>
<td>( P_j )</td>
</tr>
<tr>
<td>( s_j )</td>
</tr>
<tr>
<td>( m )</td>
</tr>
<tr>
<td>( \mathcal{F}, \mathcal{F}_p, \mathcal{F}_l )</td>
</tr>
<tr>
<td>( \gamma )</td>
</tr>
<tr>
<td>( M )</td>
</tr>
<tr>
<td>( \omega_k )</td>
</tr>
</tbody>
</table>

© 2015 The Author(s)  
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.
global coordinate system. Therefore, the radiance approximation for an internal node $j$ can be obtained directly by averaging its children coefficients:

$$C_j = \frac{C_j + C_{j'}}{2}.$$  

Taking inspiration from progressive coding schemes, we drastically reduce the memory footprint of our WPBGI tree by encoding the tree itself in a wavelet fashion, expressing the radiance wavelet coefficient vector of a node w.r.t. to the coefficient vector of its parent. In our binary tree setup, this boils down to a 1D Haar transform decomposing $C_j$ into an average vector $G_j = (G_j + G_{j'}) / 2$ and a detail vector $H_j = (G_j - G_{j'}) / 2$.

At caching time, we can retain only $G_0$ (the root average coefficient vector), discard the other average vectors and store only the detail vectors in the nodes, ignoring the ones with a $L^2$ norm smaller than a user-defined compression threshold (set to 0.002 in our experiments). To avoid storing the entire list of nodes vectors at any intermediate state, we compute this compressed representation during a post-order depth-first traversal of the PBGI tree [BB12]. At rendering time, we reconstruct the radiance coefficient vector of a given node $j$ as:

$$C_j := G_{\text{root}} + \sum_{k \in P_j} s_k H_k.$$  

Since wavelet compression provides domain-localization, evaluating radiance in a given direction does not require de-compressing the full coefficient vectors.

Assuming a full wavelet decomposition of the tree has obvious impact on reconstruction time. This pitfall can be greatly weakened by performing the compression only partially, retaining the average vector $G$ up to a prescribed level $m$ (set to 20 in our experiments). This allows reconstructing the radiance in constant time for any node upper than level $m$ and significantly shrinking the reconstruction time for the other nodes by boot-strapping it directly from the level-$m$ ancestor $r$ of the node:

$$C_j := G_m + \sum_{k \in (P_j \setminus P)} s_k H_k.$$  

3.2. Importance Driven Microbuffer

In the PBGI framework, the color response at a particular receiver $x$ and in a particular direction $\omega_o$ is evaluated using a microbuffer (MB) of $M$ pixels, which turns Eq. (1) into:

$$L_o(x, \omega_o) = \sum_{k=0}^{M} L^k_o \rho(x, \omega_k, \omega_o) V(x, \omega_k)(\omega_k \cdot n).$$

Note the slight abuse of notation here: $L^k_o$ stands for the radiance, emitted toward $x$, by a node projecting on the MB pixel $k$ (i.e., $\omega_k$ direction). As $V(x, \omega_k)$ is solved by the MB depth component, we ignore this term and retain the simpler form:

$$\bar{L}_o = \sum_{k=0}^{M} KL^k_o \rho,$$

where $K$ represents the other terms.

In the context of practical non-diffuse reflections, most of the incident energy is concentrated around a few dominant incoming directions, causing salient light accumulation patterns. To properly capture these caustic effects without resorting to a tremendous number of MB pixels (see Fig. 3.c), we propose Adaptive MBs (or AMB) driven by an importance function accounting for both the reflectance distribution at the receiver (see Fig. 3.e) and the incoming lighting (see Fig. 3.g). To cope with the refractions of transparent materials, we implement our AMBs as full spherical color+depth quad trees, considering the tree leaves as AMB pixels. For each receiver, an AMB is initialized uniformly (e.g., $16 \times 16$) and immediately refined according to the reflectance distribution before being progressively updated based on the incoming lighting during the receiver-dependent tree cut extraction (traversing and splatting). To do so, when the receiver is a final image pixel (i.e., unprojected image pixel). In both cases, importance driven cuts are gathered from the tree for each receiver using an adaptive microbuffer.
We compute \( F \) similarly to Ritschel et al. [REG*09] but using the subdivision of our quad tree structure instead of warping (see Fig. 3.d). The second term is defined as:

\[
F_i(q) = \begin{cases} 
1 & \text{if } \frac{L_i(q)}{2^\Lambda q} > \gamma, \\
0 & \text{otherwise,}
\end{cases}
\]

with \( \Lambda q \) the refinement level of \( q \) (see Fig. 3.f) and the threshold \( \gamma \) typically set to 1. When the receiver is not a final image pixel (i.e., multiple bounce simulation at caching time), we simplify the importance function to:

\[
\mathcal{F}(q) := F_i(q).
\]

Similar to the original approach of Christensen [Chr08], we handle multiple indirect light bounces by treating the PBGI leaves (i.e., original point samples) as receivers. For the sake of clarity, let’s consider in the following that every node in the PBGI tree can have two distinct roles at the same time: sender and receiver. Taking inspiration from hierarchical radiosity methods and observing that the traversal cost can be amortized over senders contributing similarly to several receiver leaves, we consider pairs of nodes \( S/R \), with \( S \) (resp. \( R \)) a node acting as the sender (resp. receiver). To avoid performing a complete cut search for each single receiver leaf, we seek for pairs \( N_l/N_j \) such that \( N_i \) contributes similarly to all the leaves which have \( N_j \) as ancestor, so that \( AMB_{N_j} \) can be factorized among all of them. To do so, for each pair, we consider the sender-receiver solid angle \( \alpha_{SR} \) and the receiver-sender solid angle \( \alpha_{GS} \). Assuming the wavelet radiance coefficients of the PBGI tree have been initialized from direct lighting as described in Sec. 3.1, we summarize our multi-bounce method in Alg. 1. The threshold \( \varepsilon \) (0.05 in all our experiments) allows to control the accuracy of the multi-bounce propagation and may be increased between each bounce. Note that the test \( \alpha_{GR} < \varepsilon \) may be replaced by a more accurate stopping predicate, at the cost of longer computation.

**Algorithm 1** One iteration of multi-bounce propagation.

Require: \( Q \) a queue of node pairs
Require: \( T \) an initialized PBGI tree
Require: \( \varepsilon \) a user defined threshold

1. \( \text{Root} \rightarrow \text{Root} \)
2. While \( Q \) not empty do
   1. Pop head of \( Q \)
   2. If \( R \) is a leaf or \( \alpha_{GS} < \varepsilon \) then
      1. If \( S \) is a leaf or \( \alpha_{SR} < \varepsilon \) then
         1. Splat \( S \) in \( AMB_R \)
      2. Else
         1. \( S \leftarrow R \rightarrow Q \)
         2. \( S^+ \leftarrow R \rightarrow Q \)
      End if
   3. Else
      1. \( S^+ \leftarrow R \rightarrow Q \)
      2. \( S^- \leftarrow R \rightarrow Q \)
   End if
3. Update \( C_i \) from \( AMB_j \)
4. End for
5. Update \( T \) bottom-up

**Wavelet Radiance Product.** Representing radiance at a point in a wavelet format motivates us to perform the entire radiance estimation in the wavelet domain, which is several times faster than performing the convolution in the spatial domain. Indeed, the reflectance distribution \( \rho(\omega_i, \omega_o) \), expressed in local coordinate with the \( z \) axis aligned to the normal direction, can be represented by a collection \( U \) of 4D wavelet coefficients:

\[
U = T_4 \rho(\omega_i, \omega_o).
\]
Similarly, the incoming radiance at a point can be wavelet-transformed and expressed as 2D coefficients:

\[ W = T_2 L_0(\omega_i). \]

Thus, the 2D wavelet transform \( D \) of the target outgoing radiance is directly obtained with:

\[ D = UW. \]

Consequently, the actual outgoing radiance \( L_o \) is retrieved with the inverse transform of \( D \):

\[ L_o = T_2^{-1} D. \]

Note that, since \( D \) is expressed in local coordinates, we first decode it from the frequency domain to the spatial domain, and sample it in a cube map which is then, as mentioned earlier, wavelet-transformed. Note also that both \( U \) and \( W \) are represented as sparse trees [SM06].

4. Results

We implemented our WPBGI algorithm in the Mitsuba Renderer [Jak10]. We use tile rendering and run one thread per tile. At construction time, the top part of the tree is processed sequentially, until enough subtrees are generated to run multiple threads. Comparisons are performed against (i) a simple PBGI implementation [Chr08] (PBGI) with internal nodes storing a single diffuse radiance value, (ii) PBGI with a large number of SH coefficients to model radiance (SHPBGI), (iii) progressive photon mapping [HOJ08] (PPM), (iv) primary sample space Metropolis Light Transport (MLT) [KSJAC02] and (v) bidirectional path tracing (BPT), which we consider as ground truth. Performances are measured on a 2.67GHz Intel i7 (8 cores) with 9GB of main memory. Images are rendered at a 1024 \( \times \) 768 pixels resolution (except for the Ring, Cornel Box and Sphere scenes,

![Figure 4: Comparison between simple PBGI, PBGI with AMB (with the MSE to PBGI) and WPBGI.](image1)

![Figure 13: MB resolution (a) without and (b) with AMB.](image2)
Comparison to Simple PBGI. We start by comparing to simple PBGI in Fig. 4. As expected, WPBGI produces convincing caustics effects, while PBGI only supports diffuse ones. Surprisingly, beyond the reproduction of these phenomena, the global WPBGI performance also shows a significant speed-up, up to one order of magnitude, thanks to our importance driven AMBs. However, this comes at the price of a larger memory footprint (twice on average). When enabling AMB with PBGI (Fig. 4), we observe a significant performance improvement, for a comparable quality, which underlines the usefulness of AMBs, even without WPBGI.

Comparison to Other Methods. In Fig. 6, 7, 8, 9 and 10 we evaluate the accuracy and the visual quality of our approach against BPT and PPM, observing in each case almost invisible differences with BPT, which is typically an order of magnitude slower than WPBGI. We also observe that PPM...
suffers from noise, in particular in the vicinity of shiny reflections, when restricting it to similar rendering time as the ground truth. Additionally, we compare to a degraded BPT (DBPT) solution, using less samples to reach similar rendering time as WPBGI; we can easily detect an abundant amount of noise in each case. In Fig. 11, we can observe that our WPBGI is significantly faster than MLT for a similar quality, with an even slightly smoother result. However, we suspect MLT shall provide better results in more complex lighting environment. In Fig. 12, we vary the BRDF glossiness of the object and observe a satisfying behavior of WPBGI up to close to perfectly specular settings, producing images free from visual artifacts, with a preserved speed-up. This excludes the last example, for which the perfectly specular BRDF underlines the limits of WPBGI.

Performance and Timings. In Tab. 2, we report timings and measured errors for the different examples illustrating this paper. The resolutions ($\omega_{\text{res}}$) used to sample the outgoing radiance prior to wavelet transform correspond to the size of a single face of the cube map and the memory value is the peak memory usage during the whole process. The pre. time value indicates the point cloud generation and WPBGI tree construction time, while the render time value includes the microbuffer initialization, WPBGI tree traversal, nodes splatting and final pixel color evaluation. Here, we can assess the benefits of our approach, with a speed-up ratio for the total time ranging from 2.9 to 10.1 compared to the BPT solution, while the MSE is negligible. Fig. 14 summarizes the memory usage statistics of the WPBGI tree and the timings for the Corner scene (5M pts) when varying the bootstrap value $m$. Our hierarchical coding approach shows here its capacity to contain the memory footprint, with an effective speed-up for a moderate memory cost up to level 16. Beyond this value, WPBGI keeps going faster but the memory usage grows significantly, establishing $m$ as a natural time-versus-memory control parameter. We also depict the gain of our adaptive scheme by visualizing the actual per-receiver MB resolution when enabling/disabling it in Fig. 13, using a low value for diffuse receivers. The adaptive scheme took 1.63 hours to render while the non-adaptive one took more than 24 hours. Last, WPBGI inherits the temporal coherence of PBGI: we demonstrate this behavior in the accompanying video.

Limitations and Future Work. There are several limitations to our approach that can trigger future work. First, the resolution of the cube map used to sample the outgoing radiance could be made adaptive itself. Second, tailoring the MB refinement based on radiance intensity can lead to unnecessary fine resolutions, with a moderate impact on the final image. Although we experimented with other metrics (e.g., radiance derivative on the MB domain), we did not find a better solution so far and we believe there is a large space for improvement here. Third, our importance driven method does not account for occlusions: a node may be considered important even if it is later occluded during the cut rasterization in the MB. Better predicting occlusion during the importance estimation would reduce the computation time significantly. Last, we use Haar wavelet to represent outgoing radiance, which are easy to implement and to use but requires an intermediate cube map. Ideally, a unified radiance/microbuffer model would provide a more elegant solution to the multiple bounces setting and could avoid the use of an intermediate cube map if designed on top of spherical wavelets [SS95].

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.
5. Conclusion

We have proposed Wavelet PBGI, an evolution of the PBGI algorithm able to simulate the indirect lighting induced by non-lambertian materials. First, we have introduced a new wavelet radiance model capturing efficiently the high frequency outgoing signal stemming from the point samples and structuring them in a compact, wavelet-inspired tree hierarchy. Second, we solved accurately for visibility over this new multi-resolution radiance cache by introducing an adaptive importance-driven microbuffer model. Third, we described an economic strategy to simulate multiple indirect light bounces, using a fast wavelet radiance product. As a result, our approach competes with bidirectional path tracing, reproducing faithfully the popular physically-based rendering features, such as caustics and multiple bounces, while providing a speed-up ranging from 3x to 10x, without noticeable image degradation. Our approach is easy to implement in any PBGI framework and has a reduced set of intuitive control parameters.

Acknowledgements. We thank the reviewers for their feedback. This work has been partially supported by the European Commission under contracts FP7-323567 HARVEST4D and FP7-287723 REVERIE, by the ANR iSpace&Time project, by the China Scholarship Council, the national natural science foundation of China under grant No.61472224, 61472225, the national high-tech research and development plan of China under grant No.2012AA01A306, the special funding of independent innovation and transformation of achievements in Shandong Province under grant No.2014ZZCX08201 and the special funds of Taishan scholar construction project.

References


Figure 11: Corner scene (47 788 tri.). The close-up views illustrate the benefit of using AMBs.

Figure 12: Comparison to BPT for low (diffuse) to high frequency (specular) BRDFs (1 154 tri., $\gamma = 0.1$). The number of samples and the total rendering time are given below the top (BPT) images. The total rendering time and the $\text{MSE}$ error with the reference are given below the bottom (WPBGI) images. The roughness value corresponds to the glossiness index of a rough conductor BRDF.


<table>
<thead>
<tr>
<th>Scene</th>
<th>Max Path.</th>
<th>Num. Total Samples</th>
<th>Pts.</th>
<th>Total Memory</th>
<th>Pr. Time</th>
<th>Render Time</th>
<th>Total Time</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring</td>
<td>3</td>
<td>16384 5.49</td>
<td>5</td>
<td>128²</td>
<td>5.98</td>
<td>3.22</td>
<td>1.14</td>
<td>1.20</td>
</tr>
<tr>
<td>Corner</td>
<td>3</td>
<td>16384 6.63</td>
<td>8</td>
<td>32²</td>
<td>6.62</td>
<td>3.05</td>
<td>1.57</td>
<td>1.63</td>
</tr>
<tr>
<td>Bunny</td>
<td>3</td>
<td>16384 10.53</td>
<td>4</td>
<td>32²</td>
<td>5.68</td>
<td>12.28</td>
<td>0.84</td>
<td>1.04</td>
</tr>
<tr>
<td>Kitchen</td>
<td>3</td>
<td>16384 11.20</td>
<td>5</td>
<td>32²</td>
<td>5.26</td>
<td>6.57</td>
<td>3.66</td>
<td>3.77</td>
</tr>
<tr>
<td>Sphere</td>
<td>3</td>
<td>16384 10.33</td>
<td>10</td>
<td>32²</td>
<td>5.85</td>
<td>11.07</td>
<td>1.75</td>
<td>1.94</td>
</tr>
<tr>
<td>Torus</td>
<td>5</td>
<td>65536 40.01</td>
<td>2</td>
<td>32²</td>
<td>5.05</td>
<td>78.56</td>
<td>0.98</td>
<td>2.29</td>
</tr>
</tbody>
</table>

Table 2: Performance measures.


