UNIVERSITÉ CÔTE D'AZUR

Inria INVENTEURS DU MONDE NUMÉRIQUE

Doctoral School STIC

Research unit: GraphDeco, Inria Sophia-Antipolis

# PhD Thesis

Presented to obtain the title of

**PhD in Computer Science**

from

UNIVERSITY COTE D'AZUR

by

Rodrigo José Ortiz Cayón

# Improving Speed and Image Quality of Image-Based Rendering

Thesis Advisor: George Drettakis

Defended on February 3rd, 2017
Jury:

| | | | |
|---|---|---|---|
| Edmond Boyer | - | *Inria, Grenoble Rhône-Alpes* | - Reviewers |
| Fredo Durand | - | *Massachussets Institute of Technology* | - President |
| Martin Eisemann | - | *Technische Universität Braunschweig* | - Reviewers |
| Florent Lafarge | - | *Inria, Sophia-Antipolis* | - Examinators |
| Fabio Pellacini | - | *Sapienza Università di Roma* | - Examinators |

UNIVERSITÉ CÔTE D'AZUR

Inria INVENTEURS DU MONDE NUMÉRIQUE

École doctorale STIC

Unité de recherche : GraphDeco, Inria Sophia-Antipolis

# Thèse de doctorat

Présentée en vue de l'obtention du grade de

**Docteur en Sciences - Mention Informatique**

de

l'UNIVERSITÉ COTE D'AZUR

par

Rodrigo José Ortiz Cayón

# Amélioration de la Vitesse et de la Qualité d'image du Rendu Basé Image

Dirigée par George Drettakis

Soutenue le 3 Février 2017
Devant le jury composé de :

| | | | | |
|---|---|---|---|---|
| Edmond Boyer | - | *Inria, Grenoble Rhône-Alpes* | - | Rapporteur |
| Fredo Durand | - | *Massachussets Institute of Technology* | - | Président |
| Martin Eisemann | - | *Technische Universität Braunschweig* | - | Rapporteur |
| Florent Lafarge | - | *Inria, Sophia-Antipolis* | - | Examinateur |
| Fabio Pellacini | - | *Sapienza Università di Roma* | - | Examinateur |

# Contents

## Representative Publications

---

- **Ortiz-Cayon, Rodrigo**, Djelouah, Abdelaziz, & Drettakis, George. 2015 (Oct.). A Bayesian Approach for Selective Image-Based Rendering using Superpixels. *In: International Conference on 3D Vision (3DV). - Oral presentation*.

- **Ortiz-Cayon, Rodrigo**, Djelouah, Abdelaziz, Massa, Francisco, Aubry, Mathieu, & Drettakis, George. 2016. Automatic 3D Car Model Alignment for Mixed Image-Based Rendering. *In: 2016 International Conference on 3D Vision (3DV). - Oral presentation*.

---

## Acknowledgments

In the period of working for my doctoral studies And lastly producing this written document, Many people have been involved. For all of them I present my gratitude. However I would like to highlight three people without whom, this thesis would not be possible: My supervisor **George Drettakis** and my colleagues **Abdelaziz Djelouah** and **Gaurav Chaurasia**. All of them are not only science-wise role models but also kind people that worry about the well-being of everybody around them. My acknowledgment and deep admiration to them.

I do not want to miss the name of anyone. I am grateful to the members of the CR-PLAY consortium and the EU commission who financially funded my thesis. Grateful to my colleagues at the GraphDeco team, engineers of this team, friends at Inria Sophia-Antipolis, co-authors and reviewers.

Finally I want to dedicate the efforts of my doctoral work to my wife, friend, and favorite person all over the earth, **Amandine Guinard** who patiently waited for me at my side during these three years.

# Abstract

Traditional photo-realistic rendering requires intensive manual and computational effort to create scenes and render realistic images. Thus, creation of content for high quality digital imagery has been limited to experts and highly realistic rendering still requires significant computational time.

Image-Based Rendering (IBR) is an alternative which has the potential of making high-quality content creation and rendering applications accessible to casual users, since they can generate high quality photo-realistic imagery without the limitations mentioned above.

We identified three important shortcomings of current IBR methods: First, each algorithm has different strengths and weaknesses, depending on 3D reconstruction quality and scene content and often no single algorithm offers the best image quality everywhere in the image. Second, such algorithms present strong artifacts when rendering partially reconstructed objects or missing objects. Third, most methods still result in significant visual artifacts in image regions where reconstruction is poor.

To tackle the first problem, we propose a Bayesian IBR algorithm, which chooses between different rendering algorithms in each image region, by making the best quality/speed trade-off. This selection is performed in a pre-process, by evaluating rendering quality according to distance in geometry and color via a leave-one-out strategy on the input images. Our method results in significant speedup in rendering time, permitting the use of this approach on mobile devices.

For the second problem, we focus on the case of reflective objects which are hard to reconstruct, such as cars. The key insight is to replace these poorly reconstructed objects with models from existing rich 3D CAD databases, and subsequently align them to the input images. We adapted deep learning-based algorithms to a multi-view context to obtain the 3D model present in the databases which is closest to the object seen in the images. We formulate two optimizations using all available information to finely position and orient the model and adapt it to image contours. Our method provides much higher quality rendering results of such objects compared to previous solutions.

Finally we present initial ideas on two ways to measure and correct the error introduced by poorly reconstructed regions in IBR in wide-baseline multi-view scenes, either with hand-crafted descriptors or a deep learning approach. Initial results show promise for improving the quality of depth and thus IBR.

Overall, this thesis addresses significant shortcomings of IBR for both speed and image quality, offering novel and effective solutions based on selective rendering, learning-based model substitution and depth error prediction and correction.

**Keywords:** *Image-based rendering and modeling, Rendering, Bayes methods, Estimation, MVS reconstruction Shape-preserving Warp, Superpixels, Deep Learning.*

# Résumé

Le rendu photo-réaliste traditionnel exige un effort manuel et des calculs intensifs pour créer des scènes et rendre des images réalistes. C'est principalement pour cette raison que pourquoi la création de contenus pour l'imagerie numérique de haute qualité a été [est ] limitée aux experts et le rendu hautement réaliste nécessite encore des temps de calcul significatifs. Le rendu basé image (IBR) est une alternative qui a le potentiel de rendre les applications de création et de rendu de contenus de haute qualité accessibles aux utilisateurs occasionnels, puisqu'ils peuvent générer des images photo-réalistes de haute qualité sans subir les limitations mentionnées ci-dessus. Nous avons identifié trois limitations importantes des méthodes actuelles de rendu basé image: premièrement, chaque algorithme possède des forces et faiblesses différentes, en fonction de la qualité de la reconstruction 3D et du contenu de la scène, et un seul algorithme ne permet souvent pas d'obtenir la meilleure qualité de rendu partout dans l'image. Deuxièmement, ces algorithmes présentent de forts artefacts lors du rendu d'objets manquants ou partiellement reconstruits. Troisièmement, la plupart des méthodes souffrent encore d'artefacts visuels significatifs dans les régions de l'image où la reconstruction est de faible qualité.

Pour répondre au premier problème, nous proposons un algorithme IBR Bayésien, qui sélectionne pour chaque région de l'image, parmi différents algorithmes de rendu, celui permettant d'obtenir le meilleur rapport qualité / temps de calcul. Cette sélection est effectuée dans une phase de pré-traitement, en évaluant la qualité du rendu en fonction de la distance en termes de géométrie et de couleur, par l'intermédiaire d'une stratégie de validation croisée sur les images d'entrée. Notre méthode permet d'obtenir une accélération significative des temps de rendu, rendant possible l'utilisation de cette approche sur des appareils portables. Pour le deuxième problème, nous nous concentrons sur le cas des objets réfléchissants, qui sont difficiles à reconstruire, tels que les voitures. L'idée clé consiste à remplacer ces objets mal reconstruits par des modèles 3D, provenant des bibliothèques de modèles CAO de bonne qualité, et d'aligner ces modèles avec les images d'entrée. Nous avons adapté des algorithmes de "deep-learning" dans un contexte multi-vues pour obtenir le modèle 3D issu des bases de données qui soit le plus proche de l'objet présent dans les images. L'estimation de la pose (position et orientation) et l'adaptation du modèle 3d sont formulés comme 2 problèmes d'optimisation qui prennent en compte l'ensemble de l'information disponible: images, contours, 3d. Notre méthode fournit des résultats de qualité supérieure aux solutions existantes pour le rendu de tels objets. Enfin, nous présentons une première réflexion sur la manière de mesurer l'impact des erreurs de reconstruction 3d sur le rendu basé image dans un contexte où les points de vue d'acquisition sont éloignés. Nous avons exploré deux approches. La première utilisent des descripteurs Daisy et la seconde utilisant le deep learning. Dans l'ensemble, cette thèse propose plusieurs améliorations significatives du rendu basé image aussi bien en terme de vitesse de rendu que de qualité d'image. Ces nouvelles solutions sont basées sur le rendu sélectif, la substitution de modèle basé sur l'apprentissage, et la prédiction et la correction des erreurs de profondeur.

Chapter 1

# Introduction

## Contents



In computer graphics, *rendering* refers to the process of generating 2D images from a *model of the scene*, with applications in video games, film industry, scientific visualization among others. Traditionally, the model of the scene is composed by 3D geometry, material properties (including textures, normal maps, bump maps etc.), lights and animations for dynamic scenes. Rendering algorithms are then applied to produce the final image. Two well-known techniques are *Scan-line Rasterization* and *Ray Tracing*. Scan-line Rasterization geometrically projects the scene to an image plane while Ray Tracing algorithms intersect rays emanating from the camera with the geometry of the scene to find the visible point at each pixel. Global illumination algorithms often use ray-tracing to implement physically-based techniques and simulate the interaction between light and materials (Pharr & Humphreys, 2004). Nowadays, with exhaustive detailed modeling of the scene, ray tracing algorithms are very close to achieving one of the most important goal of computer graphics: realism (see Fig. 1.1).

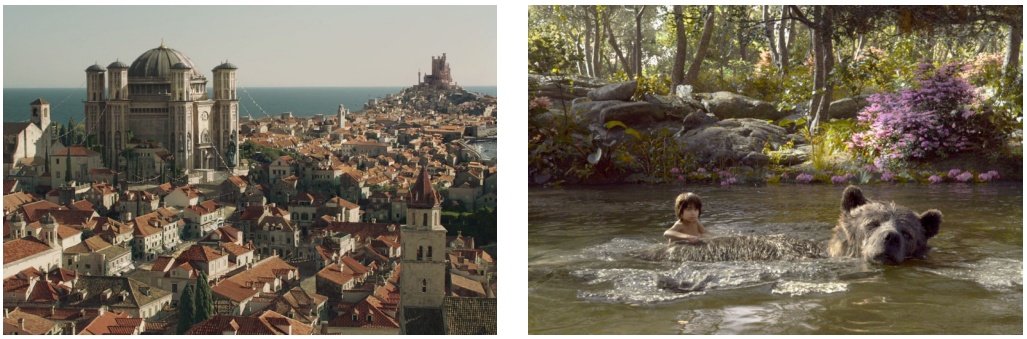


Figure 1.1: **Realistic rendering applied to film industry.** Right: Virtual middle age city created for the series *Games of Thrones*. Left: Scene from *The Jungle Book (2016)*. "*The computer graphics character Baloo is so large and furry, he took almost five hours of rendering time per frame*".

The process of modeling geometry, defining and assigning materials and lights is very complex. Depending of the size of the scene, it requires weeks of laborious work and

expertise to precisely define material properties, geometric details, etc. The degree of realism is heavily influenced by the number of primitives used in describing the scene. Some automatic methods have been developed to overcome manual efforts of creating scenes.

For geometry capture, examples include technologies for precise surface acquisition, like laser scanners, which however are still inaccessible to casual users and are too expensive for small business. Others, like Kinect-based solutions, present problems in open exterior spaces; the resulting meshes often need to be refined and completed by skillful artists. Another option is image-based reconstruction which might not offer enough accuracy and completeness for high-quality photo-realistic rendering, and currently does not easily allow material capture.

Material capture has also evolved in recent years, allowing acquisition of some properties from photographs. Most such solutions however involve controlled lighting, or laboratory setups and are not yet mature enough to substitute manual material creation for synthetic scenes. Overall, current techniques for geometry and material capture do not allow easy creation of synthetic digital assets and manual work of the artist is still central.

Once the scene is prepared, global illumination algorithms can render views from a desired camera position and direction. These algorithms require an enormous computational effort. Per single frame, it may take hours to simulate indirect light since a huge number of samples is required for accurate computation. Consequently, precise geometry plays a primary role in the simulation. For interactive graphics application, global illumination is still very expensive. Normally, such methods have been use off-line, for example, for film production.

In summary, correctly creating a scene for realistic rendering is a long process that requires expertise and long manual treatment. Despite many recent advances in rendering algorithms, expensive lighting simulation is still required to achieve realism. Content creation for traditional renderers face a trade-off between detailed complex models and time-to-deliver while rendering algorithms always provide a trade-off between rendering time and final image quality.

In this thesis, we focus on image-based rendering algorithms which avoid both the time-consuming content creation phase and expensive lighting simulation. Evidently, such approaches suffer from limitations, namely reduced image quality when moving away from input views, and large memory requirements. We provide new solutions to address these limitations.

## 1.1 Context and Motivation

In this context, content creation for rendering has evolved to more casual procedures, making graphics technologies available to wide number of users. For example, Computer Assisted Design (CAD) tools like *SketchUp*[1] allow easy creation of rough 3D graphics. With applications like *Autodesk 123D*[2] everyone can obtain 3D models of real objects

[1] http://www.sketchup.com
[2] http://www.123dapp.com

from a set of photographs taken with mobile phones. The *Tango project*[3] enables mobile devices with an on-board Time-of-Flight camera to provide point clouds of the scene. Large databases of 3D objects with semantic annotations are continuously increasing like *ShapeNet*[4] and *Objectnet3D*[5]. All these are examples of new sources of content that can be potentially used for high-quality rendering.

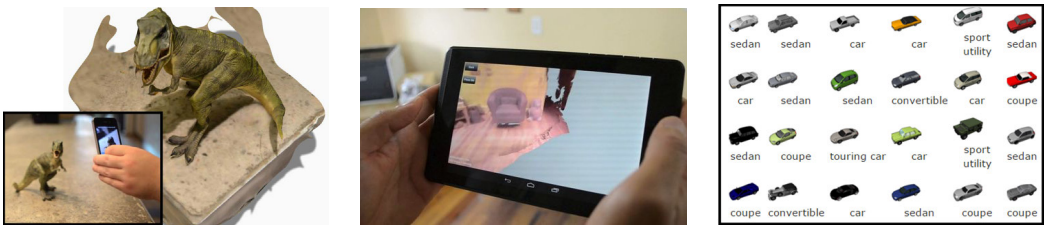



Figure 1.2: **Casual content creation.** Right: Acquisition with *Autodesk 123D*. Center: *Tango* device. Left: Some instances of 3D models available in *ShapeNet* database.

With this heterogeneous content, rendering algorithms have to adapt to integrate these kind of models into the rendering process and deal with imperfect geometric data. Image-based Rendering (IBR) presents an alternative in such scenarios where the difficulty of producing realistic models of real environments is greatly reduced by replacing the entire rendering process with image interpolation, assisted by 3D reconstruction from images. Instead of relying on 3D geometry, predefined material characteristics and a subsequent complex lighting simulation, IBR uses information contained in photographs to synthesize novel view-points of the scene. Another way to see the IBR paradigm is that "*Nature already simulates light and material interactions with the right model for free*". This is one of the informal principles of IBR, namely that 3D geometry plays a secondary role while 2D images are directly used at rendering time to look up information of possible novel view appearance.

How *we sample this information (rays, pixels, superpixels)*, where we take the information (which cameras, which regions?) and how we blend that information are central elements of the different IBR approaches. Another important component is how much geometry is involved in this process.

In Fig. 1.3 we present an abstraction of the general IBR process. By encoding the model in photographs, IBR tries to solve core problems of computer graphics: First, the need for simpler modeling techniques so the representation becomes independent from scene complexity. Second, the need to dissociate the rendering time from scene complexity.

A very popular – but simplistic – IBR application is Google *Street View*, where the user can move to discrete locations to see environment maps of streets. *Microsoft* has also developed successful IBR commercial applications with *Photosynth*[6] and *Hyperlapse*[7]. In

[3] https://get.google.com/tango
[4] https://shapenet.cs.stanford.edu
[5] http://cvgl.stanford.edu/projects/objectnet3d
[6] https://photosynth.net
[7] http://research.microsoft.com/en-us/um/redmond/projects/hyperlapseapps

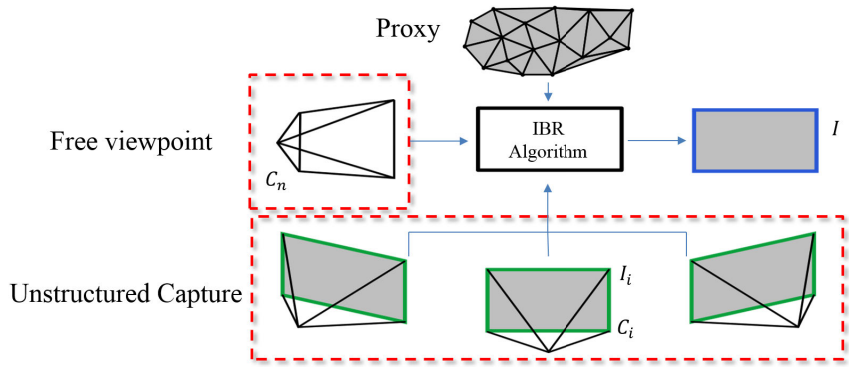



Figure 1.3: **Abstraction of an Image-based Rendering algorithm.** *Input*: A set of cameras $C_i$ and input images $I_i$ (preferably with unstructured capture); the goal of IBR is to generate an image $I$ "as seen" from the novel camera $C_n$. Approximate geometry of the scene (Proxy) can be used in the process. *Output*: Synthesized view $I$.

the former, the user can interpolate between the photographs. In the latter, an algorithm optimizes a smooth camera trajectory and synthesizes a smooth video sequence from very shaky time lapses. All applications mentioned above restrict the rendering path to the positions of input views, while IBR methods that allow free-view point navigation make assumptions about the scene geometry. For instance, *Street View* assumes spherical projections of the world, while in *Photosynth* the user must use "a type of shooting" (spin, panorama, wall, walk) and thus making assumptions about scene geometry.

Recent IBR algorithms try to compensate for geometric errors by adapting the scene geometry in preprocessing or regularizing the synthesized view at render time. The key concept is *visual plausibility*: a generated view does not need to be physically correct but plausible to the human eye (Zitnick & Kang, 2007). Methods that correct misalignments can be based on optical flow (Eisemann *et al.*, 2008), interpolate in world space from 2D correspondences (Lipski *et al.*, 2014), or use piece-wise regularization to preserve local shape (Chaurasia *et al.*, 2013). All of them process an initial approximate and incomplete 3D reconstruction; the creation of this "proxy" is considered a preprocessing step of the IBR algorithm.

In recent years, concurrently with this thesis, we have seen an accelerated progress of image synthesis using machine learning techniques. Rematas *et al.* (2014) and Zhou *et al.* (2016) have attempted to predict the appearance of novel views. Although their contributions have applications for inpainting and other vision tasks, their results are not good enough for high quality graphics as we can be seen in Fig. 1.4. Nonetheless, deep learning algorithms have reached a high level of robustness and accuracy for detection tasks. Thus, they can be used as a powerful tool to interpret image content and can be used in conjunction with large sources of available 2D and 3D data collections in order to enhance rendering.

Image-based rendering is related to, but quite different from Image-based modeling (IBM) (Xiao *et al.*, 2008, 2009; Micusik & Kosecka, 2009). The latter refers to the use

Figure 1.4: **View synthesis with learning methods.** Top row: View synthesis of Zhou *et al.* (2016). Input view (left) and Synthesized view (right). Bottom row: View synthesis of Rematas *et al.* (2014). Input view (left) and Synthesized view (right).

of images to drive the creation of explicit 3D models with textures mapped on this 3D model, consistently with the reference views. For IBM the rendering algorithm is a separate and independent process. As mentioned previously, IBM assets usually require manual post-processing (e.g., addition of materials) before being used in a traditional rendering engine. *The Vanishing of Ethan Carter*[8] is an example of a video game where most assets where acquired with IBM. Although IBR and IBM are orthogonal subjects, the natural way to acquire the scene geometry is through IBM, particularly, we are interested in acquisition with consumer level hand-held cameras due the simplicity and availability which might extend the range of users who use IBM.

## 1.2 Problem Statement

In spite of the big advances in free view-point IBR during the current decade there is still room for improvement. In a standard pipeline for IBR without user intervention, the process starts with camera calibration, multi-view stereo and mesh reconstruction. In reality, each of theses steps can be incomplete and includes inaccuracies and errors. Even under perfect reconstruction information, one would ideally need an infinite number of photographs to freely render every view-point of the scene. In this thesis, we are interested in sparse casual capture, for example 10 to 30 photographs for a walk-through of 10-30 meters. Reconstructions and IBR methods have to make assumptions about the world to fill the lack of information. The relationships of all mentioned process with the final rendered image are very complex with many interdependent components.

In this thesis, we are interesting in urban scenes that represent a special challenge and use case because:

[8] http://ethancartergame.com

- Such scenes contain complex geometry (e.g. vegetation, specularities, transparencies, etc.) with complex appearance of different objects.
- Active cameras do not work in exteriors and so, multi-view stereo is required.
- Urban scenes need scalability. Scene size is large (compared for example to single-object capture), the number of cameras must be limited and the result should not require manual intervention.

Under these conditions, the goal of this thesis is to propose new approaches to improve high-quality image-based rendering of urban scenes acquired with hand-held cameras and free-viewpoint navigation. The unique restriction we have to capture is a static scene with fixed illumination. We address three main problems:

- Given a set of IBR algorithms, each of them has different quality/speed trade-offs and no single algorithm is better than all others. We investigate how to combine them so the final image has the best quality-speed trade-off.
- We investigate how to improve rendering of poorly reconstructed objects for IBR scenes given databases of 3D CAD models.
- We provide some initial results on identifying and correcting poorly reconstructed regions in IBR scenes.

## 1.3 Contributions and Overview

The rest of this thesis is structured as follows. In Chapter 2 we review the history of image-based rendering methods and group them in a taxonomic classification. In chapter Chapter 3 we present a principled way to choose amongst a set of IBR algorithms for each image region. This allows to maximize rendering quality while keeping or increasing the speed. Our real-time implementation has low computational requirements which allowed high quality Image-based Rendering on mobile platforms. In chapter Chapter 4 we present the first approach able to automatically augment IBR using objects from 3D databases. In per-processing, a 3D CAD model is selected from a database. The model is aligned to image content and morphed to finely adapt it to image silhouettes. We applied our methods to cars which are often poorly reconstructed, and the resulting IBR quality is greatly improved. In chapter Chapter 5 we attempt to identify unreliable reconstruction information that affects rendering. We discuss two different approaches. First we use an image descriptor that detects regions with extra or missing geometry. In the second approach, we train a deep network with synthetic data to identify imprecise reconstruction. Chapter 6 concludes the thesis, and presents some possible directions for future work.

**Thesis context** This thesis was in the context of the EU project CR-PLAY[9], which focused on providing a usable IBR system for game development. As we will describe

[9] www.cr-play.eu

in Chapter 3, our work was actually used by game developers as part of an evaluation workshop, and provided concrete evidence of the potential of IBR in a real-world application.

CHAPTER 2

# Previous Work

## Contents



It has been more than two decades since Image-Based Rendering has emerged as an alternative to polygon-based rendering. Since then, although IBR has been an active field of research, few surveys have been presented (Shum & Kang, 2000; Zhang & Chen, 2004; Shum *et al.*, 2008; Lipski *et al.*, 2015). Usually, IBR taxonomies are presented according to the accuracy and amount of geometry required by the different algorithms. The spectrum of methods varies from algorithms that do not require any geometry, IBR methods that use image correspondences (implicit geometry) to methods that require detailed explicit geometry (see Fig. 2.1). This continuum reveals that the number of reference views needed is inversely proportional to the available geometry.

Another way to classify IBR input data requirements is by considering the capture setup. IBR capture setups have been heavily influenced by the evolution of Computer Vision (3D reconstruction and image analysis). We could divide IBR methods circa the

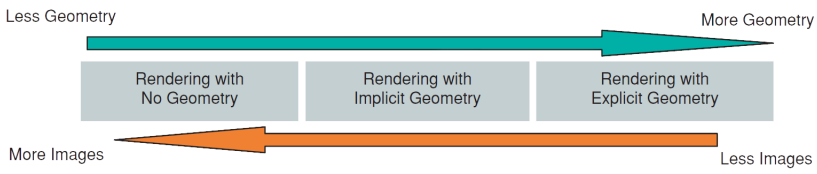
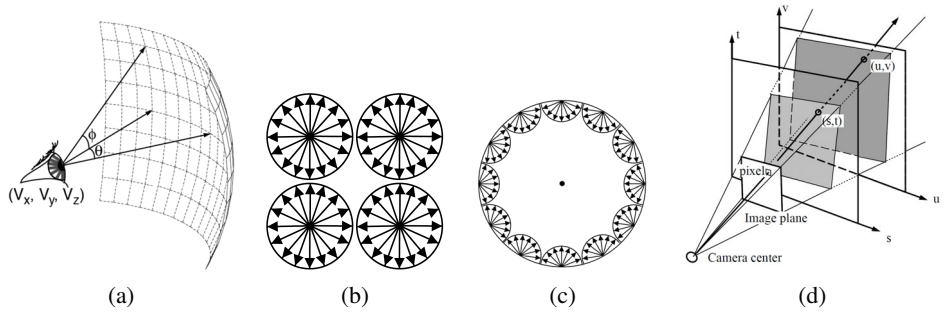



Figure 2.1: **Taxonomy proposed by Shum & Kang**. From left to right: IBR algorithms that do not use any geometry require a high number of images while algorithms that use detailed geometry need a small number of images.

year 2000. Initial methods (see Section 2.1) either used synthetic images or involved a specific setup, often with user intervention. Low complex scenes or very restricted navigation were also common. In Section 2.2 we mention some reconstruction algorithms that influenced an evolution to unstructured casual captures with more complex scenarios and greater freedom in viewpoints permitted for navigation. Then, in Section 2.3 we present modern algorithms more related to our context.

## 2.1 Initial IBR Methods

In a systematic way, Adelson & Bergen (1991) tried to derive the early visual cues[1] that are involved in the human vision process. Their work influenced researchers in the Computer Vision and Graphics communities during the 90's (Levoy & Hanrahan, 1996; Gortler *et al.*, 1996; McMillan & Bishop, 1995; Shum & He, 1999; Chai *et al.*, 2000). Adelson and Bergen showed that all the basic visual measurements can be characterized as local changes of a single function called the *plenoptic function*[2]. The plenoptic function $P(V_x, V_y, V_z, \theta, \phi, \lambda, t)$ has 7 dimensions and describes the radiance perceived from all positions $(V_x, V_y, V_z)$ and directions $(\theta, \phi)$ for every possible wavelength $\lambda$ at the time $t$. If we only consider static scenes (thus dropping $t$) with fixed light conditions (dropping $\lambda$), it takes the form: $P(V_x, V_y, V_z, \theta, \phi)$ see Fig. 2.2a.




(a) (b) (c) (d)

Figure 2.2: **(a)** Plenoptic eye: For static scenes with fixed light conditions, what the eye sees at position $(V_x, V_y, V_z)$ and direction $(\theta, \phi)$ can be considered a sample of the plenoptic function (McMillan & Bishop, 1995). **(b)** Top-view of a panoramic capture. **(c)** Top-view of Lumigraph style of capture. **(d)** Relationship between Lumigraph and an arbitrary pixel (Gortler *et al.*, 1996).

Initial IBR approaches tried to explicitly sample the plenoptic function (McMillan & Bishop, 1995; Levoy & Hanrahan, 1996; Gortler *et al.*, 1996). Previously, Chen & Williams (1993) and Laveau & Faugeras (1994) had demonstrated that sequences of synthetic images can be used to represent a scene as in Fig. 2.3. In between novel views could be created by interpolating the color and position of pixels for which dense optical flow was known beforehand. McMillan & Bishop (1995) were the first to use real scene photographs. They

[1] Visual cues -motion, color, orientation, binocular disparity, etc.- before a higher level visual process that involves the memory.

[2] From the Latin *plenus*, meaning complete or full, and the Greek *opticus* meaning vision


Figure 2.3: **Interpolation from synthetic views in Chen & Williams.** Interpolated view (middle) from the two synthetic source images (in the corners).

captured cylindrical panoramic (see top view in Fig. 2.2b). They could render cylindrical projection at discrete locations by stitching photographs, similarly to QuickTime VR (Chen, 1995). These systems are only capable of describe image variations due to view rotations. View translations can only be approximated by "jumping" discretely from one environment map to another (location of the capture) comparable to what is nowadays Google Street View.

Levoy & Hanrahan (1996) and Gortler *et al.* (1996) simultaneously presented a 4D parametrization of the plenoptic function called *Light Field* or *Lumigraph* respectively. They encoded the scene's light rays by their intersection with two parallel planes $st - plane$ (near to cameras locations) and $UV - plane$ (behind the object of interest). Levoy & Hanrahan acquisition was done by densely sampling a regular grid on the *st* plane. Novel views could be render by querying rays and interpolating between them. Some commercial implementation of *Light Fields* cameras (*Lytro camera*[3]) allow to focus in different points of the photographs after the photo has been taken. While Levoy & Hanrahan used an electronic setup to regularly acquire photographs, Gortler *et al.* allowed a semi-unstructured capture around the object (see Fig. 2.2c) and marker-based calibration cameras to map camera positions toward the *st* plane. They also allowed the use of approximate geometry to restrict and correct the queried rays (to alleviate self-occlusions and quantization error). Thanks to this parametrization and capture setup, Light fields and Lumigraph systems can navigate around objects contained in between the two planes (Fig. 2.2d). For Lambertian surfaces this representation has a high degree of redundancy because they store many rays that intersect the object surface at the same point and therefore, represent the same color.

Another group of methods use explicit geometry in the form of polygonal meshes (Debevec *et al.*, 1996) or dense depth (Mark *et al.*, 1997), (Shade *et al.*, 1998). Debevec *et al.* recover a scene "proxy", i.e., a simple geometric representation, from geometric primitives drawn by users. The accuracy of the recovered model is improved by computing re-projection offsets. This rendering algorithm texture mapped the recovered geometry with blending weights depending on the novel camera position. That was called *View-Dependent Texture Mapping* (VDTM) (Debevec *et al.*, 1998) and it is still used in

[3] https://www.lytro.com


(a) (b)

Figure 2.4: **(a)** Reconstruction method of Debevec *et al.*: Left: A photograph with user marked edges. Center: The model recovered by their photogra-metric modeling method. Right: A synthetic view generated using VDTM. **(b)** 2D Warp of Mark *et al.*: In black, areas of the scene that were occluded are now exposed due to warping from a reference view. Multiple reference frames can be composited to fill the gaps.

modern IBR algorithms to select and synthesize color keeping view-depending effects (see Fig. 2.4). To increase frame rates in remote display graphic applications, Mark *et al.* (1997) presented a forward warping IBR system. Instead of sending polygons to the client every frame, Mark et al proposed to send rendered reference views. Using z-values at each pixel a dense triangle mesh is constructed for the two reference views and with Chen & Williams forward warp them to the novel in-between position. Normal vectors and z-values at each pixel are used to locate false connections across edges of occluding and occluded surfaces. The new view is composed filling dis-occlusion holes. Another representation to fill dis-occlusion caused in forward mapping methods was presented by Shade *et al.* (1998). Shade et al called this representation *Layered Depth Images* (LDI). A LDI is a view of a scene from a single input camera but with multiple depth pixels along each line of sight. Depth pixels are revealed to fill holes along with view position changes.

The afore-mentioned IBR methods involved complicated capture (or use of synthetic data) and tedious user intervention. At the same time, they are restrictive in the navigation. Before going into more modern IBR systems, in Section 2.2 we give a brief overview of some reconstruction algorithms that allowed the evolution of IBR system.

## 2.2 Implicit and Explicit Reconstruction for IBR

In general, image-based rendering approaches that use geometry produce better rendering results. As mentioned before, geometric information could be presented implicitly in the form of correspondences between pairs of images (in Section 2.2.2) or explicitly with 3D reconstructions (in Section 2.2.1). In both cases, the natural way to obtain the scene geometry for IBR is through conventional photographs themselves without any further user intervention or specific device. At early stages, for each image, implicit and explicit reconstruction approaches detect interest points, match them and filter outliers. SIFT (Lowe, 2004) and SURF (Bay *et al.*, 2006) are well established local descriptors for interest points due to their invariance to image transformations. Inspired from SIFT, Tola *et al.* (2010) designed a fast descriptor for dense matching that reports remarkable robustness for

wide baseline camera setups. With a set of matched features, what is done after depends on the type of desired geometry. Correspondence methods use matched features to seed and propagate dense pixel-wise matches. 3D reconstruction methods use matched features to firstly calibrate input cameras and generate a sparse set of depth samples, then iteratively expand and filter 3D samples (*stereopsis*) and finally generate a mesh that approximates the scene geometry (*proxy*).

### 2.2.1 3D Reconstruction (Explicit geometry)

To be consistent with IBR, image-based reconstruction systems leverage only the information available in photographs. They start with a set of calibrated input views. Parameters of calibrated cameras are usually estimated with *Structure from Motion* (SfM). Structure from motion Tomasi & Kanade (1992) provides 3D point correspondences given 2D point matches in two or more images, and provides the camera matrices for these views.

The output of a SfM system is a sparse set of 3D points (structure) and camera parameters (motion). It was not until Snavely *et al.* (2006) presented a system called *Photo tourism* that SfM reached maturity and robustness to be used effectively with outdoor urban scenarios. Photo tourism allows navigation in a large collection of community photographs, with transitions from one view to another depending on their relative position. Snavely *et al.* (2006) iteratively refined the SfM optimization by minimizing re-projection error (*Bundle adjustment*[4]). Parallel implementations of SIFT feature detection (Wu, 2007) and Bundle adjustment (Wu *et al.*, 2011) are available in the *VisualSFM*[5] tool. Other implementations of recent SfM methods that report more precision and robustness by automatically adapting SfM parameters (Moulon *et al.*, 2012) are available in the library *OpenMVG*[6].

Given a set of calibrated cameras, recovering the static 3D information of a scene is one of the most active areas in Computer Vision. For complete and well known reviews of two-view and multi-view stereo techniques, see Scharstein & Szeliski (2002) and Seitz *et al.* (2006) respectively.

**Multi-View Stereo** The *baseline* of a stereo system is the relative distance between the cameras's optical centers with respect to the depth of the scene. In Fig. 2.5 we can see two different baseline configurations. Okutomi & Kanade (1993) showed that using wider baselines produces several local minima on the matching score along epipolar lines. They proposed a window-based search along the corresponding epipolar line and as a search parameter (window size), they used inverse depth relative to the reference image. To resolve ambiguities, Okutomi & Kanade searched along the epipolar line of more than one neighboring view. Since then, stereo systems use multiple views (*Multi-View*) (Furukawa & Ponce, 2010; Goesele *et al.*, 2007; Jancosek & Pajdla, 2011) to improve matching robustness. As a drawback, this strategy implies to choose a reference view (which could be virtual) and to deal with occlusions.

[4] refers to the bundle of light rays leaving each 3D feature and converging on each camera center.
[5] http://ccwu.me/vsfm/
[6] http://imagine.enpc.fr/~moulonp/openMVG/

Beside window-based stereo reconstruction, contour-based or voxel-based methods have been explored. An example of these alternatives is the *Space Carving* algorithm (Kutulakos & Seitz, 2000) that relies on silhouette extraction and initialization of a bounding box around an object of interest. Also, Space Carving needs to close the loop of capture; it is thus not suitable for open spaces as in street views reconstructions and requires more views than patch-based algorithms.

For IBR, rather than accuracy of reconstruction, desirable properties are robustness and scalability. Goesele *et al.* (2007) were able to extract dense depth maps from community photograph collections with high variability in the photo content. A globally consistent point cloud was obtained by merging depth maps. Another popular reconstruction was presented by Furukawa & Ponce (2007, 2010). From the center of calibrated cameras and starting with a sparse set of initial matches, they iteratively expand matches to nearby locations toward the exterior of the images while using visibility constraints to filter out false matches. If we would like to represent surfaces as triangular meshes we could use the point or patch clouds resulted from Goesele *et al.* or Furukawa & Ponce to feed a surface reconstruction algorithm as *Floating Scale* (Fuhrmann & Goesele, 2014) or traditional *Poisson* reconstruction (Kazhdan *et al.*, 2006).

Although Goesele *et al.*, Furukawa & Ponce and other approaches proposed different algorithms, the final 3D reconstructions are fairly similar (see Fig. 2.6b and Fig. 2.6c) because they all assume diffuse materials (*Lambertian model*) and thus they measure photo-consistency validity. "Holes" usually appear in specular surface such as the windows and car in Fig. 2.6a and in busy texture such as the tree and bushes in Fig. 2.6a.

The power of IBR is precisely to be able to render view dependent effects and complex surfaces without having a detailed 3D model. By definition these surfaces are not photo-consistent. Even though they are poorly supported in the point cloud, they represent real parts of the scene. Jancosek & Pajdla (2011) can roughly approximate weakly supported surfaces by leveraging visual-hulls to reconstruct them. Improvement in reconstruction of these non photo-consistent surfaces can be seen in Fig. 2.6d.

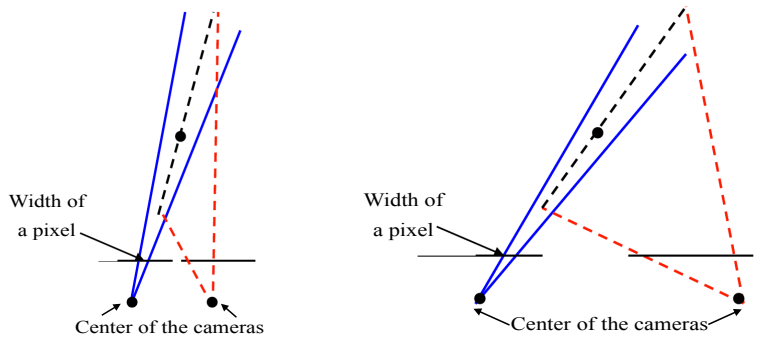



Figure 2.5: **Two baselines configurations.** On the right we have a larger baseline than on the left. The search space along the epipolar line of the is bigger (red dashed) in the right-hand configuration.

(a) (b) (c) (d)

Figure 2.6: **Different reconstructions for a 10 view dataset.** (a) One out of 10 input images. (b) Point cloud of PMVS reconstruction Furukawa & Ponce (2007). (c) Mesh of MVE reconstruction Goesele *et al.* (2007). (d) Mesh of CMPMVS reconstruction Jancosek & Pajdla (2011).

**Piece-wise Planar** Man-made scenes present strong regularities that can be approximated with piece-wise geometry. Also, planarity assumptions overcome the challenge when reconstructing texture-less surfaces. With this in mind, Zitnick & Kang (2007) and Sinha *et al.* (2009) presented two approaches designed specifically for image-based rendering. Instead of trying to extract physically correct depth, Zitnick & Kang observed that one can synthesize plausible views from view-dependent piece-wise planar depth maps. They also observed that preserving contours was crucial for plausible view synthesis and thus they presented a color based over-segmentation algorithm to precisely delineate object boundaries. Dividing input images in a regular grid (seeds), they grouped neighboring pixels color using a simple K-means clustering. Although this *superpixel* over-segmentation was originally presented by Zitnick & Kang, the version of Achanta *et al.* (2010, 2012), named *SLIC - Simple Linear Iterative Clustering*, became popular due the availability and efficient implementation limiting the range of search of K-means. Depth estimation at the superpixel level is more robust than at the pixel level. For each segment, Zitnick & Kang estimate depth solving a *MRF* on superpixel segments. For rendering, they assumed fronto-parallel segments with respect to the input camera where they belong.

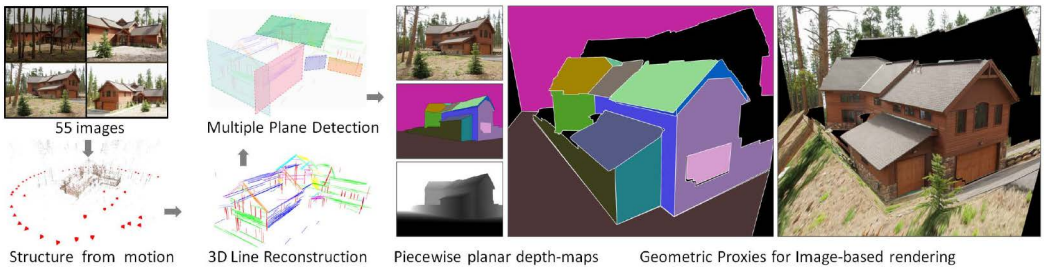



Figure 2.7: **Overview of the system of Sinha *et al.*:** Multiple planes are robustly extracted from sparse 3D points and lines; piecewise planar depth maps are then generated by graph-cut based energy minimization.

Differently, Sinha *et al.* (2009) extracted dominant planes of architectural scenes (see Fig. 2.7). Instead of constraining planes to orthogonal scene configurations with the *Manhattan world* assumption (Furukawa *et al.*, 2009), Sinha *et al.* detect vanishing directions from point clouds. Vanishing directions serve to reconstruct 3D lines that will support the 3D planes. Finally they solve a graph-cut energy to assign pixels to planes with photo-consistency, depth and visibility cues. For rendering they used VDPM (Debevec *et al.*, 1998). These reconstructions can appear incorrect for non planar objects such as vegetation and clutter.

To overcome this problem, Gallup *et al.* (2010) generate plane hypothesis from depth maps with traditional top-down RANSAC. They fuse overlapping maps across the views to obtain a globally consistent set of plane hypothesis. To assign pixels to planes, Gallup *et al.* minimizes a graph-cut labeling energy that includes a *non-planar* label. The non-planar label was found with supervised learning, trained with hand-labeled non-planar regions using traditional Computer Vision features.

### 2.2.2 Correspondences (Implicit geometry)

Per-pixel correspondences between photos is still an unsolved problem in natural scenarios. Dense *optical flow* works only for very narrow baselines. Wider baselines require awareness of preserving spatial discontinuities and tolerance to appearance changes. Analogous to dense optical flow, *SIFT flow* (Liu *et al.*, 2011) matches densely sampled, pixel-wise SIFT features between two images. Many false matches makes this method not good enough for rendering purposes by itself, but it has been used in hybrid algorithm (Lipski *et al.*, 2014).

The group of *M. Magnor* has developed several methods for view synthesis by interpolation, using correspondences based on geometry. Motivated by the way humans perceive motion, Stich *et al.* (2008, 2011) presented a piece-wise homography warp. The most important properties to create convincing smooth transitions were: keep exact edge correspondences and homogeneous region correspondences. That is because the motion of edges can be easily detected by the human eye while in homogeneous regions this is more difficult. They could interpolate space and time directly in image space without the need of

synchronized or calibrated cameras, however, to apply this method directly to real images with no user intervention would be challenging. Hence, their laboratory[7] has produced some tools for interactive correction of correspondences (Klose *et al.*, 2011) and tools to introduce depth information into dense correspondences (Ruhl *et al.*, 2012). Their tools were applied to hybrid implicit-explicit IBR renderings (Lipski *et al.*, 2014).

**Discussion** Explicit geometry allows IBR algorithms to have a more generic camera configuration and navigation with wider baselines than algorithms that use implicit geometry. On the other hand, implicit representations are better suited for dynamic scenes[8] because the number of primitives (pixels) remains constant regardless of changes in the structure. Acquisition in both representation schemes might fail and rendering algorithms should be robust to such representation inaccuracies.

Pure image-based algorithms for camera selection and color blending could integrate geometric proxies from other sources such as a scanner (Pulli *et al.*, 1997). In principle, reconstruction methods should be orthogonal to IBR approaches. However, quality and performance of modern IBR methods are tightly related to the way geometry is represented (see Section 2.3).

## 2.3 Unstructured-capture IBR

Calibrated cameras combined with approximate scene geometry provide powerful information for novel view synthesis in unstructured capture situations. Often, unstructured capture also means less restrictive navigation. With that in mind, Heigl *et al.* (1999) and further improvements (Koch *et al.*, 2001; Pollefeys *et al.*, 2004) used hand held video sequences (dense sampling) to capture IBR scenes. After calibrating cameras and performing stereo reconstruction, they project input camera's centers into the virtual image. They triangulate these projections and each triangle is texture mapped with the cameras from which the triangle vertices originated. The final color is drawn as a weighted sum of view-dependent textured triangles (see Fig. 2.8a). Due to their camera selection procedure, they could render only novel view "behind" the input ones. The pipeline presented by Heigl *et al.* globally represents what casual-capture for free view-point navigation IBR systems employ even today: scene batch preparation (SfM camera calibration and stereo reconstruction) and a view synthesis algorithm (view selection, color mapping and blending).

We can classify algorithms according to the way they map color information to synthesize rays, pixels or entire regions. We can distinguish two kinds of IBR methods. The first use re-projection (backward map) to look up color, (e.g. Debevec *et al.* 1998; Buehler *et al.* 2001). The second set of methods use forward color mapping from inputs to target views in image space (e.g. Zitnick *et al.* 2004; Chaurasia *et al.* 2013). Both types of

[7] http://graphics.tu-bs.de/

[8] IBR for dynamic scenes is a.k.a. *Video-Based Rendering* (VBR). In this thesis, we do not treat dynamic scenes.


(a) (b)

Figure 2.8: **Camera selection and rendering of Koch *et al.* (2001).** (a) Select the three spatially close cameras, triangulate the geometry and progressively sub-divided triangles for refinement. (b) Details of rendered images showing the result with initial sub-division (left) and with geometric refinement (right).

methods count on geometric information to drive their mappings and will be discussed in Section 2.3.1 and Section 2.3.2 respectively.

### 2.3.1 Re-projection Methods

To build a novel view, re-projection approaches ask the question: starting from this pixel/ray of the novel camera, from where can we take color information to "draw" the pixel/ray? Or what are the nearest rays/pixels that give the best support to synthesize the novel color? The notion of proximity was determined uniquely by the camera configuration in initial IBR methods (in Section 2.1 except for Debevec *et al.* 1996). The final rendered image would strongly depend on the structure of the capture. It should depend on both camera configuration and the available geometry.

With the proxy and input cameras, *minimal angular deviation*[9] gives an effective measure of closeness to render rays with view-dependent effects. Buehler *et al.* (2001) reviewed this and other desired properties that all IBR methods should fulfill resulting in the *Unstructured-Lumigraph Rendering* (ULR) algorithm, that has greatly influenced this domain. *Equivalent ray consistency*[10] and *sensibility to image resolution* are some of the desired properties. To enforce them, Buehler *et al.* designed an heuristic without trying to further explain the physical phenomena involved. From each ray of the novel view and guided by the scene proxy, they look up color in input cameras. With minimal angle and resolution measures, their heuristic assigns a weight to each back-projected source of color (camera).

To ensure *continuity*, Buehler *et al.* stored camera weights in a continuous *Blending*

[9] Source image rays with similar angles to the desired ray should be used

[10] A ray along the line of sight of an empty regions should be reconstructed consistently regardless of the virtual view position.

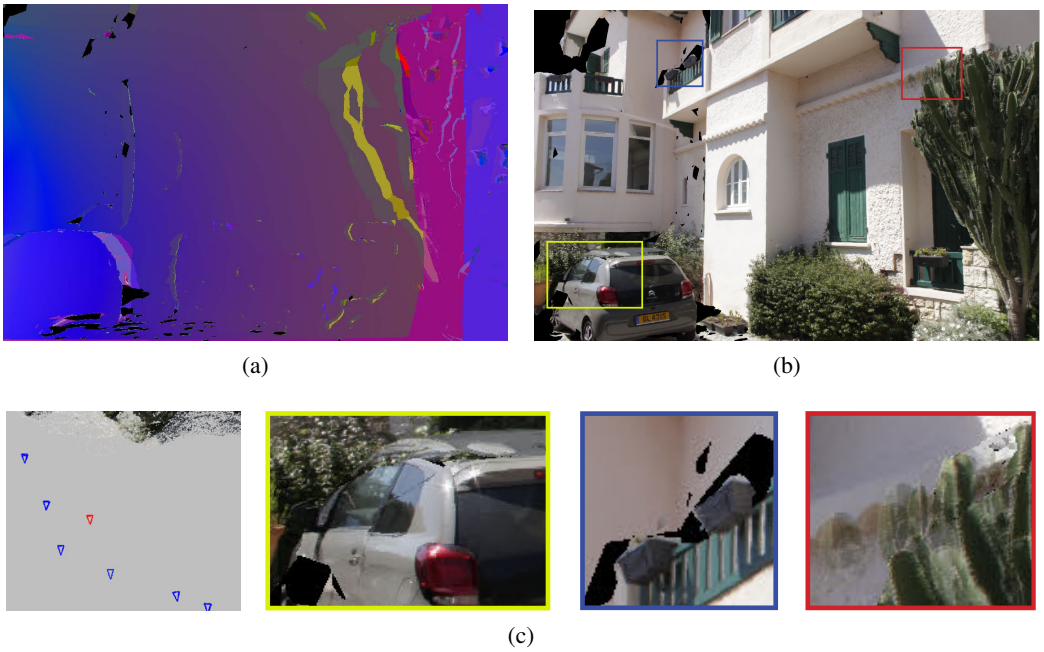


Figure 2.9: **Blending Field and rendering artifacts of *ULR*.** (a) Continuous blending field. For this visualization, we assign one color to each input camera. A "pure" color (e.g. blue) means that uniquely one camera contributes to the novel view. We observe continuous transition of color (contribution of cameras) except on the right side where some regions where exclusively seen from some cameras and in black regions where no depth was available. (b) View rendered form a novel position. (c) At left, the top view of the scene with the red triangle representing the novel view position and the blue triangles the input camera positions. Some regions of (b) are highlighted in (c).

*Field* data structure (see Fig. 2.9a). This parallel structure allows real-time and continuous blending of color. Since image based reconstruction can only approximately describe the scene, errors in camera calibration and geometry produce misalignment of re-projections. This causes the selected textures to be blended a different appearance; producing in ghosting artifacts (see Fig. 2.9b).

Eisemann *et al.* (2008) presented a method to automatically re-align the different projections using *optical flow*. Optical flow works effectively if re-projected textures coming from different input images finely approximate each other (see Fig. 2.10a). Other kind of error happens due to approximation in depth discontinuities (occlusion errors). Instead of using hard geometric-based visibility maps, Eisemann *et al.* tackle this problem by introducing "soft" visibility: they weight each pixel of the visibility maps according to the distance to the next occlusion pixel.

In some regions where the geometry is incomplete (see Fig. 2.9b), the reconstruction process cannot commit to a single depth because of ambiguity and uncertainty in the matching. Goesele *et al.* (2010) randomly sample the epipolar line of pixels with no depth to distribute multiple depth samples along segments of the viewing ray in the direction

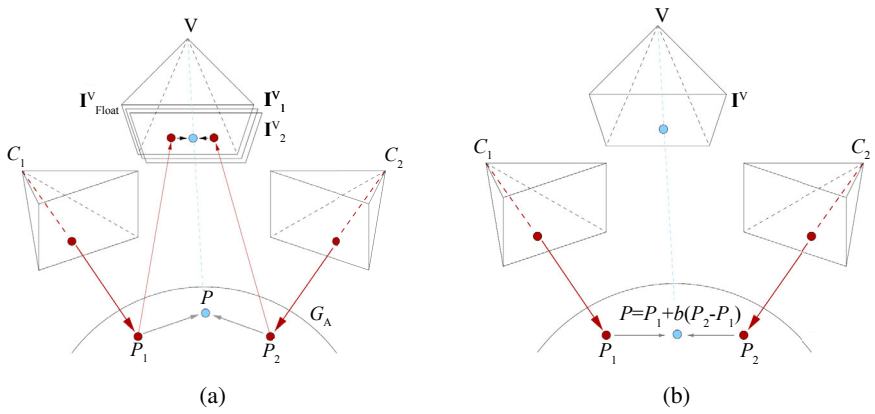



Figure 2.10: (a) Rendering with Floating Textures (Eisemann *et al.*, 2008): Input views $C_i$, are re-projected onto the novel view $V$ with the geometry $G_A$ and The resulting intermediate images $I_i^V$ are aligned with optical flow to obtain the final image $I_{Float}^V$. (b) Rendering of Lipski *et al.* (2014): image correspondences might not be at the same location in 3D (points $P_1$ and $P_2$). The location of the new point is interpolated before projection onto novel view $V$.

of uncertainty. They called this representation *Ambient Point Cloud*. Combined with the mesh, ambient point clouds create a of more appealing transition, although they produce a non photo-realistic blur effect in image transitions.

In re-projection IBR algorithms, the selection of content from input photographs is largely guided by a globally consistent representation. The global representation allows selection of content from all input views for each ray of the novel view. In this thesis, we do not commit to global geometric representations which can be inaccurate and incomplete. Instead, we prefer approaches with view-dependent representations. These approaches usually transform views from input views toward the novel one as we will see in Section 2.3.2.

### 2.3.2 Forward-mapping Methods

Different from re-projection approaches, forward mapping methods start from input cameras and transform these inputs toward the novel view position. Inspired by *Layered Depth Images* (Shade *et al.*, 1998), Zitnick *et al.* (2004) use a two-layer representation, one layer for general image content and other layer for image contours. The layer of contours is a boundary strip around depth discontinuities, containing information of background/foreground colors, alpha values and depths. From a set of synchronized video cameras, they first over-segment input frames and then compute dense stereo. Zitnick *et al.* use this to detect depth discontinuities and compute the layer of contours. Rendering is accomplished by 3D warping all layers from two reference views toward the intermediate view and blending the layers. They formally presented their reconstruction steps (superpixel segmentation and superpixel stereo) for IBR in Zitnick & Kang 2007. With an independent layer for alpha values of contours they avoid blending artifacts in

Figure 2.11: **Novel view rendered with 4 previous methods.** Top-left: *Floating textures* of Eisemann *et al.* (2008). Optical flow fails in specular regions like the front of the car. Top-right: *Ambient point clouds* of Goesele *et al.* (2010). Regions with no depth (upper corners of the image) are sampled with multiple depth pixels. Bottom-left: *Silhouette-aware warping* of Chaurasia *et al.* (2011). Regions around the car present distortion. Bottom-right: *Shape-preserving warping* of Chaurasia *et al.* (2013). A complete and plausible image is generated, however, the car and the facade present artifacts.

the contour of warped regions, however, they completely rely on accurate dense depth estimation which would make difficult the application of their system to complex outdoor environments quite challenging.

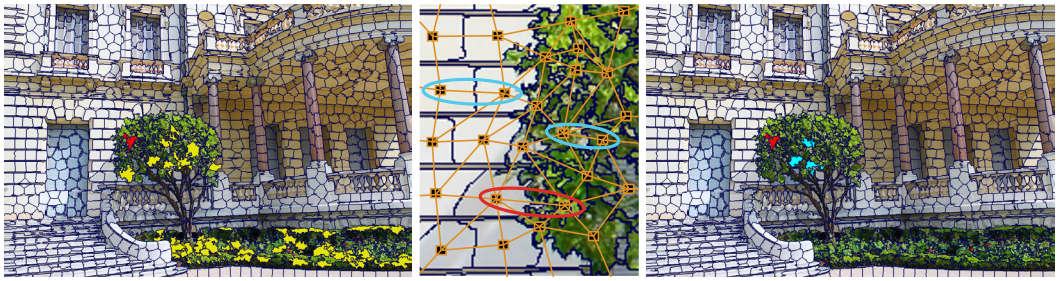
Similarly Zheng *et al.* (2009a) use depth maps, over-segmentation and matting – but in a joint optimization – to keep consistency of segments and depths. They merge superpixel segments if they contain one single depth. On top of each superpixel they construct a 2D mesh. In rendering, they choose which segments should be mapped and their contribution (blending weights) from the three spatially closest cameras. Zheng *et al.* (2009a) use 3D warp and blending with soft z-buffer to resolve depth inconsistencies. They also extend color of segments to fill holes produced by dis-occlusion. The over-segmentation without shape regularization degraded the quality, therefore they presented an off-line version with depth guided inpainting to fill holes. Their rendering method was used in Zheng *et al.* (2009b) to create parallax effect from still photographs.

The quality of previous approaches heavily depends on the accuracy of reconstructed geometry. To deal with cases when we have a sparse set of 3D points, Chaurasia *et al.* (2011) introduced a variational warping that avoids distortion of image content

with piece-wise regularization. The inputs to the rendering algorithm are a sparse set of calibrated input views, 3D points and user annotated relevant silhouettes (around foreground objects). With ULR blending weights, the novel view is rendered blending intermediate warped views and filling holes left by the forward mapping. To implement the variational warp, Chaurasia *et al.* perform 2D *Delaunay* triangulation on the photos with uniformly distributed vertices except in predefined silhouettes, where they density vertex samples. With more vertices in contours they create an elastic band to resolve distortions caused by the transformation and simulate occlusion and dis-occlusion. The warp optimizes the position of vertices given by this band, and regularizes by a piece-wise rigid transformation. This regularization is called *As-Rigid-as-Possible* and was introduced in Computer Graphics by Alexa *et al.* (2000). It enforces the constraint that each triangle of the warping mesh can only be scaled, rotated or translated. Thus triangles compete among selves to keep rigidity. This regularization had been also implemented in video stabilization applications (Liu *et al.*, 2009). Warping the whole image with big foreground objects and large baselines can introduce distortions. Their blending strategy avoids blending several images which minimizes ghosting artifacts but produces temporal artifacts like popping.

To automatically preserve silhouettes Chaurasia *et al.* (2013) opted for a greedy approach. They over-segment the image into superpixels with *SLIC* (Achanta *et al.*, 2012) under the hypothesis that superpixels preserve discontinuities and hence contain one single depth. They deal with non-reconstructed regions and approximation of geometry in preprocessing and during rendering. In preprocessing, they propagate front-parallel depth information to non-reconstructed regions from reconstructed regions. With a graph-based data structure, Chaurasia *et al.* (2013) propagate depth samples based on the appearance and proximity of superpixel segments (see Fig. 2.12). During rendering, they used a *shape preserving warp*: segment-wise 2D transformation driven by re-projection constraints and *as-rigid-as-possible* warps. With a multi-view data structure they keep track of which superpixel can be blended and as Zheng *et al.* (2009a) they render extra pixels around each superpixel to fill empty spaces produced by the forward transformation. A final Poisson hole filling render pass completes empty regions. To render with the shape preserving warp, thousands of linear equations (one per segment) must be solved; thus the rendering approach requires high-end (the linear systems are solved on the CPU) hardware (to be able to solve linear in real time), which means that it may not run on all computing platforms like mobile phones.

**Hybrid Approach.** Another approach that tackles drawbacks of incomplete reconstruction was presented by Lipski *et al.* (2014). Their hybrid algorithm leverages advantages of both correspondences-based (discussed in Section 2.2.2) and explicit geometry-based methods. They compensate for inaccurate scene geometry by incorporating visually plausible correspondences into the rendering equation, however, estimate plausible dense correspondences can be as difficult as estimate depth. From dense correspondences and calibrated cameras they obtain an explicit representation of the scene. For rendering, instead of mapping and interpolating color directly in image space, Lipski *et al.* interpolate in world space to compensate for mismatches in 3D



(a) (b) (c)

Figure 2.12: **Depth synthesis of of Chaurasia *et al.* (2013).** (a) Over-segmentation of an input view. The superpixel in red did not contain any depth (target superpixel). Superpixels with similar appearance (saturated in green) potentially offers approximated depth. (b) Chaurasia *et al.* create a graph structure with connections on adjacent superpixels. Weights of edges depend on the appearance of neighboring superpixels (high weights highlighted in cyan and low weight highlighted in red). (c) Target superpixel obtains depth sampled from the three nearest neighbors in the graph (superpixels in cyan).

position of depth pixels in correspondences (see Fig. 2.10b). This approach reported a time consuming per-possesing (hours per pair of images) and a manual foreground segmentation to compute reliable correspondences which is a very difficult problem.

In this thesis we use forward-mapping methods, specifically, we build on Chaurasia *et al.* (2013) which has been demonstrated to outperform previous methods in terms of image quality in most cases.

### 2.3.3 Handling Reflections

The methods review so far assumed that the scene can be approximated by a single geometric layer (proxy). This assumption is violated in reflective and semi-transparent surfaces. Based on *layered depth images*, Lischinski & Rappoport (1998) represent separately both view-dependent and view-independent appearance of synthetic scenes. Their IBR algorithm recombines the two layers together in a manner that produces an approximation to the correct image.

Sinha *et al.* (2012) observed that for real scenes, there is a reflective component and a refractive component that have different geometries. They separate the stereo matching process into these two-layers. They also separate the appearance of the two layers. Then they compute piecewise-planar proxies with local-plane fitting and seed-and-grow to cluster planes. For rendering they move each piece of the image according to their layered geometry and additively combine them from two closest images. Separating these two layers is not always clean producing *ringing* artifacts. Sinha *et al.* (2012)'s method does not generalize to reflections and could be only applied to strong planar reflectors (e.g., a glossy painting in a museum).

Instead of explicitly separating the scene into these components, Kopf *et al.* (2013) are able to deal with general scenes with reflections. They do so by synthesizing views in the gradient domain. Depth gradients are estimated for pixels instead of depth itself.

In rendering they splat gradients into the novel view, use additive blending and integrate to recover the color. They compute an approximate integration by solving a *Poisson* problem and apply weakly weighted data term to regularize the solution. Gradients of contours occlude each other but not necessarily gradients of reflective areas. Kopf *et al.* suggested an heuristic to detect which gradient should vanish because of occlusion. The method produces artifacts when incorrect depth values are associated to gradients e.g., for horizontal gradients with horizontal camera motions and busy textures like vegetation. This limits the application range.

### 2.3.4 Optimization and Learning-based Approaches

A different set of approaches use optimization and learning to directly synthesize novel views. Fitzgibbon *et al.* (2005) reconstruct color rather than depth. In a Bayesian framework, they estimate the most likely novel view given the input images, calibrated cameras and novel camera. The color to be synthesized has the Maximun a Posteriori probability given the colors that lie in the epipolar lines of input images. Fitzgibbon *et al.* (2005) regularize with statistics of input textures (patches around the selected depth). The off-line optimization was speeded up in Woodford & Fitzgibbon (2005). They used small baselines and reported some ghosting that depends on the size of the patches (prior).

Pujades *et al.* (2014) also use a Bayesian framework to formulate mathematically a description of the physical principles behind ULR's heuristics. They consider uncertainty of reconstruction as a gap around the surface. Re-projection color lookups propagate this gap of uncertainty. By considering the proxy uncertainty, they penalize the *minimum angle deviation* property of non-reliable geometry. Pujades *et al.* measure un-reliability of the proxy, based on photo-consistency of re-projection.

A remarkable application that yielded a commercial product[11] of IBR was presented by Kopf *et al.* (2014). They stabilize videos that were recorded in long interval of time (e.g., 1 frame per second) by very irregular motion - and thus very shaky. With calibrated frames and per-frame proxies, they optimize for a smooth rendering path where one of the criteria was the rendering quality. Given a proxy, they showed that ULR's *minimum angle deviation* does not always correlate with rendering artifacts. Kopf *et al.* (2014) used invariant texture stretching as rendering quality measure. They generate the novel video with an optimized path by rendering, stitching, and blending selected source frames appropriately.

Currently, there is a new research trend in view synthesis with machine learning algorithms like *Convolution Neural Networks*. A novel application of deep learning in view synthesis has emerged (Flynn *et al.*, 2015). Flynn *et al.* hypothesis says that deep networks trained with input cameras and poses can learn to predict appearance of novel views. Relationships of images-cameras and novel view are way too complex and the network would need to learn re-projection and encode epipolar constraints. Instead they train with a stack of re-projected input images at variable depths (*plane sweep-volume*). The network's architecture contains two blocks: one to select depth and the other to predict

[11] http://research.microsoft.com/en-us/um/redmond/projects/hyperlapseapps/

color. This system has limitations of speed, scalability and dependency of resolution and depth quantization.

Another deep learning method re-formulates the view synthesis problem as a pixel-copy task. (Zhou *et al.*, 2016). Zhou *et al.* Observed that visual appearance of nearby views are highly correlated to input views. They train a CNN to predict 2D vectors to reconstruct the novel view (*appearance flow*). This formulation does not require plane sweep-volume that prohibits view extrapolation for Flynn *et al.* but the rendering results are still far from high quality rendering. Although these two attempts are promising for the future, they do not out perform the quality of previous methods and are still far from achieving real-time performance.

**Discussion** Human vision uses a variety of depth cues to interpret 3D structures. Studies show that users of IBR hardly notice perspective distortions of complete reconstructions (Vangorp *et al.*, 2011, 2013), however, users can easily identify artifacts and missing reconstruction especially for free view-point IBR with cameras close to the scene. In spite of impressive advances in image-based reconstruction techniques, they still present missing parts of the scene. The big challenge of modern IBR techniques consists in generating views that could provide images with convincing realism from a perceptual perspective, even with incomplete information about the 3D world.

In this thesis we address this problem adapting incomplete information for rendering. Specifically, in Chapter 3 we use a set of plausible navigation IBR methods and we estimate per-region which of them allows better rendering quality. In Chapter 4 we use databases of 3D models to automatically query, align and morph models of cars in our scenes. The morphed meshes do not need to adapt precisely to the exact overall geometry but only adapt to the image silhouettes. Finally, in Chapter 5 we present two possible approaches (feature and learning based) to detect incomplete and inaccurate reconstruction information for rendering.

CHAPTER 3

# A Bayesian Approach for Selective Image-Based Rendering using Superpixels

## Contents

## 3.1 Introduction

As we have seen in the review of previous work in Chapter 2, new Image-Based Rendering algorithms (Zitnick & Kang, 2007; Eisemann *et al.*, 2008; Goesele *et al.*, 2010; Chaurasia *et al.*, 2013; Kopf *et al.*, 2013; Lipski *et al.*, 2014), build on and improve the original methods where geometry was either not used (Levoy & Hanrahan, 1996) or provided (semi-) manually (Debevec *et al.*, 1996; Gortler *et al.*, 1996; Buehler *et al.*, 2001). Recent IBR algorithms often treat specific cases very well, e.g. the floating textures algorithm (Eisemann *et al.*, 2008) reduces ghosting, shape-preserving warps (Chaurasia *et al.*, 2013) allow plausible rendering of badly-reconstructed regions (low texture, vegetation) and gradient domain rendering (Kopf *et al.*, 2013) treats reflections. These methods typically sacrifice performance for quality to treat hard cases; in well reconstructed regions, simpler and faster methods (Buehler *et al.*, 2001) perform very well. We see that each IBR algorithm has different quality/speed trade-offs, depending on the specific scene and cases it treats, and that no single algorithm is better than all others for all cases. In addition, each method has different parameters which directly affect rendering quality. Modeling such complex rendering processes to improve novel view synthesis is hard, due to the complexity of the solutions and the data, which are often uncertain (e.g., 3D reconstructions, camera calibration). In this dissertation we introduce a general Bayesian approach that models different IBR algorithms but also the possibility to *choose* between them.

Bayesian methods have been used in IBR to improve image quality for specific algorithms (Fitzgibbon *et al.*, 2005; Pujades *et al.*, 2014). Our analysis will share some common tools with these approaches, but our goal is real-time IBR. It can be considered complementary to these and allows the combined use of several different IBR algorithms by choosing between them in a local manner, i.e., at the level of image regions (see Fig. 3.1). In Section 3.2, we first present this approach in general terms which can be used in the context of several different algorithms. Our Bayesian methodology provides an intuitive description of the problem and takes the full set of complex factors into account. This formulation expresses the likelihood of a choice of rendering method by taking into account the rendering quality, the priors given the assumptions about the scene as well as the rendering algorithm and its parameters, which can be interpreted as the optimizations performed in the methods described here. We solve a Maximum a Posteriori (MAP) estimation to choose the rendering process at the granularity we target (pixel or image region). We do this by applying Bayes rule and computing posterior probability densities on rendering quality and choice of rendering process.

For the purposes of this work, we will concentrate in modeling the *choice* of algorithm, as well as rendering quality, with the assumption that the rendering algorithm and parameters are fixed. To demonstrate the utility of our framework, we apply our general approach to the class of IBR algorithms based on oversegmentation (superpixels) presented in the Section 3.3. These achieve high rendered image quality by preserving silhouettes. In this algorithmic class, we use the algorithm of Zitnick & Kang (2007) as a baseline. It uses fronto-parallel depth to render superpixels and is thus fast. We also consider the recent algorithm of Chaurasia *et al.* (2013), which uses a shape-preserving warp to

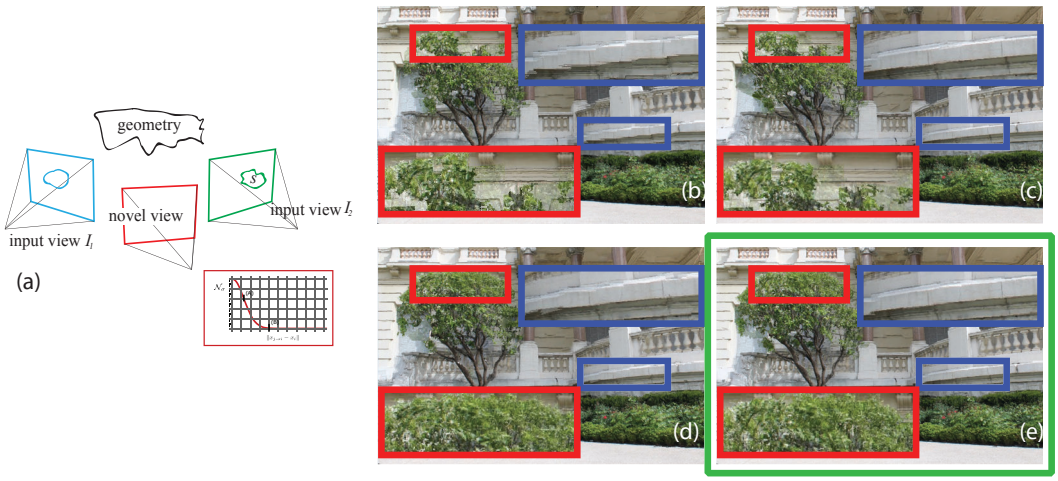



Figure 3.1: **Overview.** We propose a Bayesian formulation (a) to model rendering quality for different Image-Based Rendering (IBR) algorithms, and a Maximum a Posteriori estimation to select the algorithm producing the highest probability result for a given image region. We apply our algorithm to three IBR methods which use oversegmented input images, each having different speed/quality tradeoffs. In (b), we use planes fronto-parallel to the input view which fail for trees and slanted planes. Using local plane estimation (c) the result is improved, especially for slanted planes (blue box). Using the shape preserving warp (d) of Chaurasia *et al.* (2013), better results are achieved for the tree (red box), but the quality of the slanted planes is worse. Our algorithm (e) makes the correct choice locally, giving the best solution in each case.

regularize rendering of superpixels in hard case (in Section 3.3.3). This approach has been demonstrated to be superior in quality to previous methods especially for free-viewpoint navigation, but involves an expensive warping step during rendering. We also include an intermediate approach (in Section 3.3.2), which uses planar estimation of superpixels similar to that of Bodis-Szomoru *et al.* (2014). We render planar segments with a method akin to View-Dependent Texture Mapping (Debevec *et al.*, 1998).

In a preprocessing step, we estimate probability densities independently at each superpixel to allow real-time selective rendering at runtime. The probability density function representing rendering quality is expressed using both geometric and photometric errors in re-rendering existing input views. In preprocessing, the MAP estimation on the three possible rendering processes assigns the best choice to each superpixel based on our Bayesian formulation, and our selective IBR algorithm efficiently generates high-quality novel viewpoints in real-time accordingly.

Our main contributions are:

- A new Bayesian formulation to model the choice and quality of rendering algorithms for IBR.
- A selective IBR algorithm for oversegmentation-based methods that chooses the rendering method most suited for a given superpixel in a preprocessing step, allowing high-quality real-time rendering.

Thus, the new algorithm provides an efficient and practical IBR algorithm which offers the best of previous methods by selecting the most suitable solution in a principled manner. Our implementation shows that the selective rendering algorithm is much faster with equivalent or even better quality than the best of the three approaches taken separately.

## 3.2 Bayesian Formulation

As mentioned in previous work (Chapter 2), Image-Based Rendering uses input data which is inherently inaccurate and incomplete, e.g., the 3D geometric reconstruction, the camera calibration, amount others. The actual rendering process contains *uncertainty*. The nature of this uncertainty is unknown, and thus, rather than model uncertainty to improve an algorithm, we model the *choice* between different rendering algorithms instead.

We next introduce our Bayesian approach to model this *choice*. Our final goal is to compute the best quality image, which we model as being *the most likely image* in a probabilistic sense (Bishop *et al.*, 2006). The preprocessing step we describe next will assign a rendering algorithm to each superpixel of each image. At runtime each superpixel will be rendered using the chosen rendering algorithm.

### 3.2.1 A Bayesian Approach to IBR

We define a probabilistic model of the rendering function that generates novel images $I$. The rendering function is very general and corresponds to the set of three rendering methods we consider (i.e., Debevec *et al.* 1998; Zitnick *et al.* 2004; Chaurasia *et al.* 2013). Each of these rendering methods are respectively characterized by the sets of parameters $\xi_1, \xi_2$ and $\xi_3$.

These parameters represent all the necessary information needed by the rendering function to estimate images for new viewpoints. We define the label $l_i^s$ that identifies which of the three rendering method is used for each superpixel $s$ of the input view $i$.

Noting $\xi = \{\xi_1, \xi_2, \xi_3\}$ the set of all rendering parameters and $L$ the vector of all labels $l_i^s$, we define the probability distribution $p(\xi, L|I)$ which expresses the likelihood of a choice $L$ of rendering method with parameters $\xi$ given reference images $I$. To estimate this distribution, we use a generative model (Bishop *et al.*, 2006) as we will explicitly model inputs (input images) and outputs (renderings). The model describes the method of rendering new viewpoints as follows:

$$p(\xi, L|I) = \frac{p(I|\xi, L)p(\xi)p(L)}{p(I)} \tag{3.1}$$

The denominator $p(I)$ is a normalization factor and since we will be maximizing likelihood, we can ignore it, leading to the simpler expression:

$$p(\xi, L|I) = p(I|\xi, L)p(\xi)p(L) \tag{3.2}$$

We define this model for rendering methods based on superpixels (Debevec *et al.*, 1998; Zitnick & Kang, 2007; Chaurasia *et al.*, 2013), but it can be applied to any rendering

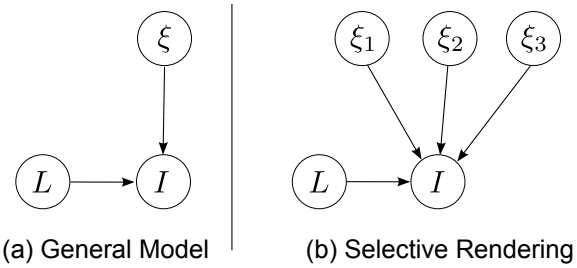



Figure 3.2: Probabilistic graphical models for selective IBR: (a) In the general model, the rendering of image $I$ is estimated according to label $L$ that indicates which rendering method to use with parameters $\xi$. (b) Selective rendering uses a set of 3 rendering methods. Each rendering method is described by its parameters $\xi_1$, $\xi_2$ and $\xi_3$. These parameters can be, for example, the number of superpixels. For superpixels, $L$ is a vector of labels specifying which rendering method to use for each one of them.

method. The selection can be defined for different parts of the process, e.g., image regions in input images, input or output camera positions, specific pixels, etc.

The general relation described by this generative model is illustrated in Fig. 3.2(a), along with the specific case of selective rendering in Fig. 3.2(b)

**Rendering quality** We model rendering quality with the term $p(I|\xi, L)$, which expresses the likelihood to generate images $I$ given vectors of labels $L$ and parameters $\xi$. It models the uncertainty in the rendered image due to the errors associated with the rendering method selected by $L$. More precisely, high probability $p(I|\xi, L)$ means that the image $I$ is close to the result obtained with the rendering method selected by the state variable $L$. In Section 3.4, we use this rendering quality to choose the rendering method for each superpixel of the input images.

**Priors on rendering parameters** We consider the rendering methods used by the rendering function as black boxes. The prior $p(\xi)$ is thus considered uniform and we do not need to further develop the list of parameters of each method. Our assumption is that, independently, each algorithm is close to optimal and our objective is to find the best way of combining them. However, if the goal is also to improve the rendering algorithm, such a PDF can be used to favor a certain set of parameters.

**Prior on the choice of rendering method** The probability $p(L)$ is a prior on the choice of rendering algorithm. It does not depend on the resulting images but only on the selected method. We can use it for example to favor a specific rendering method when we expect it to perform better in a given context. For example, if we have very precise geometry and images close to the novel view, direct warping or reprojection of input images will work well, while if the reconstruction is sparse or uncertain, the shape-preserving warp will work better. In our generative model, $p(L)$ is the the most general term. Each rendering algorithm has a different set of implicit and explicit assumptions that relate to the geometry

or to appearance (image pixels); $p(L)$ may depend on geometry or color and the challenge is to propose a distribution $p(L)$ which best expresses how well these assumptions are met.

**Discussion** In summary, we now have a probabilistic framework for image based rendering that models the relation between different algorithms and the final rendered image. Instead of estimating the most probable image which is time consuming (Pujades *et al.*, 2014), this probabilistic model can be used to choose a real-time rendering method and its parameters in a pre-process. Adopting a probabilistic approach has several advantages. First, we delay any decisions until all factors have been considered, helping us avoid incorrect decisions early on which cannot be subsequently corrected. Second, we compute an overall probability for a given choice of algorithm which allows us to improve the final blending step of rendering. Finally, our formulation can be easily adapted to different algorithms.

### 3.2.2 Rendering selection as MAP estimation

Using the proposed generative model, we can express the selection of the rendering method $L^*$ as a MAP estimation:

$$L^* = \arg\max_L p(I|\xi, L)p(\xi)p(L) \tag{3.3}$$

Quality measures on the rendered images to estimate are used to select the rendering method $L$. These rendering methods are treated as black boxes and we do not impose any prior on their parameters, so $p(\xi)$ can be ignored in further development:

$$L^* = \arg\max_L p(I|\xi, L)p(L). \tag{3.4}$$

To solve the above equation, we would ideally need to evaluate Eq. 3.4 over a large number of images $I$. Unfortunately, this is impossible since these images are not available. As an approximation, we can evaluate the density $p(I_i|\xi, L)p(L)$ for each input image $I_i$. We do this by rendering all *other* input images $\{I_1, \ldots, I_{i-1}, I_{i+1}, \ldots, I_n\}$ into the viewpoint of $I_i$, and evaluating how well the synthesized image matches the ground truth input $I_i$. This MAP estimation can thus be performed as a pre-process.

## 3.3 Superpixel IBR Algorithms

To demonstrate the utility of our approach, we will apply it in the context of a specific class of IBR algorithms that use over-segmented input images. As explained previously, IBR methods based on oversegmentation achieve high quality by preserving silhouettes while maintaining real-time performance; they are thus suited to our objectives. In preprocessing, we use the proposed Bayesian model to choose the best method for each superpixel in a principled way, allowing fast and high quality rendering at runtime. In what follows we assume that the input is a set of over-segmented images from different viewpoints, processed by SfM and MVS. We thus assume that a set of reconstructed points $\mathcal{X}^s$ is assigned to each superpixel $s$ of the oversegmentation.

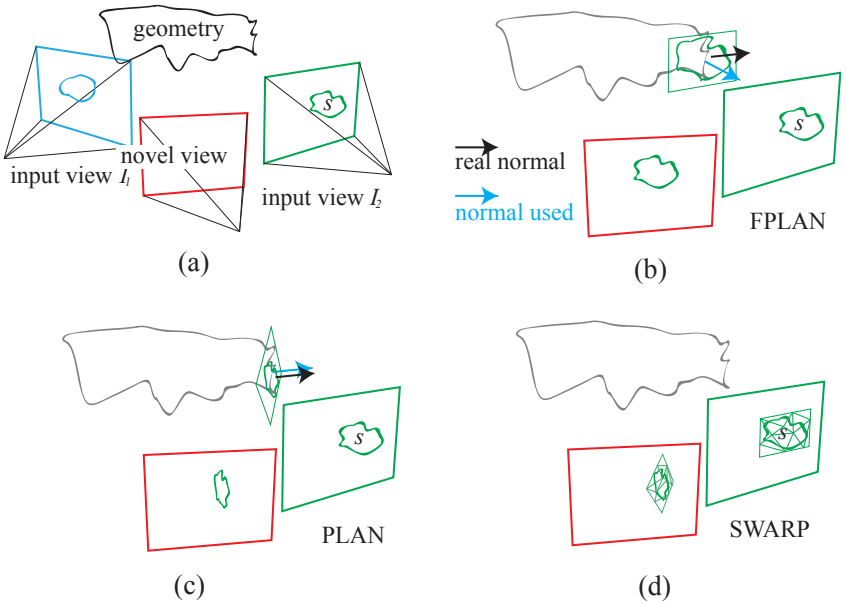



Figure 3.3: **IBR methods for our selective approach.** (a) The geometry, input cameras, and oversegmented images. (b) FPLAN: a fronto-parallel plane is assigned to superpixel $s$ (Zitnick *et al.*, 2004). (c) PLAN: a plane is estimated for $s$ similar to Bodis-Szomoru *et al.* (2014). (d) SWARP: a shape preserving warp is applied to $s$ in image space (Chaurasia *et al.*, 2013).

### 3.3.1 Fronto-parallel superpixels

We consider the algorithm of Zitnick *et al.* as the baseline algorithm which oversegments the input images, and uses fronto-parallel depth for subsequent rendering. In our case for each superpixel $s$ we use the median depth of the 3D points $\mathcal{X}^s$. We extend the original method by using depth values hallucinated by propagation from similar superpixels in the image when a superpixel does not contain reconstructed geometry (Chaurasia *et al.*). Such superpixels are assumed to be fronto-parallel to the corresponding input camera. The novel view is rendered with forward projections and additive blending. We call this algorithm *Fronto-parallel PLANar rendering* (FPLAN).

### 3.3.2 Planar superpixels for IBR

As shown in Fig. 3.1(b), view synthesis with FPLAN can result in visual artifacts when the actual surface is grazing with respect to the camera position – at the pillar base, a curved surface looks distorted because segments cannot reproduce perspective effects. To achieve perspective of piece-wise planar regions, beside depth we must assign orientation to segments by estimating planes. Thus, we also propose an intermediate algorithm that enhances FPLAN with a local plane estimation.

In general, fitting planes to point clouds follows two strategies: top-down (Argiles *et al.*, 2011) versus bottom-up (Mičušík & Košecká, 2010; Bodis-Szomoru *et al.*, 2014). Globally, the former strategy takes the whole point cloud to iteratively fit and remove big planes with robust methods like *RANdom SAmple Consensus - RANSAC* Fischler & Bolles

(1981). In Appendix A we explain how we applied this strategy combined with color appearance models to estimate and recover the ground floor of the scene since commonly MVS reconstruction fails for this region. Another alternative fits planes to local regions. Given a noisy point cloud, estimating planes with the superpixel's local geometry is prone to localized noise and results in rendering artifacts. We first present a strong conservative filter to remove local noise in Section 3.3.2.1 and in Section 3.3.2.2 we present a procedure to assign planes to superpixels after filtering. In Section 3.3.2.4 we explain how to render a novel view with this approach called *PLANar reprojection* (PLAN).

#### 3.3.2.1 Filtering Outliers

Feature points for reconstruction usually occur at color gradients and discontinuities. The boundary of a superpixel segment also occurs in color gradients. By construction, superpixel boundaries correspond to feature points that can be at depth discontinuities. As a result, 3D points close to the superpixel boundaries can often be at multiple depths which makes them unreliable for plane estimation. We design a simple filter to avoid multiple depth and filter out possible outliers. For each superpixel we define a 3D sphere centered in $X_c^s \in \mathcal{X}^s$. The radius of the sphere is proportional to the size of the superpixel and to the distance from the camera. We thus filter out points in $\mathcal{X}^s$ that lay outside the sphere.

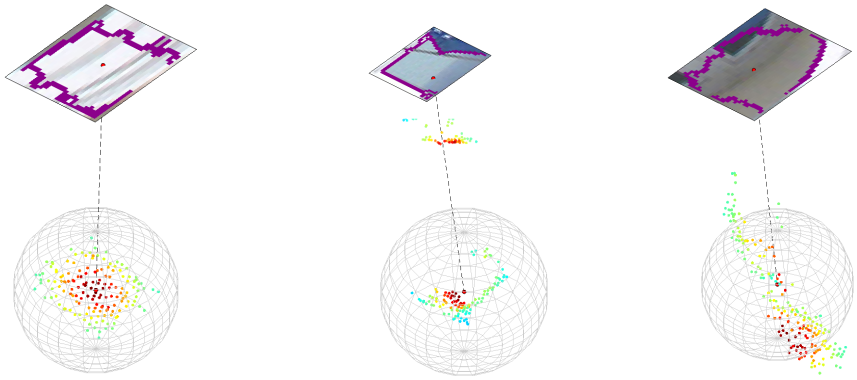
Given $x^s$, the set projections of $X^s \in \mathcal{X}^s$, we obtain $x_c^s$ as the *weighted geometric median* of $x^s$. The geometric median is the point that minimizes the weighted sum of distance to all points and provides a robust estimator for the location of the uncorrupted data even when up to half of the sample data may be arbitrarily corrupted. To discourage MVS points near boundaries, weights $w_{n\_i}$ in Eq. (3.5) are the values of a normal distribution centered at the centroid of the 2D position of pixels inside $s$ and standard deviation the minimum distance of the centroid to a contour of $s$. The equation for the weighted geometric median is:

$$\arg\min_{x_c^s} \sum_{i}^{|\mathcal{X}^s|} w_{n\_i} \|x_c^s - x_i^s\| \tag{3.5}$$

We unproject the farthest pixel in $s$ with the same depth as $x_c^s$. The distance between this 3D point and $X_c^s$ is the radius of the sphere centered in $X_c^s$. Any points outside this sphere are rejected. Intuitively, in this procedure the size of a volume that encloses 3D samples is proportional to the distance from the camera and proportional to the size of the 2D segment. In Fig. 3.4 we show some examples of this procedure after centering the sphere.

#### 3.3.2.2 Plane Estimation

After filtering out noisy 3D samples as described above, we follow a similar approach to Bodis-Szomoru *et al.* (2014) to generate plane hypothesis. Given the filtered 3D samples $\mathcal{X}^s$, we estimate a plane $\pi^s$ for the segment $s$ with RANSAC. The base of a plane model is defined by three points samples. Iteratively, we select a random base in $\mathcal{X}^s$, fit a plane $\pi_i$ to the selected samples and compute the score of that base. Points within a distance $\tau$ from $\pi_i$ are the inlier points for the base. The base with the highest score and the inlier



(a) (b) (c)

Figure 3.4: **Geometric median sphere filtering.** In the upper part of the images, segments with superpixel contours in magenta. The red point inside the superpixel is the weighted geometric median of depth samples. Weights of 3D points are color coded (the higher a weight, the closer to red). All the 3D points outside the sphere are ignored during plane estimation. (a) All points are selected. The superpixel is well formed with a single depth. (b) Two clusters of 3D points belong to the superpixel (two depths). Fitting a plane to all points would give us an slanted plane. Only the points inside the sphere are used for plane estimation. (c) Multiple points along a 3D corner. We select a subset where most points are concentrated far from contour.

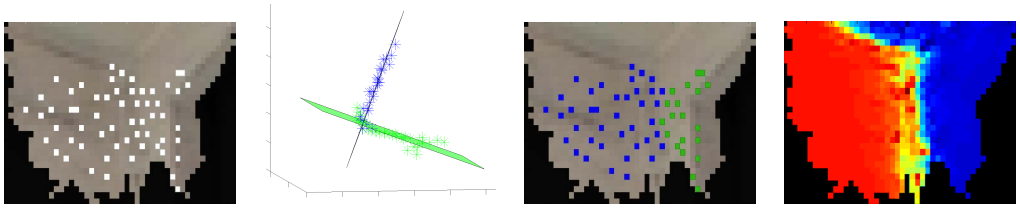
points from this base define the final plane $\pi^s$. As score, we do not use the inlier count but the weighted sum of point to plane distances $d(\cdot,\cdot)$ in Eq. (3.6). Weights $w_{X^s}$ represent the confidence of reconstruction of each point (usually normalized cross-correlation of photometric consistency) and $\tau$ is the inlier threshold. Finally, the score is given as follows:

$$score_i = \sum_{X^s \in \mathcal{X}^s} w_{X^s} \exp\left(-\frac{1}{2\tau^2} d^2\left(X^s, \pi_i\right)\right) \tag{3.6}$$

#### 3.3.2.3 Improvements

We test stability of planes with Single Value Decomposition and an heuristic considering visibility reasoning and discouraging sharp orientation of planes with respect to the camera they originate from. Further improvements for PLAN could be done by merging superpixels with the same plane description or splitting them when a single segment contains two or more planes as described in Fig. 3.5.

Concurrently with Bodis-Szomoru *et al.* (2014) and in a closely related direction, we attempted to regularize planes assigned to superpixels with pairwise multi-view and intra-view constraints. However, they attempted fast and light-weight representations of urban scenes, while our ultimate goal is rendering. Solving this problem would imply a new reconstruction algorithm, which is beyond the scope of this thesis. Instead, we follow the procedure described in Section 3.3.2.2 and with an imperfect reconstruction, leave our Bayesian Approach to select the most reliable rendering algorithm, given that *PLAN* is the least reliable in many cases.



(a) (b) (c) (d)

Figure 3.5: **Splitting superpixels and Matting.** PLAN reconstruction could be improved by splitting superpixels that contains multiple planes. (a) A superpixel in a corner of a building with depth samples in white. (b) With *J-linkage* (Magri & Fusiello, 2014) we can robustly identify that the samples actually describe two instances of planes. (c) We can observe two groups of depth samples (in green versus blue) that belong to two different planes. (d) We could separate these two planar regions with *Spectral matting* (Levin *et al.*, 2008). The two regions (red and blue) separated by a matte layer (in yellow/green).

#### 3.3.2.4 Rendering

If a superpixel $s$ is well approximated by a plane, we define a planar quadrilateral bounding the superpixel and transform the superpixel to the new view using standard *OpenGL* projection. We assume for now that the quadrilateral is a good approximation of the geometry corresponding to the superpixel since our probabilistic model will identify other cases as discussed below. Note that the actual rendering uses $s$ as a mask and only renders pixels of the rendered quadrilateral which correspond to the region of $s$ Chaurasia *et al.* (2013). This algorithm can be seen as a combination of Zitnick *et al.* (2004) and View-dependent Texture Mapping (Debevec *et al.*, 1998).

### 3.3.3 Superpixel warp

The highest quality oversegmentation-based IBR method we consider is shape-preserving warp Chaurasia *et al.* (2013). For each input view, the shape-preserving warp algorithm (SWARP) takes as input the set of superpixels and the corresponding reconstructed 3D points. The method then propagates depth into superpixels which do not contain reconstructed geometry over the mesh triangles. Rendering proceeds by building a small mesh of triangles over each superpixel $s$ and performing an image-space warp of the mesh into the novel view, using shape-preserving constraints and 3D reconstruction. This algorithm, though computationally expensive, handles poorly reconstructed regions and allows free-viewpoint navigation far from the input viewpoints. If the reconstruction of the model corresponding to 3D space covered by the superpixel $s$ is of high quality and if $s$ is (almost) planar, the warp is wasteful since it will give essentially the same result as direct reprojection. However, when the quality of the reconstruction is uncertain or unknown, the shape-preserving constraints will dominate and provide a plausible solution in many cases. Clearly the two rendering algorithms are somewhat complementary, and by carefully selecting which one to use, the resulting renderer will provide equivalent or even better quality at a lower cost.

## 3.4 Map Estimation for Rendering

We now have three rendering algorithms based on image over-segmentation, *PLAN, FPLAN,* and *SWARP*. In this section we show how the probabilistic framework presented in Section 3.2 can be used to select which rendering method should be considered for a given superpixel.

### 3.4.1 MAP selection at superpixel level

With the rendering methods precisely defined we can adapt the general formulation (Eq. 3.4) to our specific scenario. As already mentioned, the observations for the MAP estimation are the input images $\{I_1, I_2, \ldots, I_n\}$ and we can rewrite Eq. 3.4 as:

$$L^* = \arg\max_L \prod_{i=1}^{n} p(I_i|\xi, L)p(L) \tag{3.7}$$

To find $L^*$ the MAP estimate of the labels $L$, we need to evaluate $p(I_i|\xi, L)$. In this MAP estimation, we generate the image corresponding to the viewpoint of $I_i$ using the images $I_j$ ($j \neq i$). To do this, the images $I_j$ are transformed to the viewpoint of $I_i$ using depth and/or shape-preserving warps and blended together. If we note $\mathcal{R}_{j\to i}$ the *rendering* obtained by transforming $I_j$, then creating the approximation $\tilde{I}_i$ to the input image $I_i$ can be expressed as:

$$\tilde{I}_i = \sum_{j\neq i} \alpha_j \mathcal{R}_{j\to i} \qquad \text{with} \sum_{j\neq i} \alpha_j = 1 \tag{3.8}$$

So $p(I_i|\xi, L)$, which models the error in rendering, can be expressed as a function of the distance between $I_i$ and the transformed images $\mathcal{R}_{j\to i}$. With the assumption that improving any of these intermediate images improves the final blended image, we can write:

$$p(I_i|R, L) \propto \prod_{j\neq i} p(I_i|\mathcal{R}_{j\to i}, L_j) \tag{3.9}$$

The rendering methods reason on superpixels and novel viewpoints are generated by independently estimating superpixel transformations and blending them. Thereby, the choice of rendering algorithm must be made for each superpixel. The selection variable $L_j$ is defined as the vector of labels $l_j^s$ selecting the rendering method to use with each superpixel $s$ from image $I_j$. We can now expand the expression for the MAP estimation:

$$L^* = \arg\max_L \prod_{i=1}^{n} \prod_{j\neq i} p(I_i|\mathcal{R}_{j\to i}, L_j)p(L_j) \tag{3.10}$$

In our case the possible values for $l_j^s$ are $\{PLAN, FPLAN, SWARP\}$. Assuming that rendering is independent between the superpixels, the MAP estimation becomes:

$$L^* = \arg\max_L \prod_{i=1}^{n} \prod_{j\neq i} \prod_{s} p(I_i|\mathcal{R}_{j\to i}^s, l_j^s)p(l_j^s) \tag{3.11}$$

Maximizing the previous probability can be done independently for each superpixel and the MAP equation for each superpixel label is thus:

$$l_j^{s,*} = \arg\max_{l_j^s} \prod_{i \neq j} p(I_i | \mathcal{R}_{j\to i}^s, l_j^s) p(l_j^s) \tag{3.12}$$

This equation allows the selection of the rendering algorithm. Note that starting from the general Eq. 3.4 and by leveraging rendering algorithm properties, we derive a model that expresses the same ideas at the level of superpixels. In this case, $p(I_i|\mathcal{R}_{j\to i}^s, l_j^s)$ expresses the quality of rendering superpixel $I_j^s$ in the different view $i$ using the rendering algorithm $l_j^s$. The probability $p(l_j^s)$ is the prior on the choice of rendering algorithm $l_j^s$ and is considered uniform over all the labels. In the following, we show how using only rendering quality for superpixels we are able to perform algorithm selection for rendering.

### 3.4.2 MAP selection using rendering quality

We model the probability distribution $p(I_i|\mathcal{R}_{j\to i}^s, l_j^s)$ as a function of the distance between the transformed image $\mathcal{R}_{j\to i}^s$ and the observed input $I_i$ image, using two distributions:

$$p(I_i|\mathcal{R}_{j\to i}^s, l_j^s) = p_{\text{geom}}(I_i|\mathcal{R}_{j\to i}^s, l_j^s) p_{\text{pho}}(I_i|\mathcal{R}_{j\to i}^s, l_j^s). \tag{3.13}$$

The first term corresponds to the geometric rendering quality. It expresses how well the 3D structure of the scene is preserved under the rendering transformation. The second distribution is based on appearance and will be referred to as photometric rendering quality. It models the error between the rendered image and the observation in terms of color differences. We also use occlusion information from MVS reconstruction estimating rendering quality only in viewpoints where the superpixel is visible.

#### 3.4.2.1 Geometric rendering quality

To render the image at the view of input image $I_i$, the superpixel $s$ will undergo a transformation corresponding to a warp (for $l_j^s = SWARP$) or a plane projection (for $l_j^s = PLAN$ or $FPLAN$). One way to measure the error in this transformation from a geometric point of view is to use reconstructed 3D points present in the superpixel.

We define $\mathcal{X}_j^s$ as the set of the 3D reconstructed points $X$ that project in the superpixel $s$ in view $j$. We denote $x_j$ the 2D position of the projection of $X$ in view $j$. As previously described, the superpixel $s$ undergoes a transformation to the viewpoint of an input camera. The points $x_j$ will follow the same transformation and their new position is noted $x_{j\to i}$. If the transformation is well estimated, then $x_{j\to i}$ and $x_i$ (the projection of $X$ in view $i$) should coincide. To define the geometric term, we use a Gaussian distribution defined on the distance between $x_{j\to i}$ and $x_i$ (see Fig. 3.6):

$$p_{\text{geom}}(I_i|\mathcal{R}_{j\to i}^s, l_j^s) = \prod_{X \in \mathcal{X}_j^s} \mathcal{N}_\sigma \frac{\|x_{j\to i} - x_i\|}{|\mathcal{X}_j^s|} \tag{3.14}$$

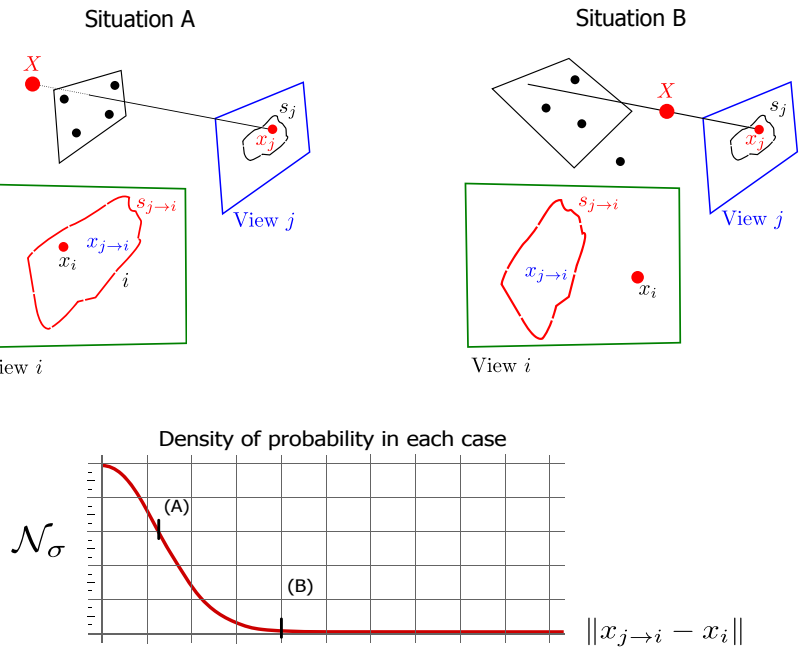



Figure 3.6: **Geometric rendering quality.** In these examples, a superpixel $s_j$ is transformed to image $i$ using a plane approximation. The geometric quality will depend on the distance between $x_i$, the projection of the 3D point $X$ in view $i$, and $x_{j\to i}$, the transformation of $x_j$ into view $i$. **Situation A:** The plane approximation is relatively good so the distance $\|x_{j\to i} - x_i\|$ is small and it results in a high value for $\mathcal{N}_\sigma$. **Situation B:** The transformation for superpixel $s$ is not well estimated and in this case the distance $\|x_{j\to i} - x_i\|$ is large resulting a small value for $\mathcal{N}_\sigma$.

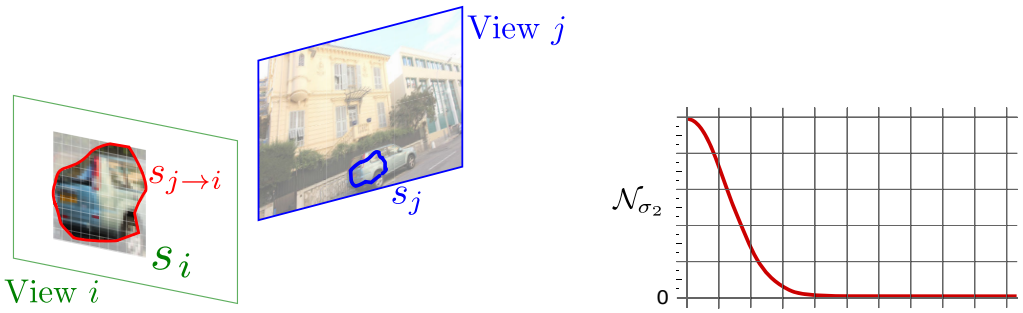



Figure 3.7: **Photometric rendering quality.** We compute the mean squared distance between the colors of the superpixel $s_{j\to i}$ and the superpixel $s_i$. The distribution $\mathcal{N}_{\sigma_2}$ will give a high density value when there is a strong match between the two superpixels.

If there are no reconstructed points, it is impossible to estimate a plane and so $p_{\text{geom}}$ is set to zero for *PLAN*. For *FPLAN* and *SWARP* depth will be propagated from neighbors. The choice between these two labels will only depend on $p_{\text{pho}}$.

#### 3.4.2.2 Photometric rendering quality

The objective is to estimate the rendering quality in terms of appearance. We denote $s_{j\to i}$ the result of transforming the superpixel $s$ to the image plane of camera $C_i$. To measure the rendering quality in terms of appearance, we use the mean squared distance (MSE) between the pixel colors of $I_j^s$ and $I_i^{s_{j\to i}}$. If the transformation is well estimated, the distance should be small. To define the photometric term, we use a Gaussian distribution defined on the mean squared distances between $I_j^s$ and $I_i^{s_{j\to i}}$ (see Fig. 3.7):

$$p_{\text{photo}}(I_i|\mathcal{R}^s_{j\to i}, I_j^s) = \mathcal{N}_{\sigma_2}(\text{MSE}(I_j^s, I_i^{s_{j\to i}})) \tag{3.15}$$

We note that other error measures could be considered but this was sufficient in our case. We also tried inverse normalized cross-correlation (NCC) which has similar results but it is computationally more expensive.

#### 3.4.2.3 Validation of rendering hypotheses

In general the probability associated with rendering hypotheses $p(L)$ (in Eq. 3.1) describes the compatibility of the rendering algorithm with the considered scene. In the context of our algorithm selection, we derive a similar probability $p(l_i^s)$ at the superpixel level.

For *PLAN, FPLAN*, we should choose the planar approximation if the 3D geometry corresponding to the superpixel is well approximated by a plane. To model this we could use principal component analysis (PCA) on the 3D points, and examine the weight associated with each principal component. If note $w_1$, $w_2$ and $w_3$ the decreasingly ordered weights associated with the three main components obtained from PCA. If 3D points are located on a planar surface, we have two main components and the third one negligibles. We could model this by defining a distribution on the bounded surface defined by the two variables $w_2/w_1$ and $w_3/w_2$. These two values are defined in the interval $[0, 1]$ and we naturally use the Beta distribution (Bishop *et al.*, 2006) (noted $\mathcal{B}_{\alpha,\beta}$) to model the validity of the planarity assumption with respect to $w_2/w_1$ and $w_3/w_2$:

$$p(l_i^s = PLAN) = \mathcal{B}_{\alpha_1,\beta_1}\left(\frac{w_2}{w_1}\right) \quad \mathcal{B}_{\alpha_2,\beta_2}\left(\frac{w_3}{w_2}\right) \tag{3.16}$$

With the distribution $\mathcal{B}_{\alpha_1,\beta_1}(\frac{w_2}{w_1})$ we can give lower probability density to small values of $w_2/w_1$ as the underlying geometry is more likely to be a line. On the other end, the distribution $\mathcal{B}_{\alpha_2,\beta_2}(\frac{w_3}{w_2})$ we can give higher probability density to small values of $w_3/w_2$ as the underlying geometry is more likely to be a plane. The Fig. 3.8 shows the resulting distribution $p(l_i^s = planar)$.

For *SWARP* there is no assumption related to geometry, since the algorithm uses the shape-preserving warp to get the best possible result, both for well- and poorly-reconstructed regions. Similarly, for *FPLAN* no assumptions are made. Since for *FPLAN* and *SWARP* we do not make any assumption, for our experiments, we decided not to include the planarity assumption for *PLAN*. Consequently, we assume a uniform distribution for all of them:

$$p(l_i^s = SWARP) = 1 \quad p(l_i^s = PLAN) = 1 \quad p(l_i^s = FPLAN) = 1 \tag{3.17}$$

We can now compute Eq. 3.7 for each superpixel of each image, for each of *PLAN, FPLAN, SWARP*. We discuss below how we use this estimation in a preprocessing step for our selective rendering algorithm.

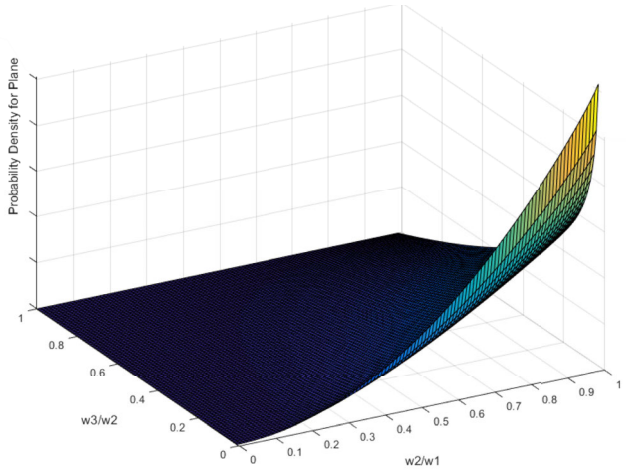



Figure 3.8: Example of probability distribution for $p(l_i^s = planar)$ with respect to PCA decomposition weights $w_1$, $w_2$ and $w_3$. When $w_2/w_1$ is small the underlying geometry is more likely to be a line and it less probable to obtain a good rendering result using a planar approximation. When geometry is closer to a plane ($w_2/w_1$ close to 1 and $w_3/w_2$ close to 0), using planar approximation for rendering is favored with higher values for $p(l_i^s = planar)$.

#### 3.4.2.4 Final labeling

To obtain a fast rendering algorithm we need to favor plane projection methods (*PLAN* and *FPLAN*) when they result in similar quality to the warp based approach. To this end we use a smaller value for $\sigma_2$ in the case of *PLAN* and *FPLAN* labels. Thanks to this, when both planar and warp based methods achieve good results, the planar rendering will be favored, resulting in significant speedup. We can now compute Eq. 3.12 for each superpixel of each image, for each of *PLAN, FPLAN*, and *SWARP*. We discuss below how we use this estimation in a preprocessing step for our selective rendering algorithm.

## 3.5 A Selective IBR Algorithm

The input to our approach is a set of images of a given scene, which have been processed by automatic camera calibration (e.g., VisualSFM Wu *et al.* 2011) and MVS reconstruction (e.g., Furukawa & Ponce 2010 or Jancosek & Pajdla 2011). These two steps provide camera calibration parameters, and a 3D point cloud of the scene, which can be sparse and inaccurate in regions with low texture or stochastic (e.g., vegetation) or reflective (e.g., cars) content.

**Preprocessing** For each image, we first run the superpixel oversegmentation of Achanta *et al.* (2010) and the depth synthesis as described in Chaurasia *et al.* (2013). We then perform the plane estimation for each superpixel, as described in Section 3.3.2. In a preprocessing step, we perform the MAP estimation on this data, following Eq. (3.7). This is done for each superpixel of each input image and each rendering algorithm, i.e. $L = \{PLAN, FPLAN, SWARP\}$. A rendering algorithm is then chosen for each superpixel in this preprocess.

**Rendering** Similarly to Chaurasia *et al.* (2013), the four spatially closest views to the novel view are chosen and each superpixel of these views is projected into the novel view. In contrast to previous methods, each superpixel is projected into the novel view using the choice of rendering algorithm $l_j^s$, as computed in the pre-processing step.

We can also use the probability of the chosen algorithm in the blending weights. Previous algorithms (Buehler *et al.*, 2001; Chaurasia *et al.*, 2013) use heuristic blending weights, i.e. if two superpixels do not contain reconstructed points, the background is preferred. Instead, for a superpixel $S$ we scale the blending weights with the probability density over all views for the chosen algorithm:

$$p_S = \prod_{i=1}^{n} \prod_{j \neq i} \prod_{s} p(I_i | R_{j \to i}^s, l_j^s) p(l_j^s) \tag{3.18}$$

The projection operation for *FPLAN,PLAN* uses standard OpenGL polygon rendering in the GPU, and is much cheaper than the superpixel warp. We measured a factor of approximately 3 times speedup, depending on the number of MVS points in each superpixel which add constraints to the warp. Speedup depends on the percentage of superpixels using the *SWARP*, as shown in the results.

**Implementation Details** The preprocessing step and rendering were implemented in C++ with OpenGL/GLSL shaders. For *SWARP* a triangle mesh covering superpixels is warped (Chaurasia *et al.*, 2013). The reconstructed points are used as constraints in the warp mesh. For Eq. (3.14) we use barycentric coordinates of the mesh triangle before the warp to determine their position in the same triangle after the warp. For Eq. (3.15) every rasterized patch is read back in RGB color space. This requires 2 min to process an image of 1M pixels. A subsequent implementation in CUDA does this computation directly on the GPU and instead of reading back at every iteration, we read the final computation only once. This reduces the labeling process to a few seconds per image.

## 3.6 Results and Evaluation

We ran our algorithm on twelve urban scenes with different conditions and one interior scene (Fancy_restaurant-26). The datasets ChapelHill1-25 and ChapelHill2-30 were captured by a moving vehicle at street-level in Pollefeys *et al.* (2008). The other datasets where captured with different DSLR-cameras and some of them have been referenced in other IBR publications as wi will see in comparisons. For all of them we calibrate cameras with *VisualSfM* structure from motion (Wu *et al.*, 2011). The scenes Yellowhouse-12, Museum-27, Street-10, Aquarium-20 were previously reconstructed with PMVS (Furukawa & Ponce, 2007) in Chaurasia *et al.* (2011, 2013). The scene Library_RM-50 was reconstructed with MVE (Goesele *et al.*, 2007) while for

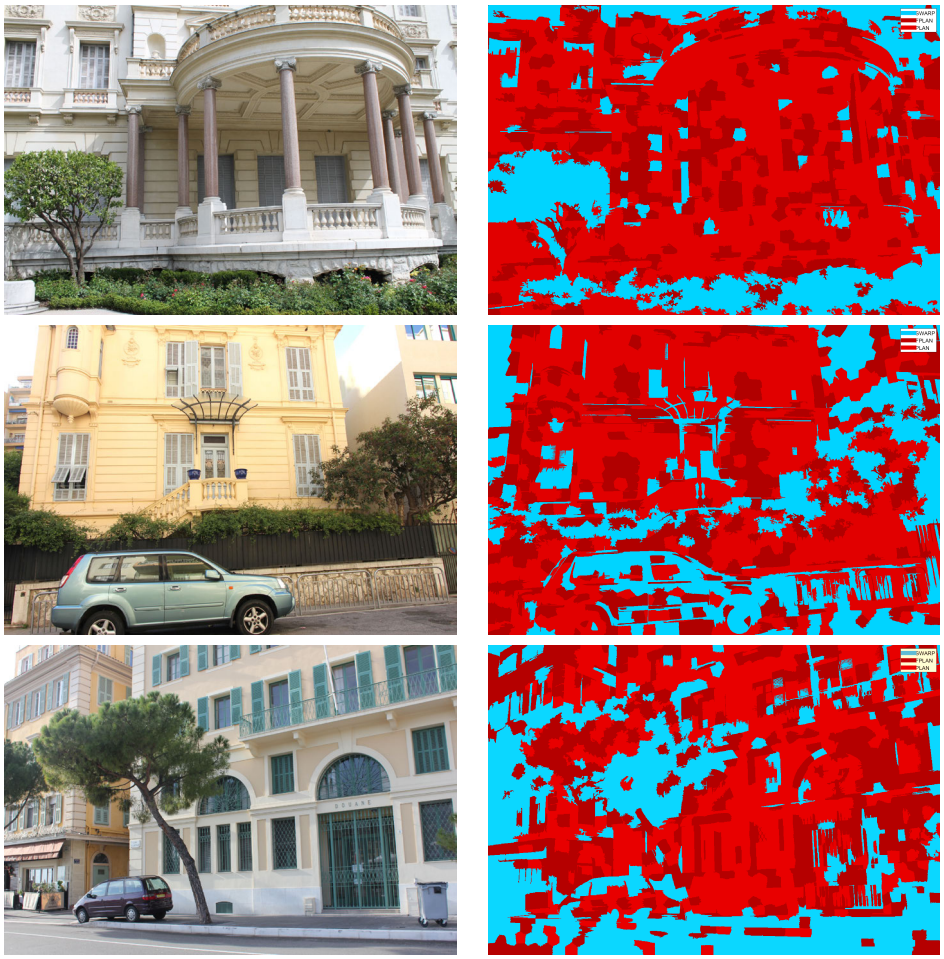


Figure 3.9: **Selection of the rendering algorithm** Superpixels in dark and medium red are rendered using planes with respectively the *FPLAN* and *PLAN* algorithm (with the same rendering time computational cost). When the *SWARP* algorithm is selected the superpixels are in blue. This last label is mainly used in regions with poor or non existing 3D information such as leaves and specular car windows.

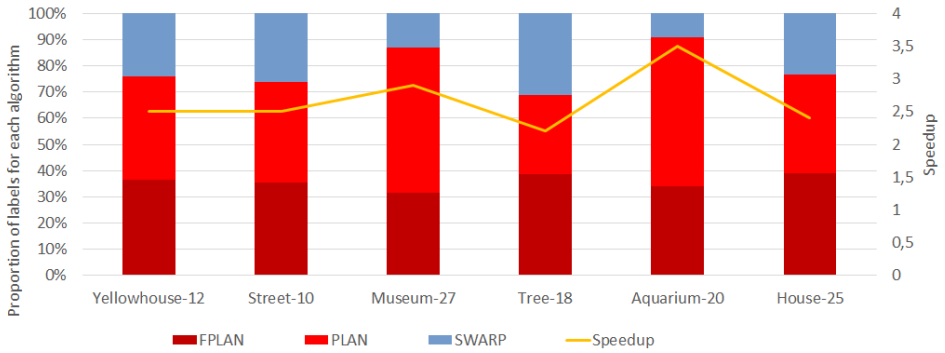



Figure 3.10: Proportion of selected algorithm for some datasets and the gain in rendering speed.

the rest of scenes (Tree-18, Bouquet_house-25, Museum_back-29, SaintAndrews-28, Fancy_restaurant-26, ChapelHill1-25, ChapelHill2-30) we used CMPMVS reconstruction (Jancosek & Pajdla, 2011). We reconstruct the scene Hotel_corner-10 with all three mentioned reconstructions (PMVS, MVE, CMPMVS) and discuss this result in Section 3.6.2. The suffix in the name of a dataset indicates the number of input photographs used. In Section 3.6.1 we present visual comparisons for all these datasets.

The main goal of our approach is to choose the most appropriate rendering process according to quality criteria. This limits the usage of expensive computations for image-space warps (Chaurasia *et al.*, 2013) to regions with poor 3D information and favors planar approximation for the superpixels where its rendering quality is high. In Fig. 3.9 we illustrate the selected IBR method for superpixels of one view for three different datasets. The planar approximations are mostly used on buildings where 3D reconstruction is most reliable. In more challenging parts of the scene, the image-space warps are more likely to be used. This is the case for leaves where geometry is not necessarily well approximated by a plane. Due to reflections, windows are also not well reconstructed and we notice a higher proportion of superpixels of *SWARP* labels selected (in blue), as well as some occlusion boundaries where it is harder to accurately fit a plane.

Table 3.1 shows the percentages of superpixels classified in *FPLAN, PLAN, SWARP* on average (± standard deviation) for each dataset. Planar approximations are used on average for 74% of superpixels allowing our algorithm to run 2.5 times faster (mean value) than *SWARP*. We ran a batch preprocess and rendering of the scene Bouquet_house-25 on a 12-core 2.5GHz Dell Z800 (NVIDIA Titan GTX GPU); all others on a 6-core Dell 3.2GHz Z420 (GTX 680). After MVS reconstruction, the whole preprocess takes about 1min/image. Warps are parallelized, explaining the difference in overhead of our approach compared to planar methods in the two configurations. At rendering time, the cost of choosing the four nearest neighbors views is negligible.

The main advantage of our method is speed, since it only uses shape-preserving warps when necessary. By appropriately selecting the rendering method to use for each region of the image, our selective IBR is on average 2.5 times faster than *SWARP*, reaching 3.5 times for the Aquarium-20 scene. In (Table 3.2) we show frames per second (FPS) for each algorithm. In Fig. 3.10 we can see how the number of segments rendered with planar

re-projection correlates with the gain in speed. This gain in speed can be critical for the usage of free-viewpoint IBR on devices with limited compute resources, such as mobile phones for example.

### 3.6.1 Qualitative Comparisons

Quality evaluation is subjective, especially for the complex imagery we consider here. In the following we present qualitative comparisons of Selective rendering against individual methods used for labeling. We also compare our approach with the three recent IBR methods (Chaurasia *et al.*, 2013; Lipski *et al.*, 2014; Flynn *et al.*, 2015). These two first approaches have already shown their superiority Chaurasia *et al.* (2013); Lipski *et al.* (2014) over methods based on optical flow estimation (Eisemann *et al.*, 2008), epipolar

| Scene | *FPLAN* | *PLAN* | *SWARP* |
|---|---|---|---|
| Yellowhouse-12 | 36.62 ± 5.84 | 39.45 ± 5.88 | 23.93 ± 7.87 |
| Street-10 | 35.30 ± 6.03 | 38,47 ± 5.12 | 26.23 ± 6.62 |
| Museum_front-27 | 31.52 ± 3.12 | 43.20 ± 17.35 | 12.98 ± 1.30 |
| Museum_back-29 | 24.79 ± 4.86 | 52.64 ± 4.88 | 22.57 ± 3.16 |
| Aquarium-20 | 34.02 ± 4.94 | 56.75 ± 5.23 | 9.23 ± 1.93 |
| Library_RM-50 | 44.27 ± 4.00 | 16.55 ± 3.47 | 39.18 ± 3.56 |
| Hotel_corner-10[1] | 29.71 ± 43.20 | 22.52 ± 3.42 | 27.09 ± 14.08 |
| Tree-18 | 33.48 ± 5.40 | 42.48 ± 5.43 | 24.04 ± 1.55 |
| Bouquet_house-25 | 38.87 ± 5.47 | 37.65 ± 5.01 | 23.48 ± 5.99 |
| SaintAndrews-28 | 45.40 ± 4.47 | 30.85 ± 4.53 | 23.75 ± 1.43 |
| ChapelHill1-25 | 29.97 ± 3.20 | 42.39 ± 3.31 | 27.64 ± 1.75 |
| ChapelHill-30 | 32.450 ± 1.60 | 41.83 ± 1.80 | 25.67 ± 1.32 |
| Fancy_restaurant-26 | 47.01 ± 11.42 | 30.96 ± 8.41 | 22.03 ± 4.70 |
| **Average** | **35.65** | **40.69** | **23.66** |

Table 3.1: **Average (± standard deviation) percent of the labels of Bayesian preprocessing phase.** The percentage of superpixels requiring a warp is low on average.

| Scene | Speedup | *Selective* | *SWARP* | *F/PLAN* |
|---|---|---|---|---|
| Yellowhouse-12 | 2.5 | 145.7 | 58.7 | 346.0 |
| Street-10 | 2.5 | 158.5 | 62.5 | 373.5 |
| Museum-27 | 2.9 | 158.3 | 55.3 | 319.3 |
| Tree-18 | 2.2 | 136.5 | 62.5 | 418.3 |
| Aquarium-20 | 3.5 | 218.0 | 62.5 | 314.3 |
| House-25 | 2.4 | 97.0 | 41 | 102 |
| **Average** | **2.67** | **152.33** | **57.08** | **312.23** |

Table 3.2: **FPS for each algorithm and our selective approach.** The speed up factor is relative to the *SWARP* method.

constraints (Goesele *et al.*, 2010) or manually defined silhouettes (Chaurasia *et al.*, 2011).

Figure 3.11: **Rendering close-up of the scene Bouquet_house-25.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: proposed. From top to bottom, the scenes Bouquet_house-25.

From visual inspection of interactive sessions, using different navigation paths for the various datasets, our approach globally outperforms the other superpixel based algorithms. To illustrate this, Fig. 3.11 to Fig. 3.16 show a selection of challenging viewpoints, off the view-interpolation trajectory. Each time the proposed selective approach results in rendering quality equivalent or better than the three methods taken separately. This is more obvious in a continious navigation path than in single photography. We provide a video in supplementary material available here: `http://team.inria.fr/graphdeco/publications`. In this video, artifacts (e.g., incorrect plane reconstruction) become particularly visible during camera motion. The choice of *SWARP* for unreconstructed regions results in improved overall visual quality compared to *PLAN, FPLAN* albeit with an increase in computational overhead, depending on the CPU used.

In Figs. 3.17 to 3.19, we compare selective rendering only with *SWARP* (Chaurasia *et al.*, 2013). For each scene we present the spatially closest reference camera to have in idea of how the scene should look like. For these extreme cases, we can see on the top view that the novel camera positioned and oriented much farther than previous examples. In general, we observed that Selective and SWARP present equivalent results.

In Fig. 3.20, we show a visual comparison with the dense correspondence approach of Lipski *et al.* (2014), notably the rendered image for a given position on the view interpolation path. Overall visual quality is close, although our methods avoids some

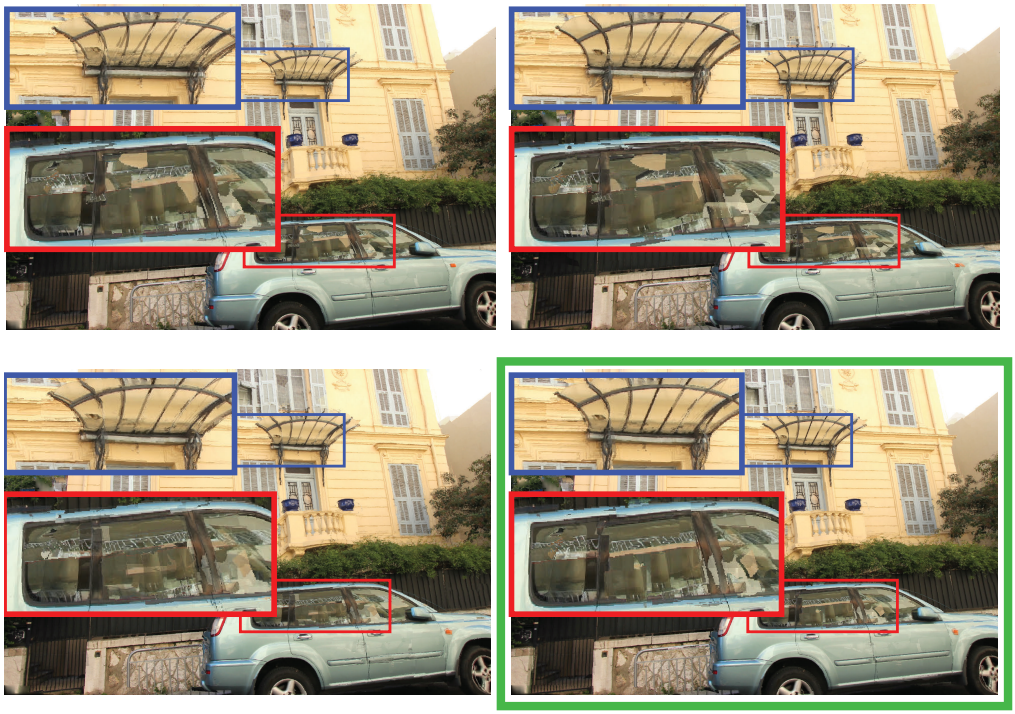


Figure 3.12: **Rendering close-up for the scene Yellowhouse-12.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: our result.

of the blurring due to correspondence tracking and interpolation. We note however that the rendering based on dense correspondences (left image) has the typical artifacts due to a bad estimation of correspondences (see close up on the tree). In regions with poor 3D information (building of the left) both methods show some artifacts as well as in thin structures as the flag pole.

In Fig. 3.21, we compare with the most recent paper for IBR walkthroughs (Lipski *et al.*, 2014) that uses pre-trained deep networks. Their offline synthesis takes several minutes to render this low resolution image. Other artifacts that characterize their system, include the vanish of thin foreground structures. Also, partially occluded objects tend to appear overblurred. However, they manage to obtain a sharper region on the flag (upper left) which in our case, presents ghosting because during the capture it was waving.

### 3.6.2 Comparison using Different Reconstructions

To test the effect of the reconstruction methods on our labeling, we used three different multi-view reconstructions algorithms publicly available on Hotel_corner-10: PMVS (Furukawa & Ponce, 2007), MVE (Goesele *et al.*, 2007) and CMPMVS (Jancosek & Pajdla, 2011). We have previously shown reconstructed point cloud for PMVS and recovered meshes MVE and CMPMVS in Section 2.2. The average percentage of superpixels labeled as SWARP is about 50%, 37% and 27% for PMVS, MVE and CMPMVS respectively. This result was expected considering the improvement in

Figure 3.13: **Rendering close-up for the scene Street-10.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: our result.

completeness and quality of reconstruction from PMVS to CMPMVS and shows that our approach has the very desirable property of improving speed as the quality of the reconstruction improves.

Consider the point of view presented in Fig. 3.22. We can observed that for this point of view, CMPMVS clearly offers better rendering results with a more complete mesh even in regions of vegetation and specularities. With MVE and PMVS we observed in Fig. 3.22(c) and Fig. 3.22(d) respectively, different kind of artifacts inherent to the kind of explicit 3D information provided by the reconstruction. For example, PMVS provides a point cloud with visibility information per 3D sample. Visibility information helps to decide which segments do not contain depth samples and so we must synthesize their depth. We see on the left side of Fig. 3.22(d) visual errors in the slanted wall and the branches of the cactus because of depth approximations. On the other hand, for MVE provides a mesh with no visibility information. Regions that where not initially reconstructed, like the branches of the cactus, might take the depth from what is reconstructed behind. As a result, we obtain ghosting when this vegetation projects into wrong places.

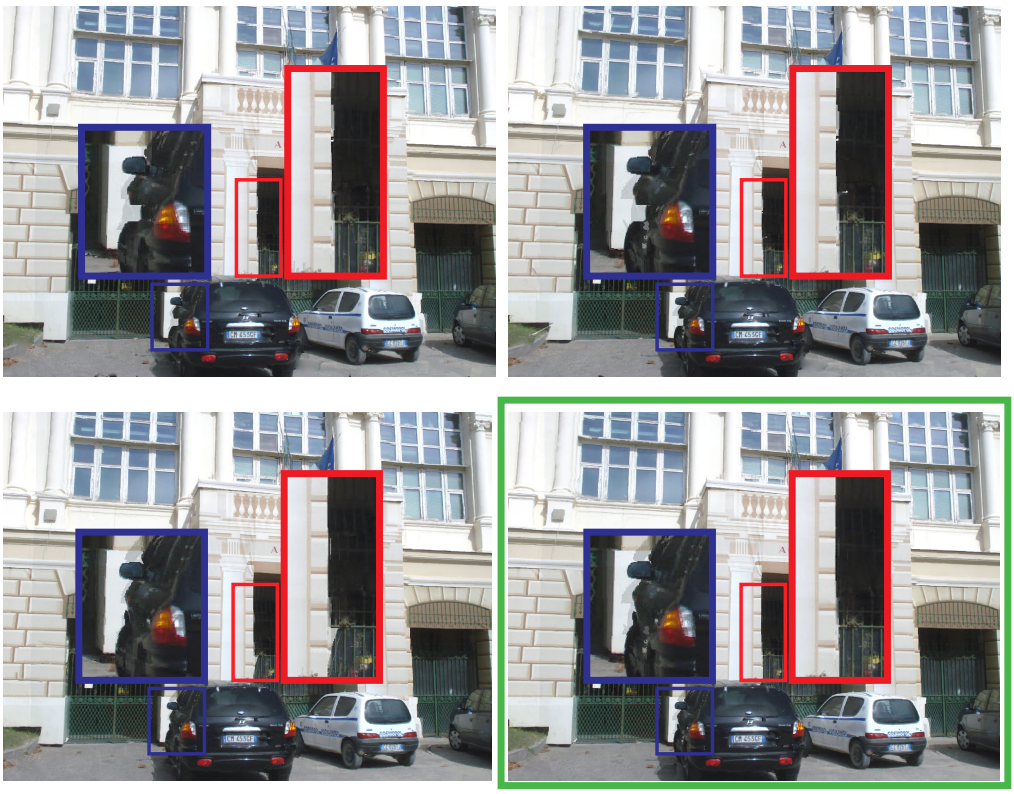


Figure 3.14: **Rendering close-up for the scene Aquarium-20.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: our result.

### 3.6.3 Quantitative Evaluation

To validate our results quantitatively, we investigated the use of visual metrics for image quality assessment like *Structure Similarity Index* SSIM (Wang *et al.*, 2004) or *Visual Difference Predictor* VDP (Mantiuk *et al.*, 2011). These metrics need a reference image to compare against. Hence the procedure followed was *leave-one-out*: We calibrate all input photographs but we set apart a subset of inputs to use them as reference images for quality metrics. MVS reconstruction and subsequent steps were performed only in the subset of input images. For test algorithms (*PLAN, SWARP* and Selective), rendered images were generated at calibrated cameras of the reference images. In Table 3.3 we set some results (the higher the number, the better the quality) which are actually inconclusive since according to VDP, we are not significantly better and even in some cases, our score is lower.

To understand these results, we show the probability maps of error detection for SSIM in Fig. 3.23(c) and VDP in Fig. 3.23(d) for the synthesized view in Fig. 3.23(b) given the reference image Fig. 3.23(a). Higher error is encoded as a color closer to red. We observe that the error is high and localized around the windows, at the flag and all the vegetation. However, in terms of subjective visual quality, the human eye has difficulty perceiving small errors in reflections or in cluttered vegetation (Reinhard *et al.*, 2012). Given the

Figure 3.15: **Rendering close-up for the scene Tree-18.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: our result.

bias of high error, it is difficult to differentiate the quality of different algorithms. For more rare cases the error is slightly worse due the difference in color and other aspects. We can see mainly red everywhere in Fig. 3.23(e) and Fig. 3.23(f) which biases the result of the metric, to the point that the numbers presented in Table 1.3 are essentially meaningless. The weakness of these metrics has been highlighted by Čadík *et al.* (2012) while evaluating the performance of state-of-the-art metric for detection and localization of rendering distortions. As a result, quantitative assets Image-based Rendering methods in a reliable way is still an open research topic.

| Scene | *Selective* | *SWARP* | *PLAN* |
|---|---|---|---|
| Yellowhouse-12 | 65.44 | 66.14 | 64.96 |
| Street-10 | 65.78 | 65.77 | 68.8 |
| Museum-27 | 65.96 | 66.17 | 66.41 |
| Tree-18 | 66.10 | 65.39 | 65.17 |
| Aquarium-20 | 60.26 | 60.24 | 60.22 |
| House-25 | 66.10 | 65.39 | 65.17 |

Table 3.3: **Leave-one-out quantitative evaluation with VDP (Mantiuk *et al.*, 2011).** Results are inconclusive for this metric.

Figure 3.16: **Rendering close-up for the scene Library_RM-50.** Top left: FPLAN. Top right: PLAN. Bottom left: SWARP. Bottom right: our result.

## 3.7 Conclusions and Discussions

We proposed a Bayesian formulation to model the choice of the most suitable rendering method for IBR algorithms based on superpixels, using probability distributions to model rendering quality and choice of rendering method. We solve for the most suitable rendering method using MAP estimation, which chooses a rendering method for each superpixel in a preprocess. We use the result to define a selective IBR algorithm combining the benefits of previous algorithms, with very good overall speed/quality tradeoff. One important strength of our approach is that it identifies regions of the image where using a more expensive IBR approach (e.g., Chaurasia *et al.* 2013) is wasteful, and replaces it with a cheaper planar reprojection method of equivalent quality while another important aspect is that as the 3D reconstruction improves, the speedup offered by our approach is higher.

We currently use the camera selection and blending of Chaurasia *et al.*. These can definitely be improved, but both topics are hard problems involving different tradeoffs which we will investigate in future work. A good solution will improve the quality of our algorithm significantly. The recent rendering method for indoors methods (Hedman *et al.*, 2016) is an interesting avenue to explore in this direction.

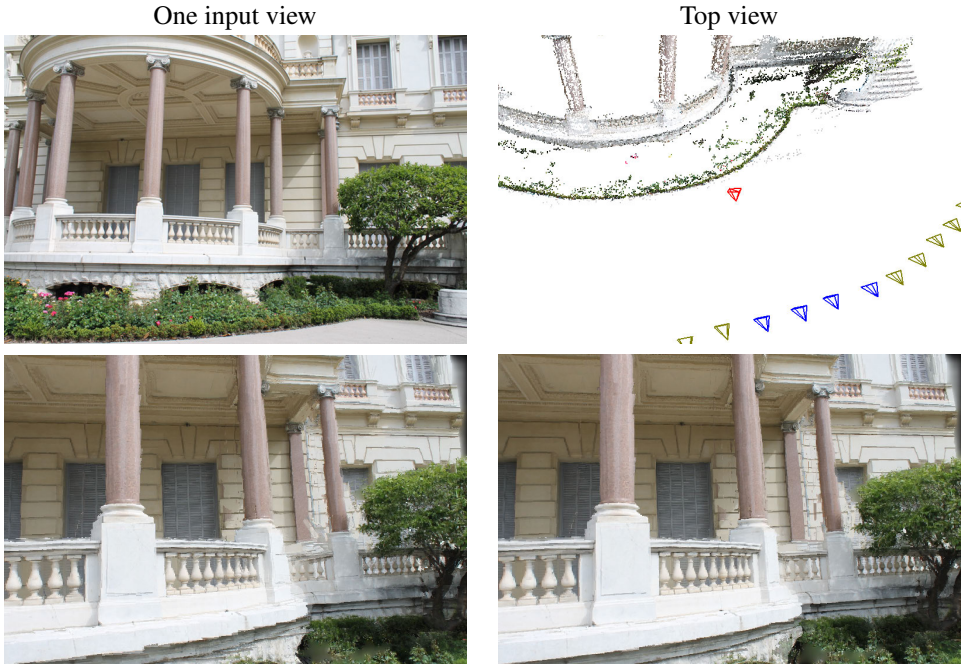
This work provides a first indication on the utility and power of MAP estimation as a preprocess for real-time IBR. An interesting future direction is to pursue these ideas further in a more general context taking the prior $p(\xi)$ into account, improving the rendering methods and their parameters. Developing such solutions raises several hard

challenges, including a way to estimate quality of IBR in the absence of a reference and preferably online. Another important issue is the balance between preprocessing and runtime: optimization per pixel at rendering time is prohibitively expensive, but some combination of preprocessing and well-designed GPU data structures could result in significant improvements in rendering quality using an extension of our Bayesian approach.

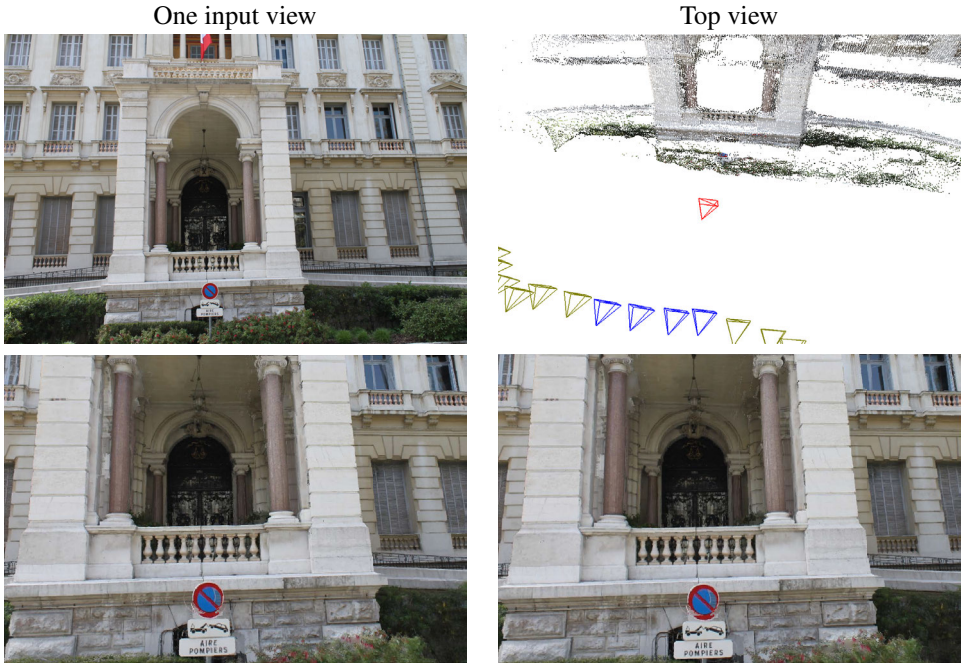
This thesis was funded by the *CR-PLAY*[2] project, led by a video-game company[3]. Our selective IBR algorithm has been implemented in *Unity* engine framework which makes integration with prototypes very easy. Thanks to the reduction in rendering time and memory footprint, our algorithm has been deployed on mobile devices such as the Google Tango (see Fig. 6.1) and used for video game prototypes like *Silver-Arrow*, *IBR-Basketball* and others. We will discuss this in more detail in Chapter 6.

[2] cr-play.eu

[3] *Testaluna*: http://www.testaluna.it

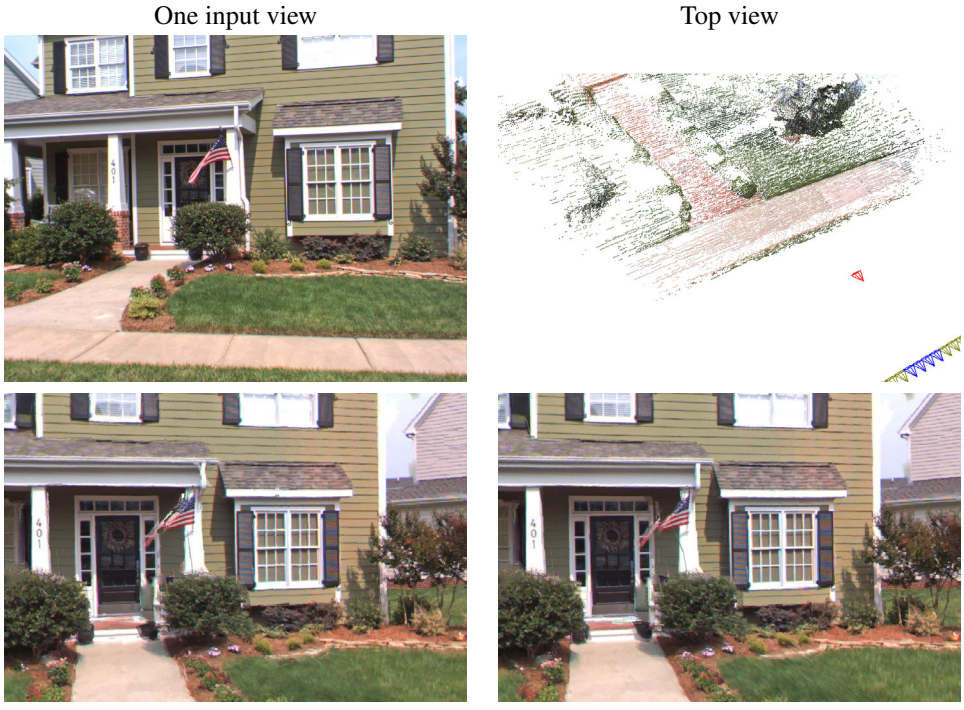



(a) Museum_front-27




(b) Museum_back-29

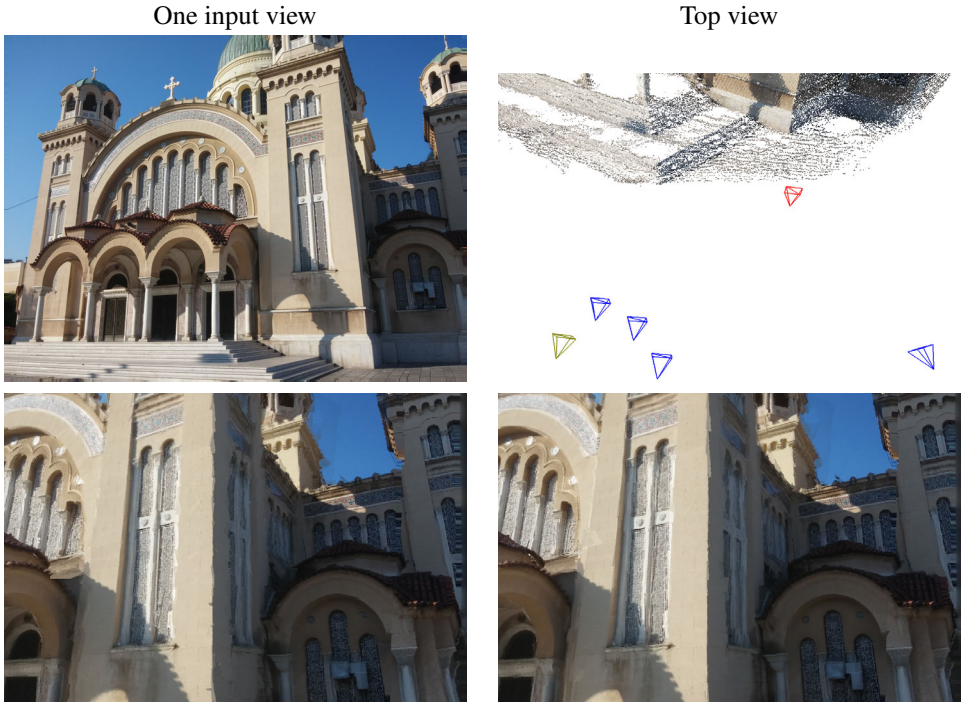
Figure 3.17: Bottom row: novel views rendered far from reference images with *SWARP* (left) and Selection (right). Top row: a reference view (left) and top view (right). Position and orientation of the novel camera is represented by the red pyramid in the top right view.



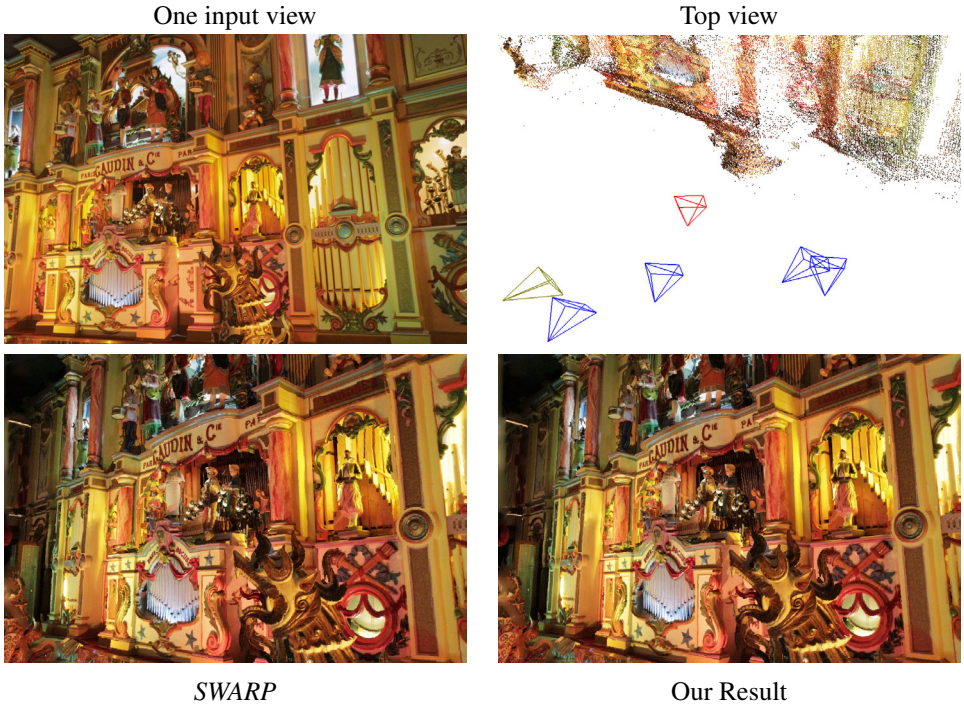

(a) ChapelHill1-25


(b) ChapelHill3-30

Figure 3.18: Bottom row: novel views rendered far from reference images with *SWARP* (left) and Selection (right). Top row: a reference view (left) and top view (right). Position and orientation of the novel camera is represented by the red pyramid in the upper right.




(a) SaintAndrews-28




(b) Fancy_renstaurant-26

Figure 3.19: Bottom row: novel views rendered far from reference images with *SWARP* (left) and Selection (right). Top row: a reference view (left) and top view (right). Position and orientation of the novel camera is represented by the red pyramid in the upper right.

(a) Lipski *et al.*

(b) Ours

Figure 3.20: **Comparison of *Dense correspondences and DIBR* (Lipski *et al.*, 2014) (on the left) and our Selective Rendering method (on the right).** For a given position on the view interpolation path. We note that the rendering based on dense correspondences has the typical artifacts due to a bad estimation of correspondences (see close ups). In regions with poor 3D information (building of the left) both methods show rendering artifacts.

(a) Flynn *et al.*

(b) Ours

Figure 3.21: **Comparison of *DeepStereo* (Flynn *et al.*, 2015) (on the left) and our Selective Rendering method (on the right).** Views were synthesized with parameters of a camera left out for rendering. Flynn *et al.*'s network took about 12 minutes on a multi-core workstation to render this 512x512 pixel image. In our case, our system renders this image in real-time at 1200x800 resolution.

(a) (b)

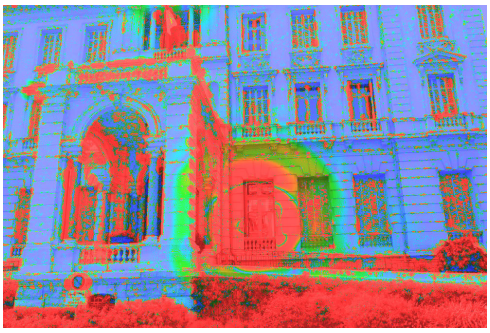
(c) (d)

Figure 3.22: **Rendered images with three different reconstructions.** (a) In red, the point of view from where the images were rendered. (b) Rendering with selective algorithm with CMPMVS reconstruction. (c) Rendering with selective algorithm with MVE reconstruction. (d) Rendering with selective algorithm with PMVS reconstruction.

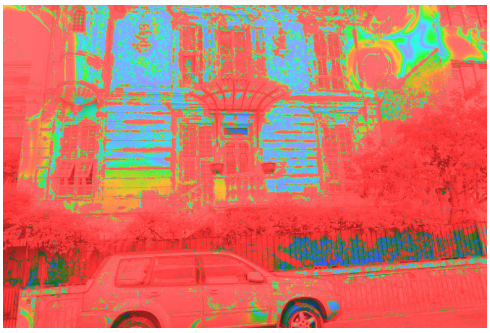
(a) Reference image

(b) Selection

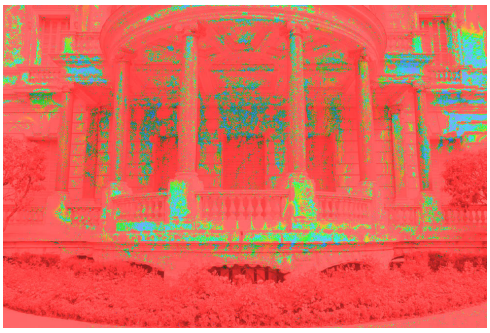
(c) SSIM local error values



(d) VDP probability detection map for (b)



(e) Prob. Map for Yellowhose-12



(f) Prob. Map for Museum_front-27

Figure 3.23: **Maps of Visual Metrics for Image Quality.** (a) Reference image. With the calibrated camera from the point of view of (a) we render the image in (b) with our approach. (c) Structural Similarity Index for the image in (b) with respect to the reference image in (a). (d) VDP detection map for the image in (b) with respect to the reference image in (a). (e) VDP detection map for a view in Yellowhouse-12. (f) VDP detection map for a view in Museum_front-27.

CHAPTER 4

# Automatic 3D Model Alignment for Mixed Image-Based Rendering

## Contents

In the previous chapter we presented a new method providing a good trade-off between speed and quality. We achieved this by choosing the best IBR algorithm to render a given local region. Our approach provides satisfactory results, but does not inherently improve the *quality* of the rendering, since we are limited by the capabilities of available algorithms. The algorithms we use suffer from artifacts on poorly reconstructed objects, e.g., reflective surfaces such as cars. To alleviate this problem, we propose a method that automatically identifies stock 3D models, aligns them in the 3D scene and performs morphing to better capture image contours. Our method provide models which are well-aligned in 3D and to contours in all the images of the multi-view dataset, allowing us to use the resulting model in our mixed IBR algorithm. As we will see in the results section of this chapter, our method shows significant improvement in image quality for free-viewpoint IBR, especially when moving far from the captured viewpoints.

## 4.1 Introduction

A key element of a high quality IBR is good 3D reconstruction estimated from the images. While great progress has been made in this domain, these methods do not work well in the case of transparent surfaces or reflective objects. IBR methods try to compensate for missing 3D geometry using strategies like fronto-parallel assumptions (Zitnick & Kang, 2007), 2D image warps (Chaurasia *et al.*, 2013) or interpolation from image correspondences (Lipski *et al.*, 2015). For poorly reconstructed foreground objects, close to the novel, synthesized view, this is usually not sufficient to mask errors in the reconstruction. A typical example of incomplete reconstruction of foregrounds in outdoors scenes are cars. (see Fig. 4.1).

We focus on urban environments, where captured scenes contain buildings and many man-made objects (cars, signposts, benches etc.). With the recent development of 3D model databases it is more and more likely to find a corresponding model to objects in a captured scene. Examples of these databases are *Trimble Warehouse 3D*[1], *ShapeNet*[2] (Chang *et al.*, 2015), and *ObjectNet3D*[3] (Xiang *et al.*, 2016). Trying to use CAD models is a promising strategy that has been used for various application such as depth correction (Lee *et al.*, 2015) or image editing and manipulation (Kholgade *et al.*, 2014), even though selected 3D models usually do not exactly correspond to the images, requiring deformation of the model. However, these previous methods rely entirely on user interaction for selection, placement, alignment and deformation of the model and often are not designed to handle multi-view data which is required for IBR.

In this chapter, we present an *automatic* method which leverages databases of 3D CAD models to augment IBR scenes and allow better navigation. The core idea is to use the stock models as a better proxy for the objects in the scene. To be visually convincing the stock models first needs to be correctly placed in the scene and carefully aligned with the silhouettes in input photographs. We then use the model in our mixed IBR algorithm that

[1] https://3dwarehouse.sketchup.com
[2] http://shapenet.cs.stanford.edu
[3] http://cvgl.stanford.edu/projects/objectnet3d

Figure 4.1: **MVS reconstruction of specular objects of an urban scene.** Left: input view. Right: reconstructed scene.

renders the background and the objects in two passes, blending them in a final pass. Our new approach which builds on learning methods in a preprocess and proposes an improved contour-based alignment and morphing approach to *automatically* chose, align and morph 3D models in a reconstructed scene for IBR. To the best of our knowledge this is the first method automatic method for this process to improve IBR.

Our contributions can be summarized as follows:

- We first adapt learning-based methods to detect and identify an object class and an object pose in the input views.
- We present an accurate object alignment and correspondence selection method for morphing with more fine-grain and accurate treatment. We then propose a method which exploits all available information, namely partial and inaccurate 3D reconstruction, multi-view calibration, image contours and the 3D model to achieve accurate object alignment for morphing.

Our method provides fine-grain alignment and automatic correspondence detection for contours, achieving a good initial placement in the scene. Thanks to the good initial placement and the correspondence detection, we can automatically morph the stock-model to better align with contours in all the images of the multi-view dataset. The resulting model is then used in our mixed IBR algorithm, greatly improving image quality, especially

when moving far from the captured viewpoints thanks to the stability given by an explicit geometric representation of objects nearby the cameras.

Our approach is fully automatic and can directly benefit from any future improvement in model selection or larger databases. We demonstrate our method on the example of cars in urban environments, since very often instances of this class are present in street-view datasets and MVS algorithms struggle to recover their geometry. Another good reason to focus on cars, is that the number of models available is sufficiently large 3D model database (Chang *et al.*, 2015); when databases of models for other objects become available, our approach can be directly applied. Our experiments demonstrate a clear improvement in rendering quality compared to state of the art methods.

Before presenting an overview (Section 4.3) and details of our system, in Section 4.2 we discuss related work which is specific to the method developed in this chapter, which complements the general review of IBR presented in Chapter 2.

## 4.2 Related Work

We propose a new mixed IBR algorithm, improving rendering quality of objects by automatically retrieving, aligning and morphing 3D geometry from stock models. Since each of these components is a vast area in itself, we restrict discussion of previous work only to the most closely related methods. In contrast to previous IBR methods, we provide an end-to-end automatic pipeline which finds a good match for such objects from stock 3D models, and then performs fine-grain alignment and morphing to provide *multi-view* consistent 3D model for the object, resulting in much higher visual quality for free-viewpoint IBR. In the Following, we review some methods for learning with 3D object databases (in Section 4.2.1) and geometry alignment (in Section 4.2.2).

### 4.2.1 3D Object Databases and Learning

Initial methods to align 3D models to photos for model retrieval, extract edges from both the 3D model and the photograph (Roberts, 1965; Huttenlocher & Ullman, 1987; Lowe, 1987). Such approaches work well especially for untextured objects (Lim *et al.*, 2013; Arandjelović & Zisserman, 2011), but can be limited by the difficulty of extracting reliable and consistent edges in 2D and 3D. The most popular approach for matching CAD models to photographs is to use handcrafted descriptors such as HOGs (Dalal & Triggs, 2005; Su *et al.*, 2014; Aubry *et al.*, 2014b) or learned Convolutional Neural Network (CNN) features (Massa *et al.*, 2015; Aubry & Russell, 2015). Hueting *et al.* (2015) use both HOGs and CNN features to co-align and sort collections of class-labeled images and CAD models. Considering the recent success of CNN features, we follow this option.

CNNs (LeCun *et al.*, 1998) are composed of a succession of simple operations, such as convolutions, non-linearities and max poolings, whose parameters are optimized to perform a given task. They have demonstrated impressive results in many domains such as image classification (Krizhevsky *et al.*, 2012a; Simonyan & Zisserman, 2014) and object detection (Girshick *et al.*, 2014; Gidaris & Komodakis, 2015). In addition, they provide intermediate features which have proven to be generic enough to be re-used or adapted for very different

tasks (Yosinski *et al.*, 2014; Girshick *et al.*, 2014). Here we use CNNs to first detect object categories using the framework of Gidaris & Komodakis (2015) and then simply match images and rendered views from the 3D models using the intermediary *pool4* features of the network of Simonyan & Zisserman (2014) to retreive the corresponding 3D model in a way similar to Aubry & Russell (2015). As mentioned before, here we focus only on cars to validate our approach since this is the best database available for objects in the scenes we target.

We next review previous methods for car pose estimation. Koller (1993) proposed a method based on a polyhedral 3D vehicle model which was further improved by Ferryman *et al.* (1995). The use of more complex deformable models has become more common (Leotta & Mundy, 2011). Hödlmoser *et al.* (2013) applied Random Forests (using handcrafted features) to track cars in videos. More recently, fine-grained model classification has received more attention (Lin *et al.*, 2014). By taking advantage of 2D detections from deformable part models, a 3D model is deformed to better fit the estimated landmark positions. Part based features are then used for fine grained model classification. Finally, Mottaghi *et al.* (2015) use a coarse-to-fine hierarchical representation for object detection, pose estimation and sub-category recognition. This method is not limited to cars and results are demonstrated with planes and boats. These car pose estimation methods focus on the single-view case. The inherent 3D ambiguity for our multi-view setting makes these methods suboptimal for our goal.

Model selection and pose estimation are the first component of our solution; To be as generic as possible we build on the CNN-based methods and extend it to handle the multi-view data we treat here.

### 4.2.2 Geometry Alignment

A first category of methods treat automatic 3D model alignment. With the goal of a more compact geometric representations, Lafarge *et al.* (2010) attempted to insert basic primitives in the reconstructed geometry while preserving details. In Lafarge *et al.* (2013) this insertion was jointly done with multi-view reconstruction. In their multi-object energy model a photo-consistency term would favor the mesh alignment with content of photographs, however some objects would not be correctly represented by piece wise primitives. Other solutions (Rosenhahn *et al.*, 2007; Dambreville *et al.*, 2008b; Prisacariu & Reid, 2012) were based on a level set formulation and assumed a fixed 3D model and assume a separation between the color models of the background and the object of interest. To handle different types of 3D models, a common approach is to learn an embedding into a lower dimensional space using kernel principal component analysis (Dambreville *et al.*, 2008a) or Gaussian process latent variable models (Prisacariu & Reid, 2011). Recent methods take advantages of these dimensionality reduction approaches to estimate 3D shape, 2D-3D pose and image segmentation (Sandhu *et al.*, 2011; Prisacariu *et al.*, 2012). Although discussed in Prisacariu & Reid, results on multi-view data are not demonstrated, and in their examples images have uniform color background which is not the case of cluttered urban scenes. Relying on dimensionality reduction, the unique parts present in a few models may disappear from the generic model. Our solution takes advantage of the

diversity of models in the dataset and finds the closest one. It can also directly benefit from any improvement in matching or any new instance added to the database.

In Computer Graphics applications, 3D models are fitted to images for various applications such as image edition (Prasad & Fitzgibbon, 2006) or 3D modeling (Xu *et al.*, 2011). In all cases user input is required for 3D geometry alignment. Zheng *et al.* (2012) rely on user interaction to help approximate underlying geometry with cuboids. A similar approach is adopted in Chen *et al.* (2013) but the user has access to richer 3D components. In Kholgade *et al.* (2014), the user selects a similar 3D model which is deformed to fit the image according to user provided constraints. The main advantage of this method is its ability to generate novel viewpoints of the object. The closest method related to our work is the depth estimation method using 3D models by (Lee *et al.*, 2015). Here the objective is to help the 2D-3D conversion for images and videos. Using a model aligned with the input image, depth is coherently in-painted. However the process relies heavily on manual user interaction at all stages: model selection, initial constraints for pose estimation and image segmentation for morphing.

In contrast, our approach provides a fully automatic pipeline for all these steps, leading to a solution that is scalable and thus usable for IBR.

## 4.3 Overview

The goal of our approach is a high-quality mixed Image-Based Rendering algorithm for urban scenes, by using stock 3D models to represent hard-to-reconstruct geometry such as cars. To achieve this, we build on recent advances in object detection/recognition and the ever growing 3D object databases. To avoid confusion we refer to the 3D mesh from the shape database as the *3D object mesh* and any geometry coming from reconstruction as the *reconstructed geometry*.

**Input.** The input to our approach is a set of photographs of the scene. As in the previous chapter, we obtain calibrated cameras and an approximate geometry of the scene (proxy) using structure from motion (*VisualSfM* (Wu *et al.*, 2011; Wu, 2013) and multi-view stereo reconstruction CMPMVS (Jancosek & Pajdla, 2011); other alternatives could be used for this step. We use the *ShapeNet 3D* object database (Chang *et al.*, 2015) for the object retrieval step.

**Object Selection and Preprocessing.** Object bounding boxes are obtained using a recent detection algorithm (Gidaris & Komodakis, 2015). We use the 3D reconstructed geometry to put the detections into correspondence ( Fig. 4.2.a). These images are used to find the corresponding 3D model and its orientation with respect to the cameras (Fig. 4.2.c).

**Object Alignment.** After obtaining the 3D models, we place the object in the scene. We use a multi-view approach taking advantage of the detection bounding boxes (Section 4.5.1), the available geometry and silhouette matching (Section 4.5.2).

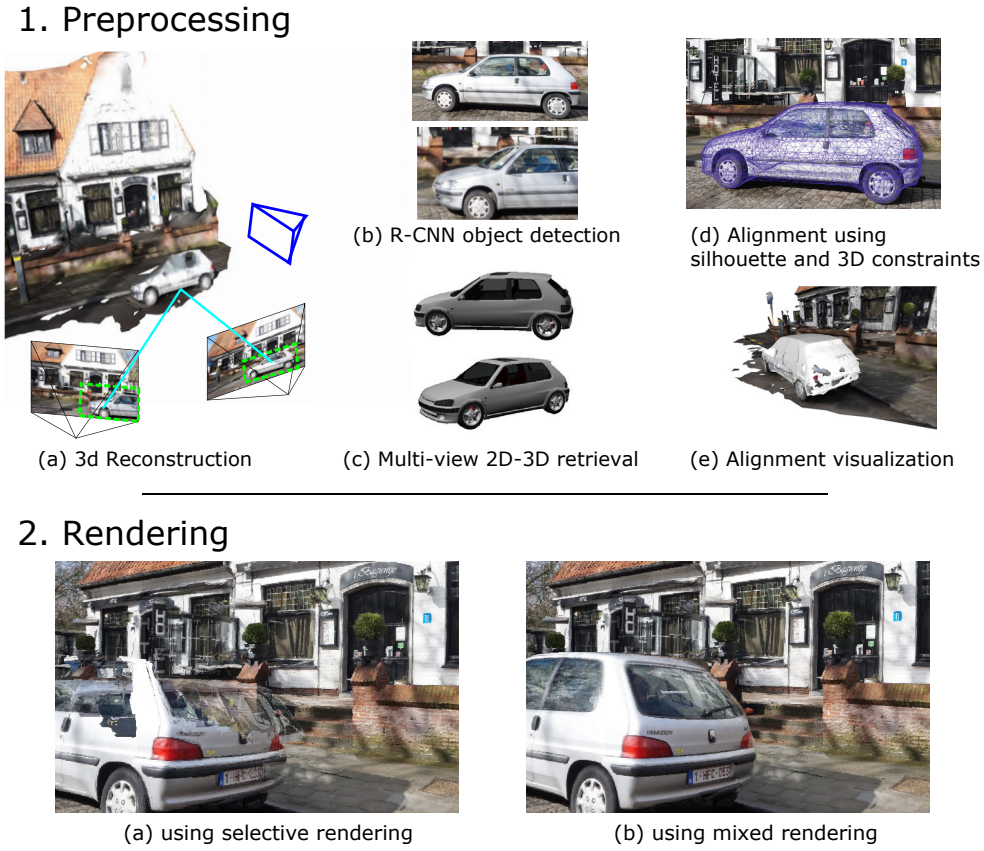



Figure 4.2: **Overview.** We propose a new method for Image-Based Rendering that takes advantage of 3D model databases. During a fully automatic *Preprocessing* stage, (a) cameras are calibrated and 3D reconstruction is estimated using multi-view stereo (Jancosek & Pajdla, 2011). Objects of interest (here cars) are detected using R-CNN (Gidaris & Komodakis, 2015). These detections are matched using viewing rays in cyan. (b) The corresponding images are used to query our database of 3D objects, (c) which returns a 3D model and an orientation relative to the cameras. (1.d) combining depth and silhouette cues in multi-view, (e) a better alignment of the object is obtained. During *Rendering*, we generate the novel viewpoints illustrated by the blue camera (a). The selective approach of the previous chapter (Ortiz-Cayon *et al.*, 2015) exhibits strong artifacts on the car (2.a). Using our current approach (2.b) we achieve good rendering quality for both the car and the background.

**Object Morphing and Rendering.** After the alignment, the object mesh can still be different from the actual object geometry. In this case, we use morphing based on silhouettes to obtain a closer fit. Rendering is achieved by compositing background rendering using our selective rendering (Ortiz-Cayon *et al.*, 2015) and an ULR-like (Buehler *et al.*, 2001) rendering for the foreground. Results clearly demonstrate the advantage of our approach (Fig. 4.2.2.b).

## 4.4 Stock 3D Model from Multi-view Images

The first step of our system is to identify the regions in the input images containing the objects of interest, then select the stock 3D model and finally create a model which is suitable for further processing.

### 4.4.1 Multi-View Object Class Detection

We use the multi-region R-CNN (Gidaris & Komodakis, 2015) – one of the top-performing detection algorithms. The original method treats each input image of the dataset independently and produces candidate bounding boxes corresponding to the objects requested (e.g., car, aeroplane, chair, etc.).

For each input view, we run the R-CNN, which provides a set of candidate 2D bounding boxes (Fig. 4.3). To put these bounding boxes into correspondence we rely both on appearance and geometry. We first cluster candidate regions based on appearance. Candidates belonging to different color clusters should never be matched. For each pair of object candidates – from different views – we compute the intersection of the viewing line passing through the center of the 2D bounding boxes (Fig. 4.2.a). The largest 3D-point clusters identify the objects in the scene and match 2D detections. After this step we have a set of candidate objects with corresponding images from different viewpoints. Next we use this data to find the corresponding stock models.

Figure 4.3: **Object detection.** Using the region aware detection algorithm (Gidaris & Komodakis, 2015) we obtain tight 2D bounding boxes of the objects of interest in the scene.

### 4.4.2 Multi-View 2D-3D Retrieval

We use the *ShapeNet* (Chang *et al.*, 2015) database to find stock models using the 2D bounding boxes from all views. Currently, *ShapeNet* has a rich collection of the class "car" which we use to validate our approach. We downloaded approximately 5K car models from this database and for each 3D model we rendered the object from 108 viewpoints of the viewing sphere, with azimuth and elevation increasing 10 degrees in the range of $[0, 360)$ and $[0, 30)$ respectively (see Fig. 4.4). This constitutes our database of 5K car models, each associated to 108 views of the object.

We will now use the images obtained from the bounding boxes to query our object database. Following Massa *et al.* (2015), we compare the images using the cosine distance on *pool4* features. We tested features of *AlexNet* (Krizhevsky *et al.*, 2012a), VGG

Figure 4.4: **Example of Renderings**. Four out of 108 views generated for one car model of the *ShapeNet* database.

(Simonyan & Zisserman, 2014) as well as their adapted version using the framework of Massa *et al.*, which aims at bridging the domain gap between rendered 3D models and photographs. As expected, we found that the retrieval using *AlexNet* features were of lower quality than using the more powerful VGG features. More surprisingly, we also found that the retrievals with adapted features were of lower quality, probably because the adaptation layer of Massa *et al.* also destroys some image information.

Each query is matched with an image from the database. This gives both a 3D model ID and an orientation with a matching score (of the object in the scene). Orientation is expressed as azimuth-elevation angles $(\theta, \phi)$ with respect to a camera viewing the object at the origin. Because we have access to several views of the same car, we were able to further refine the retrieval using this information. Interpreting the comparison score between CNN features as a log probability, we compute a single score for each 3D model by simply summing the maximum score for this model for each of the unoccluded view of the model. Typical results of this step are shown in Fig. 4.5.

Figure 4.5: **Model and orientation matching.** The detection algorithm provides 2D bounding boxes for the object. After cropping, the images are used to query the database. Right: the matched rendering which provides both a 3D model and a rough orientation.

## 4.5 Positioning the Mesh

We now have a mesh corresponding to each object identified in the images of our multi-view dataset. The rendering with the highest matching score gives the orientation of the model with respect to one camera. To place the mesh in the scene we follow a multi-view strategy taking advantage of the detection information from all the views. To achieve more precise alignment we use both reconstructed geometry and silhouette matching. The final output is a set of rigid transformation parameters $\Lambda$ corresponding to scale, translation and rotation for each 3D model.

### 4.5.1 Initial Pose Estimation

Initial pose of each mesh is computed as follows. We perform the initial alignment separately for rotation and then translation and scale. To orient the 3D model obtained from the previous step (Section 4.4.2), we rely on the matched image with the highest score from the database. We know the orientation of the 3D models with respect to the corresponding virtual camera $C_r$, which we represent as rotation $R_r$. This rotation is not sufficient to align the 3D model, since our database consists of images rendered with the object at the center (Fig. 4.4) while in real world images the object can be present at different locations (Fig. 4.3). To compensate for this, we compute the rotation matrix that aligns the camera central axis ray with the viewing line passing through the center of the detected bounding box. We compose all rotations and represent the orientation of the mesh in a global coordinate system.

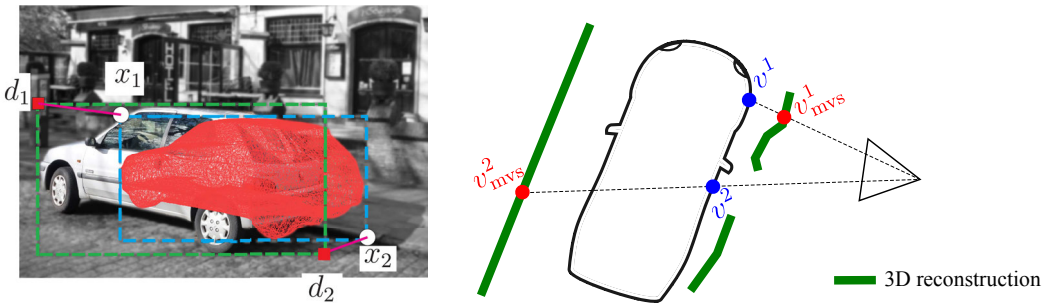
After setting a rough orientation, a first estimate of the car position is computed as the point that minimizes the sum of squared distances to all rays cast through the center of 2D bounding boxes. We define the transform parameters $\Lambda \in \{\lambda | t_x, t_y, t_z, s\}$ where $t_i$ is the translation in the *i-axis* and $s$ is the scale. We further improve this first estimate of the transform parameters $\Lambda$ by leveraging the 2D bounding boxes from object detection (Fig. 4.6a). By minimizing the distance between detection bounding boxes and the bounding boxes from 3D projection, we obtain a better initial estimate of the rigid transform parameters $\Lambda$:

$$\Lambda^* = \arg\min_{\Lambda} \sum_{i=1}^{n} dist(x_1^i, d_1^i)^2 + dist(x_2^i, d_2^i)^2 \tag{4.1}$$

where $\{x_1, x_2\}$ and $\{d_1^i, d_2^i\}$ respectively define, for view $i$, the sets of upper left and bottom right corners of the 2D bounding boxes of the object and the detection. The distance function $dist$ is defined as:

$$dist(x, x^d) = \begin{cases} 0 & \text{if } x \text{ or } x^d \text{ on image border} \\ \|x - x^d\|_2 & \text{otherwise} \end{cases} \tag{4.2}$$

For this optimization we only consider an edge of a bounding box if the edge is entirely within the image (i.e., the car is not truncated at that side). We use gradient descent to estimate these initial set of parameters $\Lambda$. We provide the full formulas for the partial derivatives in Appendix B.




(a) (b)

Figure 4.6: **(a) Constraints from detection 2D boxes:** To have a better starting point for object alignment, we use the constraints from the detection bounding boxes. The bounding box corners $x_1$ and $x_2$ of the mesh should match the corners $d_1$ and $d_2$ of the R-CNN detection box. **(b) Constraints from 3D reconstruction:** For each camera, the line of view for object vertices $v^1$ and $v^2$ intersects the available 3D reconstruction (in green; note that part of the car is missing) at 3D points $v^1_{\text{mvs}}$ and $v^2_{\text{mvs}}$. In this case, the constraint from $v^2_{\text{mvs}}$ is ignored.

### 4.5.2 Multi-View Alignment

After the initial pose estimation, the model is placed in the correct general region of the 3D scene. It now becomes possible to further improve the alignment of the 3D model using contours and available 3D reconstruction. We continue to use $\Lambda$ to indicate pose parameters (rotation, translation and scale) but this time we include rotation quaternions $\Lambda \in \{\lambda | t_x, t_y, t_z, q_x, q_y, q_z, q_w, s\}$. We model the alignment step by solving the following optimization problem:

$$\Lambda^* = \arg\min_{\Lambda} E(\Lambda) = E_{3D}(\Lambda) + E_{edge}(\Lambda) \tag{4.3}$$

The first term ($E_{3D}$) corresponds to distance between the partial 3D reconstruction from MVS and the mesh. The second term ($E_{edge}$) tries to align the silhouette of the mesh with the corresponding edges in the images.

**Constraints from partial 3D reconstruction.** Multi-view stereo algorithms (Jancosek & Pajdla, 2011; Goesele *et al.*, 2007) provide a coarse 3D reconstruction of the object. This reconstruction often contains inaccuracies and holes, typically in regions containing windows or strongly reflective surfaces where depth estimation from stereo matching algorithms is unreliable. Nevertheless existing 3D information should be used to align the model. The initial pose of the matched 3D model provides a rough overall scale and position of the object, allowing us to identify with some accuracy which parts of the 3D reconstruction correspond to the model we wish to align.

We note $\mathcal{V}^i_{\text{visible}}$ the set of the visible 3D model vertices from camera $i$. When MVS reconstruction is available, we associate to a vertex $v$ its closest point $v_{\text{mvs}}$ on the line view. $E_{3D}(\Lambda)$ is defined as:

$$E_{3D}(\Lambda) = \sum_{i=1}^{n} \sum_{(v, v_{\text{mvs}}) \in C^i} \|v - v_{\text{mvs}}\|^2 \tag{4.4}$$

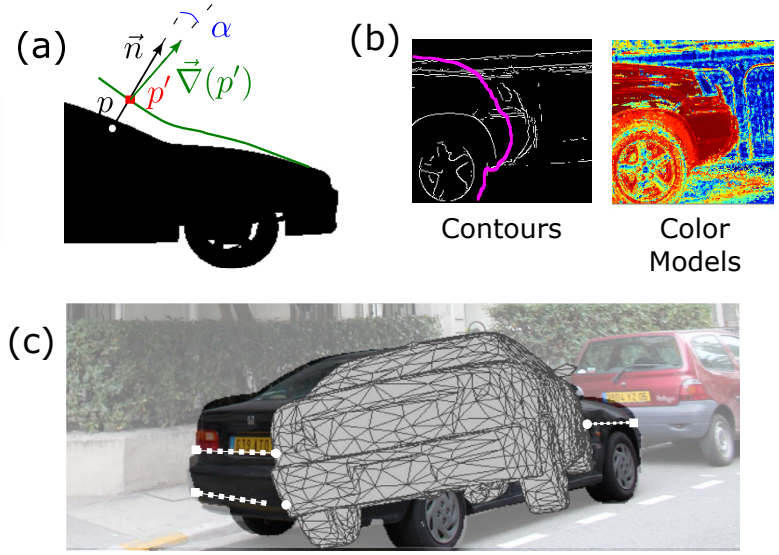



Figure 4.7: **Constraints from silhouettes.** (a) For a point $p$ on the mesh contour we look for a matching point along the lined defined by $\vec{n}$. A candidate matching point $p'$ is added as 2D constraint if $\alpha < 15°$. (b) In certain situations, the mesh contour in pink can have two corresponding contour points. In this case it is necessary to use color models (warm colors indicate high object probability). (c) The resulting constraints help align the 3D mesh (in gray).

with $C^i$ the set of valid $(v, v_{\text{mvs}})$ pairs obtained from view $i$. As illustrated in Figure 4.6b, when the reconstruction is far from the model, it is unlikely that $v_{\text{mvs}}$ is part of the car and it should not be considered. In our case we use 1/5 of the object length as the filtering threshold. This is a common strategy in point cloud alignment literature (Rusinkiewicz & Levoy, 2001).

**Constraints on the silhouette.** For silhouette matching we first need to identify relevant contours in each image. We use the Canny filter (Canny, 1986) to detect edges in the images. The output is an edge map only based on local color differences. Isola *et al.* (2014) describe a method to estimate the statistical dissociation information on boundaries. We use this information to filter the edge map and only keep edges likely to correspond to object contours.

Once the contours in the image have been identified, the next step is to match points from the object's silhouette with edges detected in the image. If we consider $p$ a point on the silhouette and $\vec{n}$ the normal to the silhouette at this point, we search for matching candidates along the normal line passing by $p$. We keep the two closest edge points to $p$ as a matching candidate. This procedure can generate a large number of candidate matches with potentially many outliers. We first use the normals as filtering criteria. Let $\vec{\nabla}(p')$ be the color gradient at the image candidate pixel $p'$. If the angle difference between the vectors $\vec{\nabla}(p')$ and $\vec{n}$ is larger than 15°, the candidate pixel $p'$ is discarded (Fig. 4.7.(a)). This criterion is sufficient in many cases but certain situations require the usage of color information (Fig. 4.7.(b)). Pixels inside and outside the model projection are

used respectively to estimate a foreground and a background color model. Here, these color models are histograms noted $\mathcal{H}^F$ and $\mathcal{H}^B$. Fig. 4.7 shows the resulting *object* probability for pixels (warm colors indicate high object probability). We use these models to help filter incorrect matches by defining the *color* energy associated to a contour point $p$ and a vector $\vec{n}$ as

$$E_{\text{color}}(p,n) = \sum_{p\in\mathcal{S}_{\text{int}}} -log(\mathcal{H}^F(I_p)) + \sum_{p\in\mathcal{S}_{\text{ext}}} -log(\mathcal{H}^B(I_p)) \tag{4.5}$$

$\mathcal{S}_{\text{int}}$ and $\mathcal{S}_{\text{ext}}$ are set of points sampled along the line defined by $\vec{n}$. They respectively correspond to interior and exterior points. The defined energy is lower when interior and exterior points respectively satisfy foreground and background color distributions. Any silhouette match that results in an increase in contour energy is ignored. In essence, this is similar to energy terms used in level set segmentation (Cremers *et al.*, 2007).

We further enforce multi-view coherence by extending our filtering scheme. Each 2D constraint is transformed into a 3D constraint by using the vertex depth value. These 3D constraints are projected in all the other views, resulting in new 2D displacements. Views where this displacement causes an increase in the appearance energy $E_{\text{color}}$, vote to drop the constraint. We only keep the constraints for which a majority of views agrees. The energy term from silhouette matching is defined as:

$$E_{edge}(\Lambda) = \sum_{i=1}^{n} \sum_{(p,p')\in\mathcal{M}^i} \|p - p'\|^2 \tag{4.6}$$

with $\mathcal{M}^i$ the set of 2D matching points from silhouettes in view $i$.

**Optimization.** We solve the alignment problem of Eq. (4.3) using gradient descent. All differentiations with respect to each pose parameter are provided in Appendix B.2.

Typical outputs of initial pose described in Section 4.5.1 and in the fine alignment Section 4.5.2 are shown in Fig. 4.8. These two steps are not sufficient, since the mesh of the model is not exactly the same as the model in the input photos as we saw in Fig. 4.8. This happens due to inaccuracies at the retrieval step or just because the appropriate model may be missing from the database. A subsequent *morphing* step (in Section 4.6.2) will be required to accurately fit the model in the images; however high-quality alignment using a rigid transformation is indispensable for the success of morphing and finally high-quality IBR.

## 4.6 Automatic Geometry Morphing and Rendering

We now have a well aligned 3D model in our scene. However, due to the inevitable differences between 3D models retrieved and the 3D object observed, the 3D mesh needs to be adapted to fit the object in the images as best as possible. This happens due to several reasons:

Figure 4.8: **Positioning the mesh.** Upper row: Initial alignment for two views; Bottom row: fine-grain alignment.

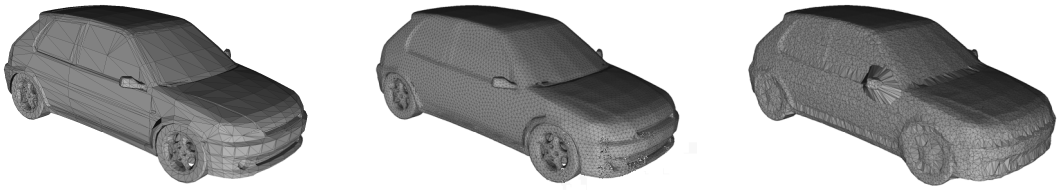
- Inaccuracies at retrieval step.
- Inaccuracies at the alignment step.
- Inaccuracies of the 3D model.
- Or simply, because objects such as cars exist in various options and the appropriate model might be missing in the database, or the specific instance of the object may have small differences such as different accessories etc.

A key element of our approach is that we *automate* this morphing process using the 2D silhouette matching obtained in the previous step, contrary to previous methods based on manual user interaction (Kholgade *et al.*, 2014).

### 4.6.1 Mesh simplification

Mesh deformation techniques, such as the As Rigid As Possible (ARAP) morphing (Sorkine & Alexa, 2007), require high quality manifold meshes. Unfortunately, meshes available in *ShapeNet* and other large databases are unsuitable for such transformations. They often present poor quality due to their diverse and "casually modelled" origin, often containing duplicate vertices and faces, self-intersecting polygons, disconnected components which are generally unsuitable for further geometry processing.

We tried different methods to repair the meshes, including re-sampling the hull of the meshes with points and using screened Poisson reconstruction with the software *Graphite*[4] While the resulting meshes are reasonable, the final mesh is not guaranteed to be manifold and small holes due to details in the 3D modeled mesh remained which create problems during rendering. An example of this kind of mesh is presented in Fig. 4.9(b). As a reasonable compromise, we opted for the *semi-convex hull* representation (Guney & Geiger, 2015) (see Fig. 4.9(c)). This method preserves the shell of the objects, and outputs a manifold mesh suitable for further processing while keeping desired properties as a side-effect.



(a) Original mesh (b) Re-sampled mesh (c) *Semi-convex hull*

Figure 4.9: **Mesh simplification.** (a) Retrieved mesh from *ShapeNet*. (b) Re-sampled mesh with *Graphite*. (c) *Semi-convex* hull mesh of Guney & Geiger (2015).

### 4.6.2 Morphing

To deform the mesh, we obtain 3D constraints on vertices from 2D silhouette matching. For every silhouette point $p$ matched with an edge point $p'$ in the image, we obtain the new position of the mesh vertex $v^i$ projecting on $p$. This new position, $v^i_{\mathcal{M}}$, is located along the viewing line of $p'$ at the same depth as $v^i$. We enforce a smooth deformation of the rest of mesh using the As-Rigid-As-Possible surface deformation framework (Sorkine & Alexa, 2007) available in *libigl*[5] (Jacobson *et al.*, 2016).

### 4.6.3 Rendering

The rendering proceeds in three passes. The first pass renders the background environment using the mask generated by the previous process. We use our previous implementation of *selective* rendering (Ortiz-Cayon *et al.*, 2015) to render this layer, and modify the shader to discard all pixels on the objects which have geometry using the mask. We see this layer in Fig. 4.13(left), which is the result of blending the four closest views. The second pass performs a ULR-like rendering of the car using the aligned and morphed geometry (figure middle). Specifically we used deferred shading to render the depth and normals of the 3D model, look up the color in the closest images and use the ULR blending weights (Buehler

[4] http://alice.loria.fr/software/graphite/doc/html
[5] https://github.com/libigl/libigl

Figure 4.10: **Mesh morphing.** Because the selected 3D mesh may not correspond exactly to the model in the images, a deformation step is necessary. We use silhouette matching to define constraints on vertices: red points indicate original positions and white points target positions. A multi-view filtering step (see text for details) is necessary to remove remaining outliers (indicated in green). After morphing, the mesh better fits the images and can be used for rendering.

*et al.*, 2001) to synthesize the final color on the object. Finally we blend background and object layers directly on the GPU to produce the final result (figure right).

## 4.7 Results and Comparisons

We evaluate our method on 5 scenes. We use the scenes YellowHouse-12 and Street-10 from Chaurasia *et al.* (2013) that contain cars. We also propose the new scenes HotelBruges-19, Bosquet-16 and Street2-26. Results of our real-time rendering are shown in Fig. 4.2, Fig. 4.13 and 4.14. but are best appreciated in the accompanying video, available here: `https://team.inria.fr/graphdeco/publications`.

**Comparisons.** We compare our method with previous rendering algorithms (Buehler *et al.*, 2001; Ortiz-Cayon *et al.*, 2015). Fig. 4.14 shows the result of rendering for each algorithm, at a novel viewpoint far from the input cameras. ULR (Buehler *et al.*, 2001) relies entirely on the available geometry and errors in the reconstruction are particularly visible (one can also notice the black regions where no 3D data exist). Our previous *selective* approach using superpixels, performs better on the background in general. It compensates a little for errors in the geometry (1st row), but starts to show strong artifacts as we move closer to the reconstructed background (3rd and 4th row). Just using the initial pose of the stock model may improve the rendering when no 3D data is available (row 3) but results are blurry and artifacts are created around edges. Our approach outperforms other rendering algorithms and it is able to compensate for the errors in the geometry. Thanks to alignment and morphing, the renderings around contours look natural and much fewer artifacts are visible. It now possible to move closer to objects in the scene contrary


Figure 4.11: **Full Mesh Fitting** (a) Using 2D detection bounding boxes, an initial transformation of the 3D model is computed. (b) Using available 3D reconstruction and constraints from silhouette matching, a fine grain alignment of the model is estimated. (c) Because the 3D model does not always correspond to the images, deforming the mesh is necessary. Red points indicate mesh vertices to be displaced. White points indicate their target position.

to previous methods. To show the importance of the alignment step, we also compare with rendering based on initial pose of the 3D model in Fig. 4.15.

As we explained in previous chapter, we attempt to quantitatively validate our results with image quality metrics following a leave-one-out procedure but comparing only cropped images (with bounding boxes from detection). Although this time we always obtain higher index of quality, the differences are insignificant with respect to the values obtained with ULR and Selective. We could not conclude anything with this measure since as see in Fig. 4.16, the error is spread through the image, making it hard to reach any meaningful conclusion. Again, we concluded that these metrics are not suited for Image-based rendering evaluation.

## 4.8 Conclusions

In this chapter, we introduced a new mixed Image-Based Rendering algorithm that builds on recent advances in object detection and recognition to augment IBR scenes with explicit geometry. We propose an entirely automatic pipeline that starts from object detection in

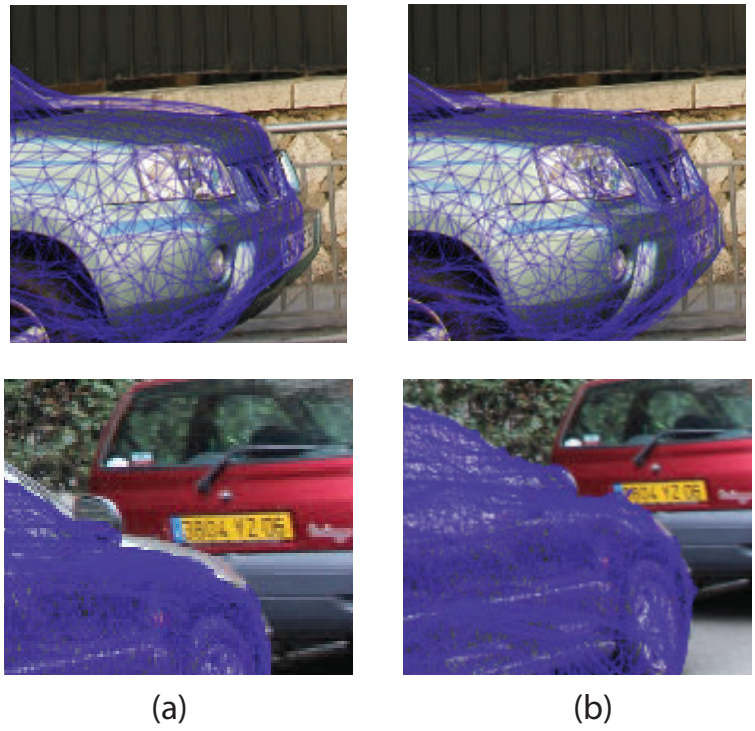



Figure 4.12: **Detail of mesh morphing.** (a) Detail before morphing. (b) After morphing we obtain a better alignment of the 3D model.

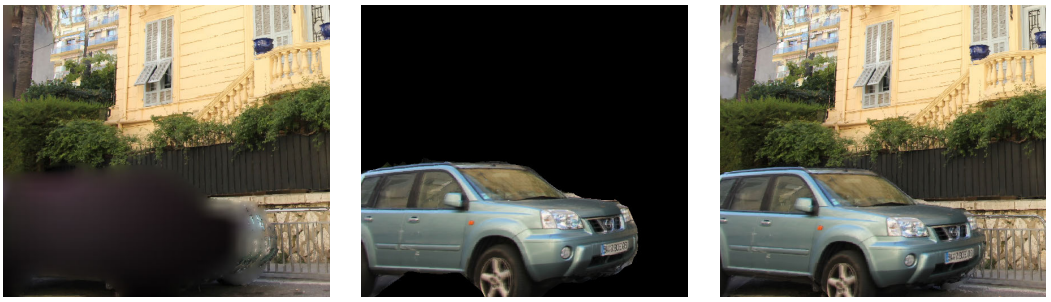


Figure 4.13: **Mixed-rendering.** Left: Background layer. Middle: Foreground object. Right: Final novel view.

images, then accurately places the object in the scene using a multi-view approach taking advantage of available geometry and silhouettes. As the stock 3D model may not exactly correspond, the 3D mesh is morphed to better fit the images. Our results demonstrate that we obtain improved rendering quality even when moving away from the input cameras.

For the moment only car 3D model databases are rich enough to be used in our context, but our method is generic and can be directly applied on other object categories and we would like to test our approach with other classes. We see our approach as a first step in a more general trend, in which traditional 3D models will be combined with image-based techniques to greatly simplify 3D content creation and interactive display. This is a

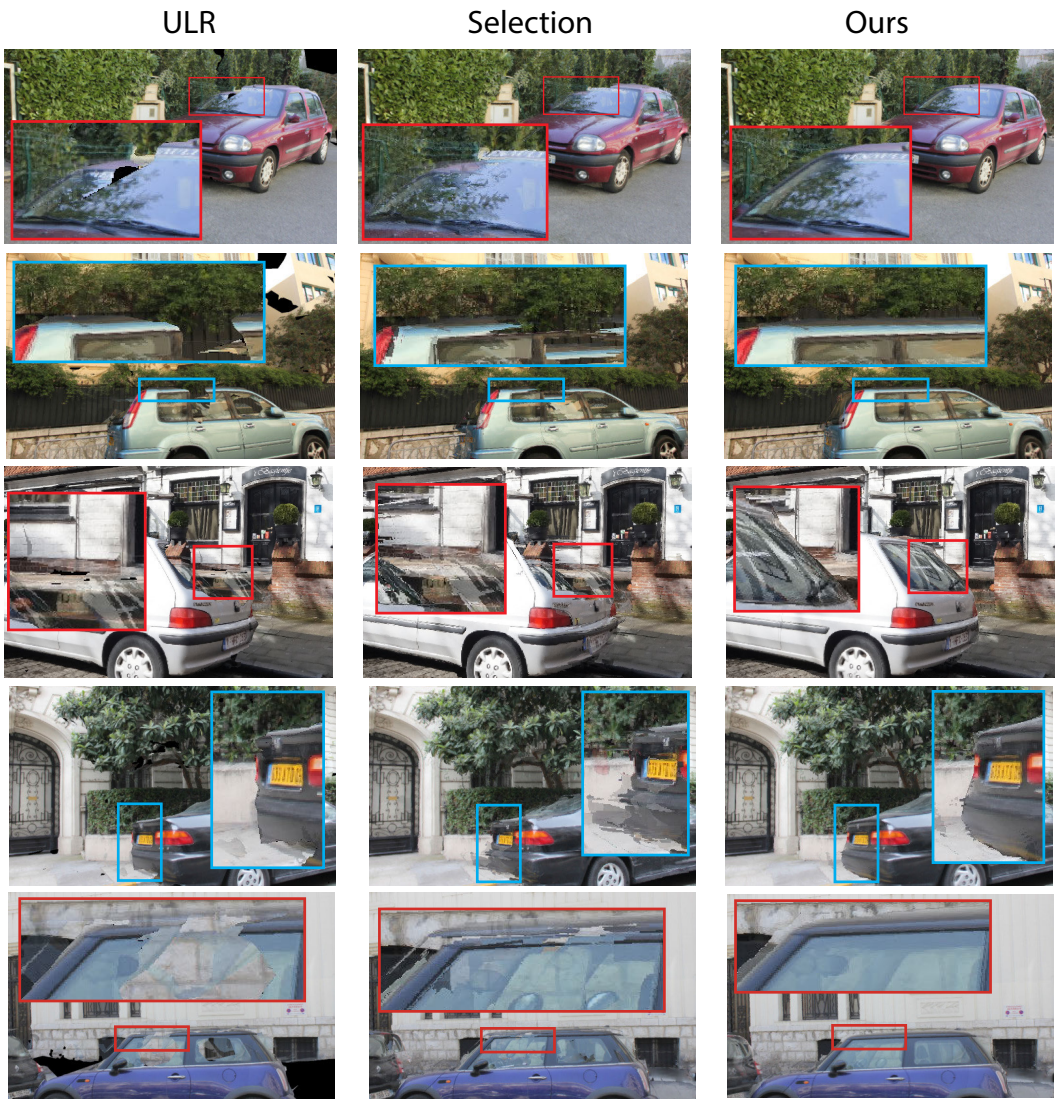



Figure 4.14: **Comparisons.** Rendering of novel viewpoints on Bosquet-16, YellowHouse-12, HotelBruges-19 Street-10 and Street2-26 datasets (from top to bottom). The rendering methods are, from left to right, ULR (Buehler *et al.*, 2001), Selection (Ortiz-Cayon *et al.*, 2015), Ours without alignment, Ours after alignment.

promising direction as more and more 3D models are available. Additionally, we would like to incorporate the semantic knowledge of new databases as *ObjectNet3D* (Xiang *et al.*, 2016) to render retrieved objects bases on their material properties.

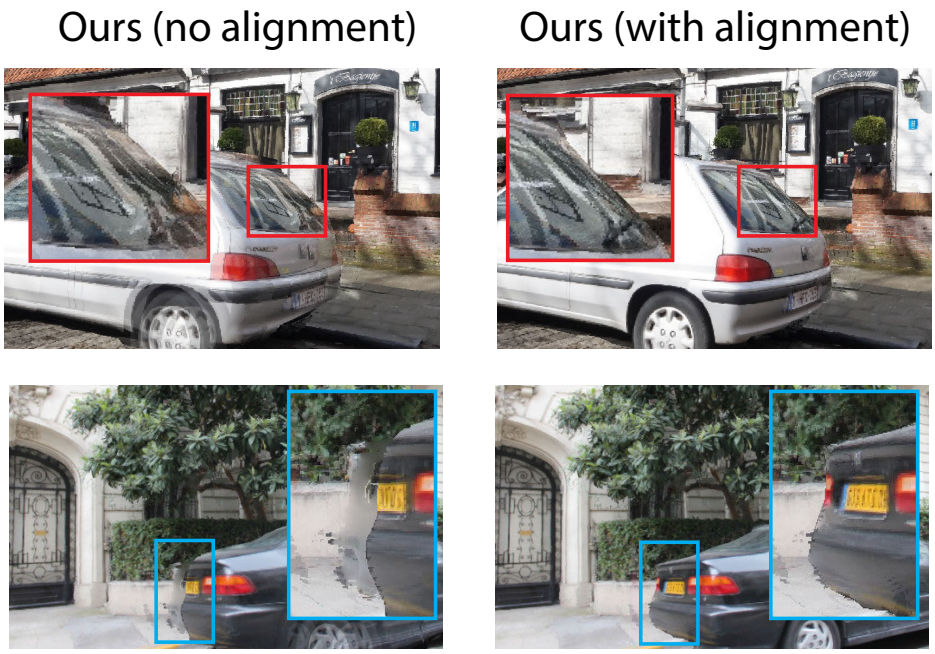
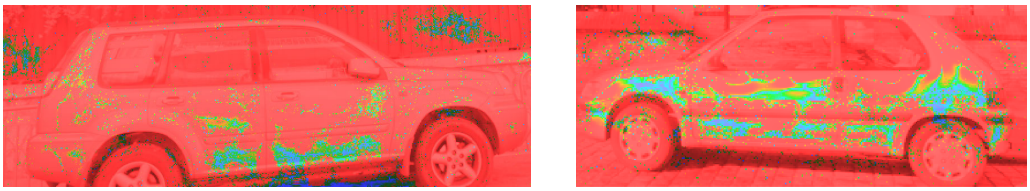



Figure 4.15: **Comparisons with and without alignment.** (a)Ours without alignment. (b) Ours after alignment.



(a) Original mesh (b) *Semi-convex hull*

Figure 4.16: **Visual Difference Predictor map.** Example of prediction maps using VDP (Mantiuk *et al.*, 2011) in our cropped rendered images.

CHAPTER 5

# Quality of Reconstruction and Geometry Correction for IBR

## Contents



In Chapter 3 and Chapter 4 we presented Image-Based Rendering methods that can generate plausible views under imprecise geometry. We expect better rendering quality as the reconstruction completeness and accuracy improves. An ideal IBR method should also be capable to adapt to different levels of reconstruction quality, and still perform well if only a poor geometric proxy is available. An interesting question is to ask what is the level of precision required for geometric reconstruction to still allow rendering of *plausible views*? The answer is unclear because:

1. Rendering methods are complex. Their artifacts are hard to analyze.
2. We have not completely understood the human vision process and we miss good perceptual models.

IBR methods presented in this thesis as well as state of the art methods use heuristics to minimize target functions, achieving excellent results. So far we have used the reconstruction process as a black box system, without explicitly measuring uncertainty introduced in the final rendering. Being aware of depth uncertainty would allow us to define rendering strategies and possibly improve depth for IBR.

In this chapter we attempt to improve IBR by identifying erroneous depth with a classification task formulated as a feature-based energy (in Section 5.2) and with a

learning-based energy (in Section 5.3). In the former, we try to identify areas labeling them based in a energy using classic descriptor-based approach. In the latter, we train a deep network to learn to predict corrections of depth maps. We present partial results of both approaches which give us a first insight on how to obtain better geometry for IBR.

## 5.1 Introduction

We reviewed recent multi-view stereo reconstruction algorithms at a very high level in Section 2.2.1. With around 20 images of the scene viewed in Fig. 5.1a, Jancosek & Pajdla's method gives us the best proxy we have been able to obtain from tests with publicly available solutions (seen in Fig. 5.1b). We use mesh-based surface representation of the geometry which generally offers fewer outliers compared to point clouds representations. Projecting the proxy onto input cameras produces the depth maps we use for rendering (see Fig. 5.1c).

Given this scene's complexity, the results of these state of the art computer vision methods are impressive. Nevertheless those are far from being perfect. Inaccuracies on the proxy introduce rendering artifacts. Defining how imprecise geometry degrades rendering quality is a complex task. Pujades *et al.* model uncertainty of the geometry by back-projecting – into 3D space – feature points uncertainties of the matching algorithm. The estimated uncertainty weights the "minimal angular deviation" criteria of *ULR* in their optimization.

For real-time rendering application, we can informally describe this uncertainty of depth maps as three main classes. "Holes" (black regions in Fig. 5.1c) are the most evident and trivial to identify. Chaurasia *et al.* proposed an iterative algorithm to synthesize view depending depth base on the appearance of neighboring regions with depth. This procedure produces front-facing depth good enough when we move the nearby input cameras but rendering artifacts appear when we move far. When displaying in stereo or at slanted planes (like the ground floor in Fig. 5.2(b)), we clearly see the "billboard-like" nature of the synthesized depth.

Another kind of problem harder to identify, happens in regions with inaccurate depth as in Fig. 5.1d. We get depth information everywhere, but actually, it is too far from what it should be. It could be either extra geometry – thus some fake geometry occludes the background – or missing parts of objects – and thus it takes depth of the background. Rendering artifacts of this kind of error can be seen in Fig. 5.2(b).

Our goal is to reduce the rendering artifacts due to inaccurate view dependent or multi-view reconstruction. We target complex environments, typically outdoors urban content, in which a lot of depth cannot be reconstructed. Indoors scenes pose specific challenges (Hedman *et al.*, 2016), but in principle the solutions we propose here could be extended to interiors.

A frequently used approach to solve this kind of problem is measuring the evidence with hand-crafted descriptors (data term) and formulating an energy with some prior observations to regularize the solution (smoothness term). We follow this approach in Section 5.2. Although we have control over the feature, it is difficult to define meaningful

descriptors for the task. Deep Learning has proved to be effective for this kind of tasks. The challenge consist in gathering a massive amount of data for training and in establishing an appropriate network architecture for the problem. In Section 5.3 we explain the procedure to create our training data and the proposed network architecture.

## 5.2 Descriptor-based Approach

In this section we present an attempt to identify problems of depth maps for IBR, using a descriptor to measure the evidence of inaccurate geometric information in our input views. In particular, different regions of depth maps generated from MVS proxies can be classified into: Reliable (reconstructed), Unreliable (missing occluder or extra occluder) and No geometry (no depth). Next, we describe some observations of these regions to understand the their characteristics:

**Reliable depth - reconstructed.** Surfaces with texture and *Lambertian* properties, visible in two or more views. Photo-consistent properties of this type of region can be

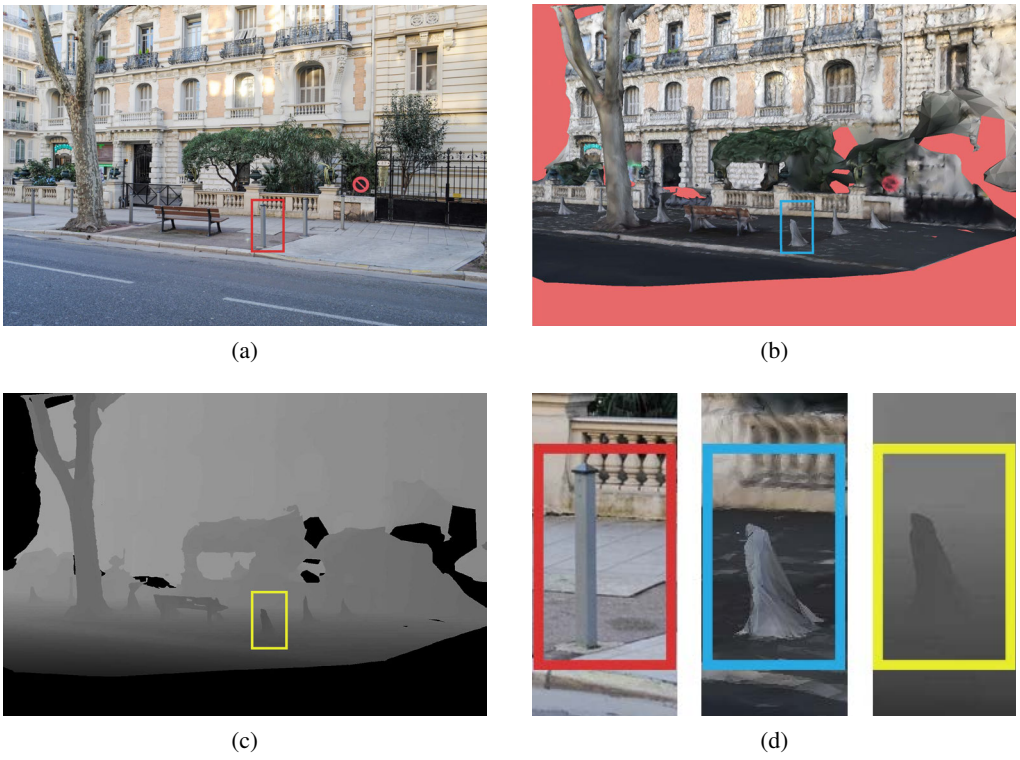



Figure 5.1: **Problems of MVS reconstruction.** (a) One input image. (b) Reconstructed proxy seen from (a). Proxy''s holes in red. (c) Depth map extracted from (b). (d) Example of two kind of inaccuracies of the reconstruction: Geometry of the upper path of the pole is missing while the base has inexistent geometry.

(a) (b)

Figure 5.2: **Rendering artifacts produced by erroneous geometry.** (a) Top view with input cameras in blue and novel camera in red. (b) Image from the novel camera in (a): Visible rendering artifacts around the vertical poles and in the ground floor from another novel camera far from inputs.

verified by re-projecting depth into other views.

**Unreliable of class 1: missing occluder.** Rarely, an object in the reconstruction scene is completely missing. Thus, image regions containing unreconstructed objects, but *with* a reconstructed background, belong to this category

**Unreliable of class 2: extra occluder.** This case happens when the reconstruction algorithm hallucinates geometry (and thus depth), which does not exist in the real world. In the reconstruction algorithms we tested this often occurs around vegetation or in-between thin repetitive structures (like fences). Usually the parts that subtend the extra geometry are well reconstructed.

**No geometry.** Regions where no reconstructed geometry is projected. In recent reconstruction algorithms, regions with noisy texture (clutter) or with repetitive patterns are less problematic for reconstruction. The real challenge remain regions that change their appearance drastically from one view to another or simply do not have matching pairs (e.g. at the border of a photograph).

Trivially, any region of the depth maps without depth will be labeled with "*no-geometry*". But for the other two types of regions, which tools do we have to measure these observed properties? For "*missing occluder*" one could measure the "objectness" of regions and verify that depth of regions that belong to the same object should not change quickly. This is hard to measure in our setups where there is no single object which is the center of attention but we move freely in the scene. Instead, we can reason on visibility, considering that the background should not be visible by the view which is missing the occluder, as we will propose in Eq. (5.3).

"Reliable depth" can be measured via photo-consistency. For this task we use the *Daisy* descriptor proposed by Tola *et al.* (2010), inspired by *SIFT* and *GLOH* descriptors.

Daisy was efficiently computed at each pixel (two orders of magnitude faster than *SIFT*) It has been specifically designed for dense wide-baseline stereo matching, handling occlusions correctly and presenting desirable robust properties to perspective distortions, lighting changes and rotations. We also expect that low photo-consistency indicates "extra occluder".

### 5.2.1 Descriptor for Geometry Labeling

For each of our $M$ input images $I_m$ associated with camera $C_m$ and depth map $\mathcal{Z}_m$, we can obtain Daisy descriptor maps $\mathcal{D}_m$. Given this map, the Daisy descriptor at a pixel coordinate $\mathbf{x}^i$ of $I_m$ is equal to:

$$\mathcal{D}_m(\mathbf{x}^i) = \mathcal{D}_m(X^i) = [h_1, ..., h_Q] \tag{5.1}$$

Where $X^i$ is the 3D point that projects to location $\mathbf{x}^i$ with the projection matrix of $C_m$. $\mathcal{D}_m(\mathbf{x}^i)$ is a feature vector with a set of $Q$ normalized oriented histograms $h_q$ – centered around the point of interest at different levels and considering regions of different sizes. For a pixel $\mathbf{x}^i$ in view $m$ we obtain the matching pixel in view $n$, projecting $X^i$ onto camera $C_n$ and thus a matching descriptor $\mathcal{D}_n(X^i)$.

The distance $d^i_{m\to n}$ between matched points can be computed as the mean $\chi^2$ error between histograms[1] of $\mathcal{D}_m(X^i)$ and $\mathcal{D}_n(X^i)$. We define the descriptor *for Geometry labeling* of the pixel $\mathbf{x}^i$ in $I_m$ (Eq. (5.2)) and its complement (Eq. (5.3)) as:

$$\mathbf{d}^i = \sum_{n\in\mathcal{N}}^{|\mathcal{N}|} w_n d^i_{m\to n} \tag{5.2}$$

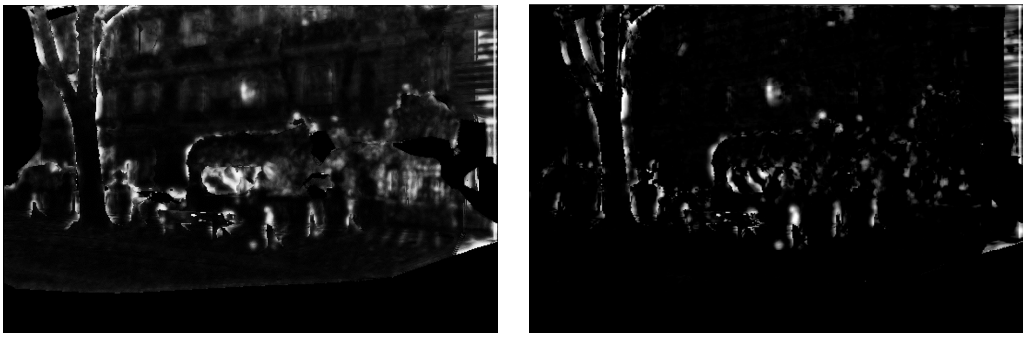
$$\mathbf{d}^{*i} = \sum_{n\in(\mathcal{N}\setminus m')}^{|\mathcal{N}|-1} w_n d^i_{m'\to n} \tag{5.3}$$

where $\mathcal{N}$ is the set of views where $\mathbf{x}^i$ is visible – according to the proxy. The weights $w_n$ are the cosine distances (dot product of normalized vectors) between rays from $X^i$ to $C_m$ and to $C_n$. It penalizes matches between slanted views. The spatially closest view to $m$ is denoted by $m'$. The intuition of Eq. (5.3) is that excluding the reference view, we can measure the error between descriptors of other views that see the point of evaluation. Thus if other views generate a small error $\mathbf{d}^{*i}$ with respect to $\mathbf{d}^i$, probably $C_m$ is missing something and the other views see the background correctly reconstructed. Maps of $\mathbf{d}^i$ and $\mathbf{d}^{*i}$ computed for the Fig. 5.1a can be seen in Fig. 5.3.

### 5.2.2 MRF Energy to identify Problematic Regions

To classify depth map regions into the afore mentioned classes we can define the problem as a multi-label MRF optimization on a regular grid (each pixel is a node). To each node $\mathbf{x}^i$ we assign a label $l^i \in \{0, 1, 2, 3\}$ which represents *reconstructed, missing occluder, extra*

[1] To account for occlusion contours, we only consider pairs of histograms with approximate depth.



(a) (b)

Figure 5.3: **Maps of descriptors for Geometry labeling.** (a) Map of $\mathbf{d}^i$ per pixel. (b) Difference between $\mathbf{d}^i$ and $\mathbf{d}^{*i}$ accentuates the missing regions

*occluder* and *no geometry* respectively. We need to find the optimal labeling $\mathcal{L}^*$ that minimizes the following energy with multi-view (*mv*) and single view (*sv*) regularization:

$$\sum_{\mathbf{x}^i \in I_m} E_{data}(\mathbf{x}^i, l^i) + \lambda_1 \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{N}^{sv}} E_{smooth-sv}(\mathbf{x}^i, \mathbf{x}^j, l^i, l^j) + \lambda_2 \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{N}^{mv}} E_{smooth-mv}(\mathbf{x}^i, \mathbf{x}^j, l^i, l^j) \tag{5.4}$$

The data term attempts to measure the properties previously discussed. Concretely:

$$E_{data}(\mathbf{x}^i, l^i) = \begin{cases} \infty, & \text{if } \mathbf{x}^i \text{ has no depth and } l^i \neq 3 \\ e^{\frac{\rho(\mathbf{x}^i)-1}{\sigma}}, & \text{otherwise} \end{cases} \tag{5.5}$$

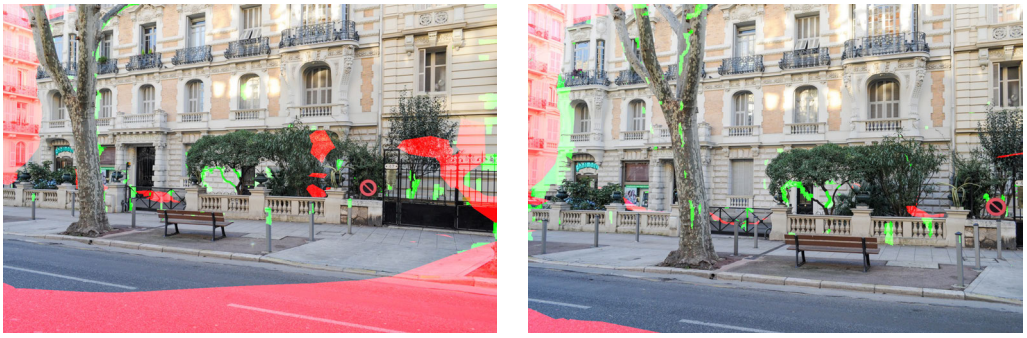
The constant $\sigma$ (between 0.0 and 1.0) indicates a threshold for variability of photo-consistency and the function $\rho$ measures photo-consistency according to the label as:

$$\rho(\mathbf{x}^i) = \begin{cases} \mathbf{d}^i, & \text{if } l^i = 0 \\ \mathbf{d}^{*i}, & \text{if } l^i = 1 \\ \sigma, & \text{if } l^i = 2 \end{cases} \tag{5.6}$$

The pair-wise terms $E_{smooth-sv}$ and $E_{smooth-mv}$ regularize the energy considering the 4-connected neighbors $\mathcal{N}^{sv}$ in single and the 2-closest correspondences $\mathcal{N}^{mv}$ in multi-view respectively. Penalizing the color difference between neighboring nodes as:

$$E_{smooth-sv}(\mathbf{x}^i, \mathbf{x}^j, l^i, l^j) = \begin{cases} w_z(\mathbf{x}^i, \mathbf{x}^j) e^{-\gamma_1 \|I_m^s(\mathbf{x}^i) - I_m^s(\mathbf{x}^j)\|^2}, & \text{if } l^i \neq l^j \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

where $w_z(\mathbf{x}^i, \mathbf{x}^j)$ depends on the difference of depths at $\mathbf{x}^i$ and $\mathbf{x}^i$ as: $e^{-\gamma_2 \|\mathcal{Z}_m(\mathbf{x}^i) - \mathcal{Z}_m(\mathbf{x}^j)\|^2}$. $\gamma_1$ and $\gamma_2$ are constant relative to the relevance of depth and color. $I_m^s(\mathbf{x}^i)$ is the mean color of the superpixel where $\mathbf{x}^i$ belongs to. In this way, we use *SLIC* (Achanta *et al.*, 2012) oversegmentation as a soft constraint to guide the label boundaries towards image edges,



(a) (b)

Figure 5.4: **Labeling for two input views of the test scene.** Regions saturated in red correspond to label $l_3$. Saturated in green correspond to label $l_1$. Original colored pixels correspond to label $l_0$. Any pixel was labeled with $l_2$. (a) A correct labeling: missing parts are detected on the top of the poles, around the tree and the branches of the plant. (b) The consistency is kept on branches of plant, but labeling is been incorrectly projected from (a).

instead of defining the MRF on superpixel nodes. While this could be efficient in terms of computations, some superpixels do not necessarily follow occlusion contours and thus, a superpixel might be "partially reconstructed". $E_{smooth-mv}$ is defined as Eq. (5.7) without considering $w_z$. We downscale input views to a quarter of the original (downscaled to $600 \times 450$ pixels) to reduce computational complexity of the MRF solution.

### 5.2.3 Results and Conclusions of Descriptor-based Approach

We solve Eq. (5.4) with standard Graph Cut minimization (Boykov *et al.*, 2001). The optimized labeling for two input views is shown in Fig. 5.4. Overall, the results were not conclusive. We can explain that with two main reasons. Firstly, the descriptors presented in Eq. (5.2) and Eq. (5.3) might not discriminate correctly between labels and particularly, for the label $l_2$ ("extra occluder") which is very hard to model correctly. Secondly, the choice of optimization method, which might not be appropriate. Graph Cut may not be suitable for this kind of problem because is biased to produce small contours since it finds the minimum cut. This means that Graph Cut is most probably not suited to produce isolated and thin segments presented in complex street images. As a result, we decided to turn the solution of the problem as a learning-based system which we describe next.

## 5.3 Learning-based Approach

Recent years have seen the emergence of several learning-based methods to improve geometric reconstruction. Reynolds *et al.* (2011) presented a supervised learning method to estimate geometric inaccuracies – in the form of confidence – of reconstructions with Time-of-Flight cameras. They train a classifier (*Random Forest regressors*) and as ground truth, they used real-world data, acquired with laser scanners. We can not apply their

classifiers to multi-view stereo reconstructions because artifacts of MVS proxies (described in Section 5.1) are visibly different errors to those produced by Time-of-Flight cameras.

Convolutional Neural Networks (CNNs) have become a very popular supervised learning technique in years. CNNs have been very successful in solving classification and detection tasks (e.g. Krizhevsky *et al.* 2012b). Deep network architectures have been extended to solve segmentation, pose estimation, stereo depth and surface normal estimation (Eigen *et al.*, 2014; Eigen & Fergus, 2015). Broadly speaking, implementing a solution based on deep learning requires computational power, training and validation data and a network architecture. Computational resources and the amount of initial data are the main limitations when implementing such systems. The first one is less of a problem with the availability of deep learning libraries optimized for GPUs processing (e.g. *Caffe*[2], *Torch*[3], and *Theano*[4] ) and architectures for training with distributed resources allows to train deep networks in a matter of hours.

In spite of an impressive growth of publicly available data for training and testing, we did not find suitable databases for our task. Most databases, with depth information, involve Kinect-like captures which themselves do not capture the kind of thin structures we target. Creating databases with meaningful and reliable data to generalize the learning problem is probably what takes the largest amount of time when setting up deep learning systems. Instead of capturing the ground truth depth with laser scanners the option we choose is to generate synthetic data. Renderings have been used before for classification and other vision tasks (e.g. Aubry *et al.* 2014a and Rematas *et al.* 2014) but usually those do not require high quality realistic imagery.

In this section, we explain the procedure to create our training data (discussed in Section 5.3.1), a network architecture (in Section 5.3.2) and the training procedure (in Section 5.3.3). Specifically we need to generate high definition photo-realistic images with ground truth depth that “look similar" to the scenes we capture for IBR. For each scene we use a small subset (15-30) of the synthetic images to run a MVS reconstruction and obtain approximate depth maps from IBR (see Fig. 5.5). Then we train a network in order to find patterns between the input color images, approximate depth and ground truth depth. Building training data from realistic synthetic renderings is a non trivial task and it is a major element of our work.

### 5.3.1 Datasets Generation

Our goal is to produce high-definition natural looking images with ground truth depth. We need to get accurate geometry and lighting details. With the availability of highly detailed hand-crafted models[5] or procedurally generated vegetation[6], and combined with *Mitsuba*[7] (implementing physically-based rendering algorithms, e.g. path-tracing), we

[2] caffe.berkleyvision.org
[3] torch.ch
[4] deeplearning.net/software/theano
[5] www.evermotion.org
[6] xfrog.com
[7] www.mitsuba-renderer.org/

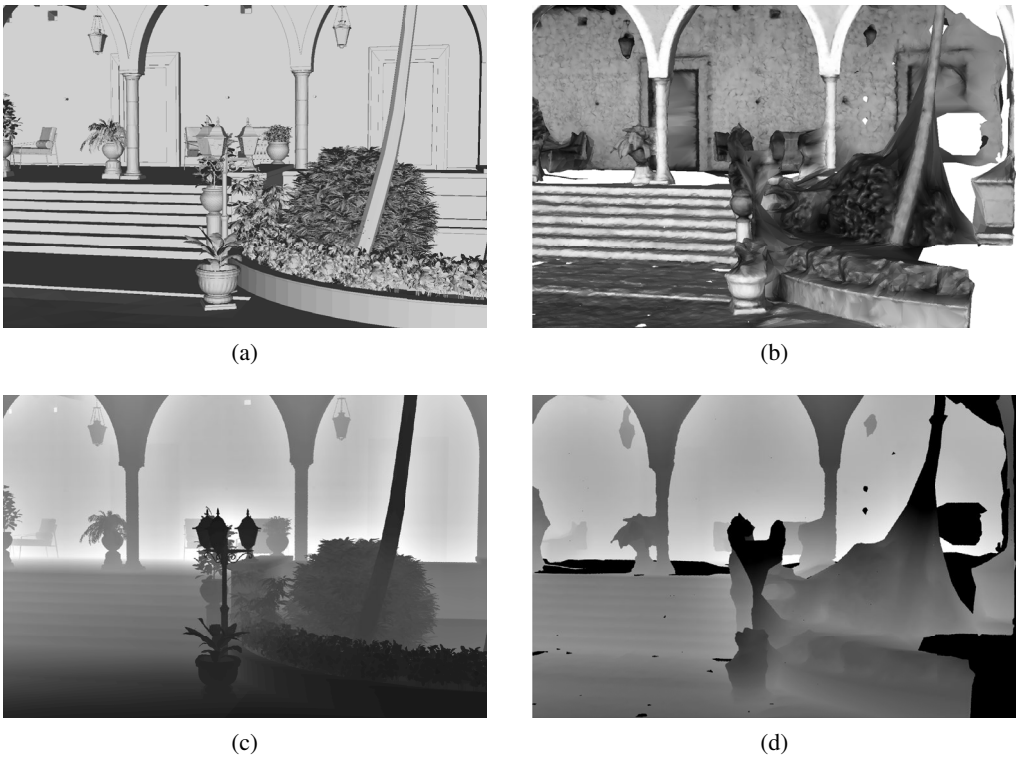
can produce extremely high quality renderings with complex effects and ground truth for different graphics components:, geometry, materials and lighting.

As a proof of concept, we started working only with four scenes (see Fig. 5.6) in 3DS-Max format with *VRay*[8] materials. These scenes were built by professional artists who create these assets for commercial purposes. The quality of the geometry and materials is very high, and includes complex shade trees to give realistic appearance to the rendered images. We can augment this data and generate a lot of variability if we manipulate the scene configuration, materials and lights.

To use these scene many components of our pipeline require some practical solutions developed in our research group[9]. In particular we need to deal with the problems of exporting materials from V-ray to Mitsuba, compatibility between models, different coordinates systems and automation of camera generation. Additionally the elevated cost of physically-based rendering thousands of images, requires high-end computation resources. We chose to use the Inria compute cluster for this task.

[8] www.chaosgroup.com/

[9] The various components of our system were created by G. Kopanas, S. Bangaru and A. Djelouah.



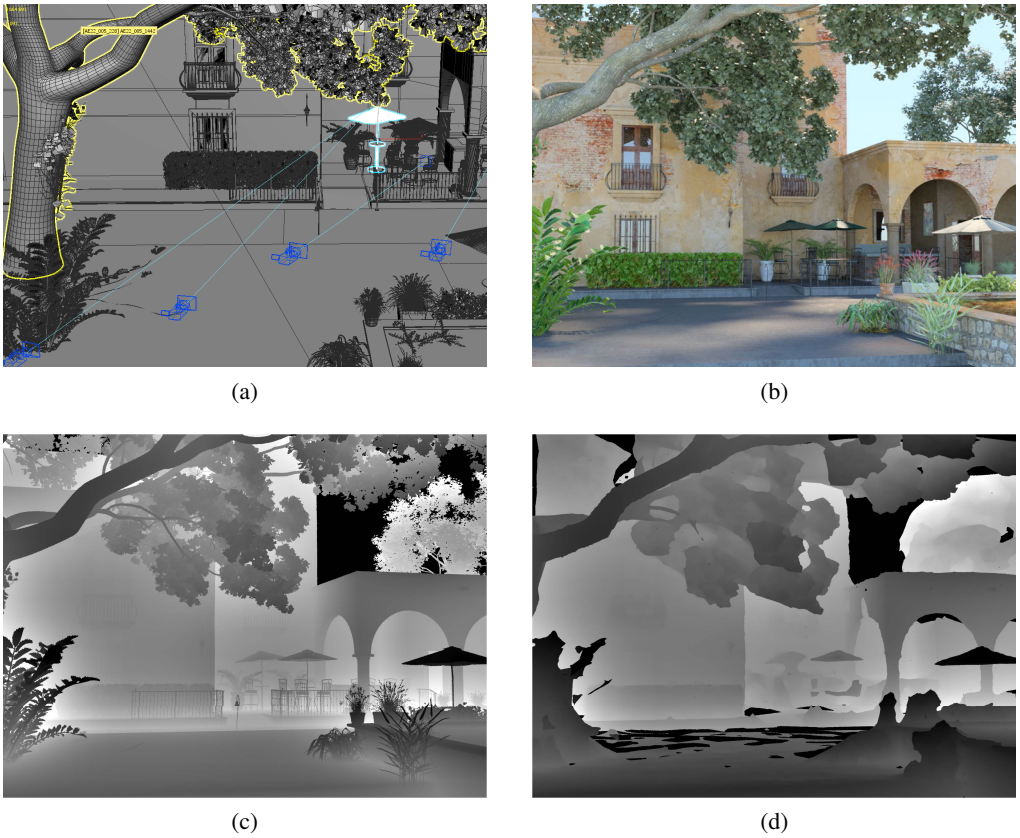
(a) (b) (c) (d)

Figure 5.5: **Depth maps generated from geometry.** (a) Ground truth geometry. (b) Reconstructed geometry aligned with the ground truth geometry. (c) Depth map extracted from (a). (d) Depth map extracted from (b).

**HD Photo-realistic Images and Ground Truth Depth.** We developed a plug-in for 3DS-Max which allows the creation of hundreds of cameras and allows the export of scenes to Mitsuba. The process of creating the cameras for a scene only requires two *splines* (origin and target) used to randomly sample position and orientation of cameras. The plug-in exports the scene geometry and materials to the Mitsuba format which we use in the cluster to generate our synthetic data: RGB images and ground truth depth. Using path tracing in Mitsuba, we render about 600 images of resolution $1600 \times 1200$ for each of our 4 scenes. Computing global illumination for our scenes takes around 2.5 hours per image on a single machine with 16GB RAM Intel-i7 with 12-cores running at 3.20GHz. On the cluster it took less thank two week to render all this data. An example of rendered view and ground truth depth is shown in Fig. 5.7b and Fig. 5.7c respectively.

**Reconstructed Depth.** We select a subset of each dataset (15 to 30 images) in similar paths to those that were used in previous chapters for IBR captures. We use CMPMVS (Jancosek & Pajdla, 2011) to reconstruct the scene proxy. This proxy and the ground truth geometry are not in the same coordinate system. We use an automatic alignment tool to align both the proxy and the ground truth geometry, by using the 3D point information available for each camera from the SfM step. Specifically, for each point reconstructed in each camera, we know the 3D point corresponding to the pixel in the SfM coordinate system, and we can find the 3D position of the same pixel in the original synthetic image.

Figure 5.6: **Four scenes used to generate the training data.** The scenes present highly detailed geometry and include complex effects that "look similar" to the scenes we capture for Image-based rendering.



(a) (b)

(c) (d)

Figure 5.7: **Image patches for training.** (a) View of one of our scenes with virtual cameras in blue. (b) Difference between ground truth and reconstructed depth. (c) Ground truth depth. (d) Reconstructed depth.

We use these matches as constraints to build a linear system and solved with an iterated least squares solver which provides high quality alignment. Once this is done, we obtain depth maps as the one in Fig. 5.7d.

### 5.3.2 Network Architecture

Our task is to learn the mapping between reconstructed depth and an ideal depth guided by color. In principle this task is easier than learning depth only from color. Considering we want good precision, we base our architecture on that developed by Eigen & Fergus (2015), with the only variation that we have an initial approximate depth. They presented a multi-scale architecture to learn coarse features and progressively refine the prediction to a higher resolution. The first coarse layer is equivalent to AlexNet (Krizhevsky *et al.*, 2012b) since it has been designed to discriminate elements in the image. This give us meaningful localized descriptors using only image content. We concatenate our depth at the second and third scale to learn the relations between image content and depth. We use soft-max to compute our loss between the prediction and ground truth depth.

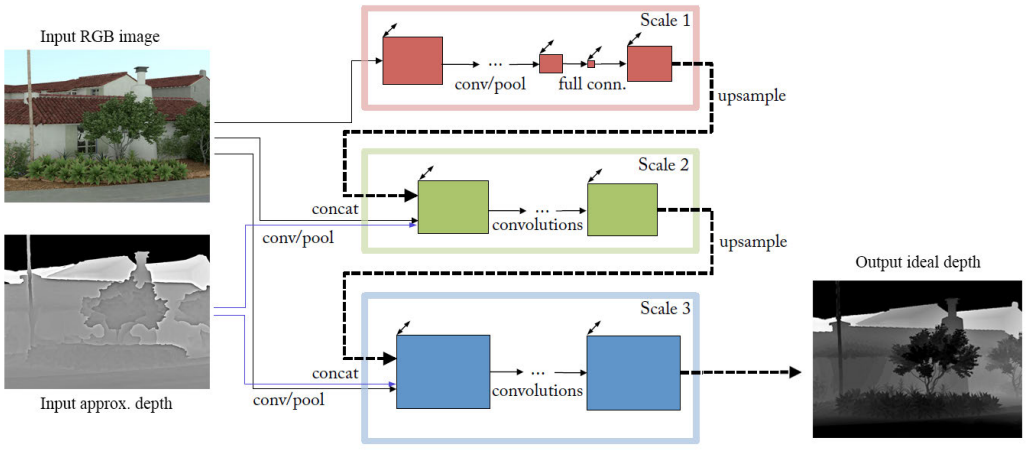
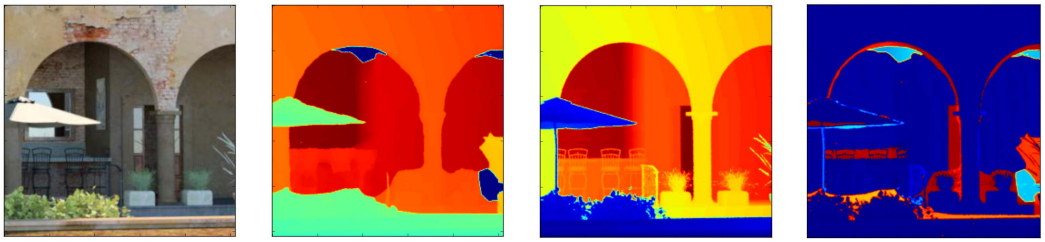



Figure 5.8: **Network architecture.** We use a three scale architecture from Eigen & Fergus (2015) with the small variation that we feed the the second and third scale of the network with our approximate depth maps obtained from MVS. For detail of the layer, refer to Eigen & Fergus 2015.

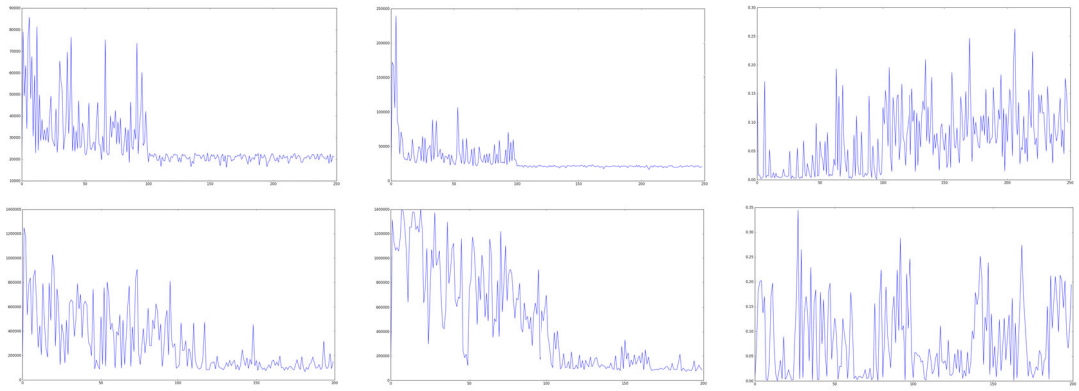
### 5.3.3 Network Training



(a) (b) (c) (d)

Figure 5.9: **Image patches for training.** (a) Color image. (b) Reconstructed depth. (c) Ground truth depth. (d) Difference between ground truth and reconstructed depth.

The training happens in two phases. In the first phase we train the scale 1 (over AlexNet) and scale 2 with low resolution patches of depth ($64 \times 64$ pixels). In the second phase we train the scale 3 to learn refinement of depth maps. Beforehand, we create databases in format HDF5 of patches of images (Fig. 5.9(a)), depth (Fig. 5.9(b)) and ground truth depth (Fig. 5.9(c)). Each patch of the size $256 \times 256$ pixels is created from overlapping regions of our full resolution rendered views. To avoid over-fitting, we augment our data creating patches from three different level of the views: at full resolution, half and a quarter size of the original images from where we create 60, 20 and 4 patches respectively. Our database contains about 180k of patches for each type of data. We randomly take 15% of the patches for validation. Depth is quantized in 255 levels and with a label for absence of depth.

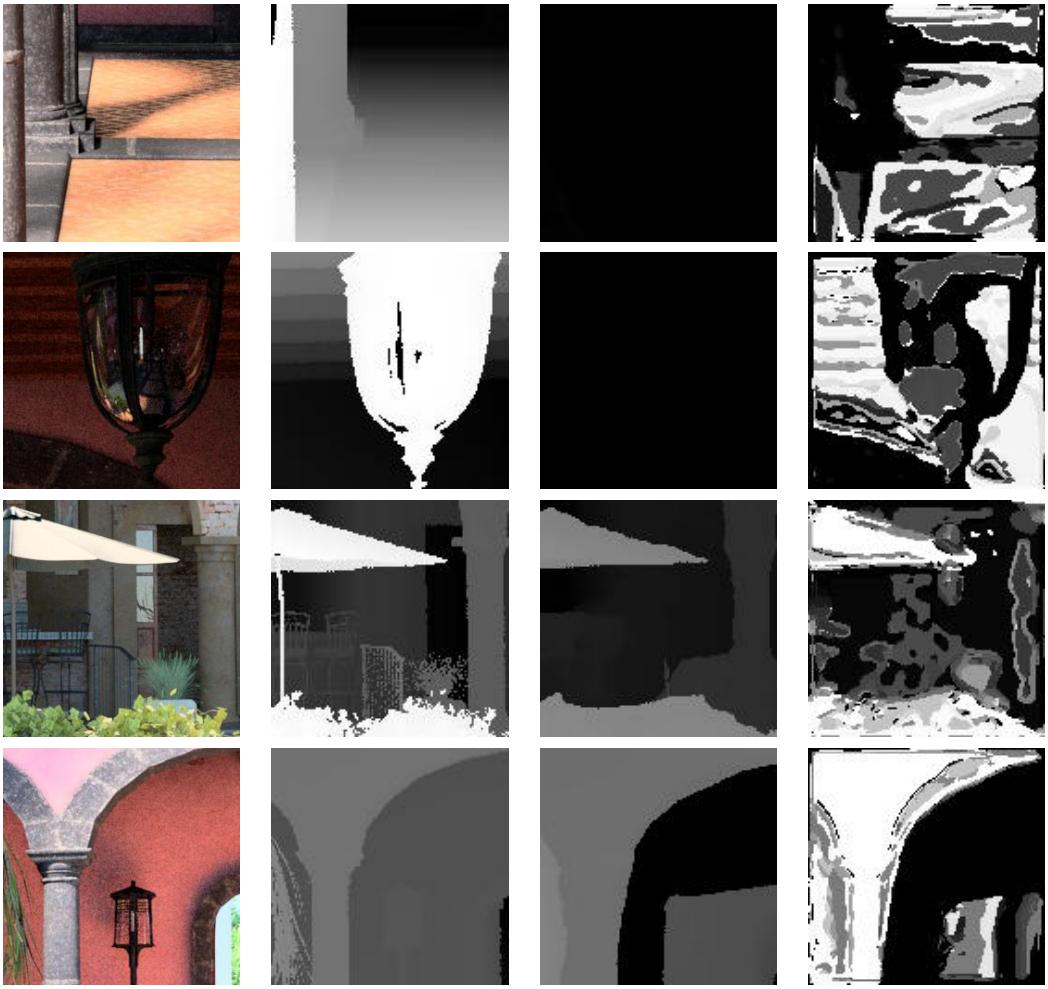


(a) (b) (c)

Figure 5.10: **Learning curves.** Upper row: Training of phase 1. Lower row: Training of phase 2. (a) Train loss vs number of iterations ×500. (b) Test loss vs number of iterations ×500. (c) Accuracy on the validation database vs number of iterations ×500.

### 5.3.4 Results

We have run initial experiments with the data and network described above. In Fig. 5.10 we show the graphs of the loss function value with number of iterations for both the first and the second training phases. In Fig. 5.11 we show the result of depth prediction for four images of the validation dataset. Even though these preliminary results are not conclusive, there are some encouraging traits in the results. In particular, the fist two rows where the MVS depth maps was not available and for thin structures of the last two rows has been recovered (as can be seen in the umbrella's pole, the plans and lamp).



(a) (b) (c) (d)

Figure 5.11: **Example of depth prediction in our validation set.** (a) Color image. (b) Ground truth depth. (c) Reconstructed depth. (d) Predicted depth.

## 5.4 Conclusions and Further Work

In this chapter we have discussed two strategies to address the difficult problem of obtaining accurate depth which is aligned with image contours, in the presence of large reconstruction errors such as those existing in MVS reconstructions of outdoors scenes. We first presented a hand-crafted feature based approach, which despite some encouraging indications does not succeed in the goal of providing high-quality depth. We then present initial steps of a learning-based system which uses *realistic synthetic data* to create high-quality training data for depth estimation. Since recording high quality depth for use as ground truth of natural environments is a challenge itself, we discussed the data generation system for ground truth data and the initial network architecture used for training and depth estimation.

Also, we would like to investigate loss functions which are more meaningful for

view synthesis by combining our powerful ground truth depth information with a quality measure on the rendering itself. Recent work along similar lines has been proposed by Godard *et al.* (2016) and Kalantari *et al.* (2016). Godard *et al.* use stereo pairs to learn to predict each other and do not actually need ground truth depth. Their loss function enforces consistency between disparity pairs, implying good quality depth. While Kalantari *et al.* use Light Field images and use a two step procedure to first estimate disparity and then use a loss function based on the quality of re-projection. The data already existing in the light field provides the required ground truth, thus optimizing for the quality of the the resulting image rather than the depth. A promising future direction for our approach would be to use a similar loss function, but to combine it with the high quality ground truth depth we have at our disposal, especially for thin structures and hard-to-reconstruct objects.

Even though our experiments are incomplete, we believe that such learning approaches based on synthetic data hold great promise for the future.

CHAPTER 6

# Conclusions and Further Work

Contents



The goal of the thesis was to develop solutions that allow IBR to become more usable, making it possible to adopt this approach in much wider settings. To achieve this goal we presented several novel solutions to improve speed and quality of Image-based Rendering under incomplete and inaccurate geometric information. Specifically, we investigated three research directions to address these issues.

## 6.1 Contributions

At the beginning of this thesis work, the highest-quality free-viewpoint IBR algorithm for outdoors scenes involved expensive warp operations on superpixels Chaurasia *et al.* (2013), making the method too expensive for many uses, e.g., on mobile platforms. To address this problem we developed and implement an approach based on Bayesian principles. We model the rendering quality, the rendering process and the validity of the assumptions of a set of IBR algorithms. Our method improves the quality/speed trade-off of the input algorithms, selecting the best rendering algorithm for each region (represented by a superpixel segment) in a preprocessing step. At runtime our selective IBR uses this choice to achieve significant speedup at equivalent or better quality compared to previous algorithms.

A recurring problem in IBR algorithms is rendering of shiny objects. Even though solving the general problem is very hard, we observed that the recent availability of models for certain classes of objects (such as cars), opens an opportunity to exploit existing 3D models to improve the quality of IBR. We thus introduced a new mixed Image-Based Rendering algorithm that builds on recent advances in object detection and recognition. Our system automatically augments IBR scenes with explicit geometry of a semantic class present in the images. The geometry is obtained from publicly available databases of 3D CAD models. As a test case, we use the class “car” which is often present in street views and which are usually poorly reconstructed due the transparency and reflectivity of such objects. Our pipeline starts by detecting the object of interests and then accurately places the object in the scene using a multi-view approach taking advantage of available geometry

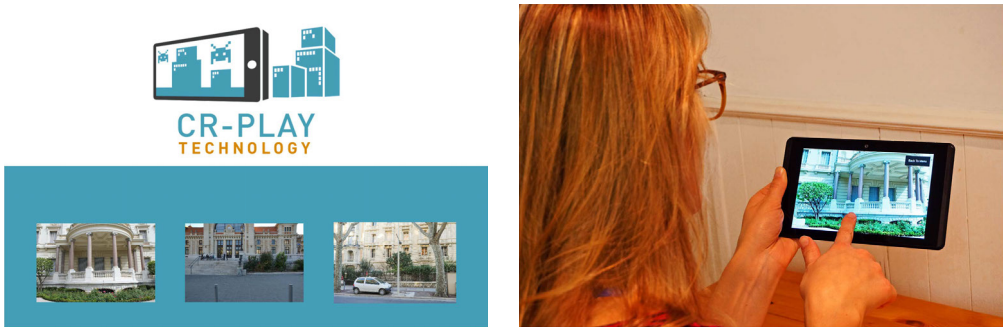



Figure 6.1: **Our Selective rendering running on a mobile device.** Left: Screen-shots of some demos of IBR scenes. Right: IBR scene running on a mobile device.

and silhouettes. Since the retrieved CAD model might not be the exactly the same as the actual captured object by our photographs, the 3D mesh is morphed to better fit the images contours. Our results demonstrate that we obtain improved rendering quality even when moving away from the input cameras.

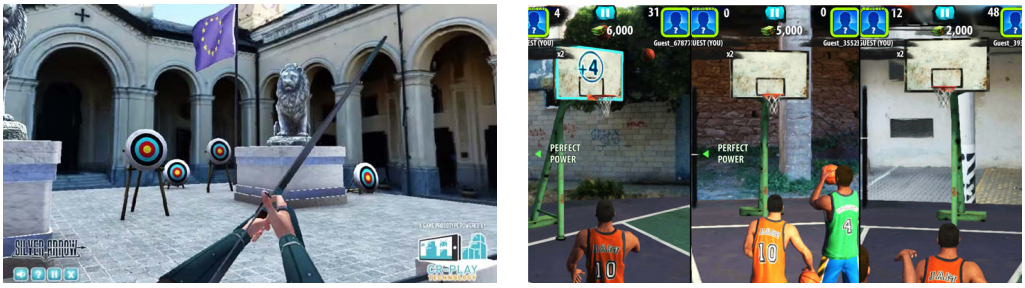
Another significant problem in IBR for outdoors scenes is the inaccuracy of the 3D reconstruction using SfM and different MVS methods. Specifically, such methods often miss some parts of the geometry, or create incorrect additional geometry in other places. We reported initial experiments addressing these issues, first based on a hand-crafted descriptor and then based on the use of realistic synthetic data to train a deep network for accurate depth prediction.

## 6.2 Research Impact and Applications

We addressed the problems of IBR in three different projects. The results of Chapter 3 were presented at the International Conference on 3D Vision (3DV) 2015 (oral) and those of Chapter 4 at 3DV 2016 (oral). The research reported in Chapter 5 is still preliminary, but shows much promise, in particular in the possible use of synthetic data for deep learning.

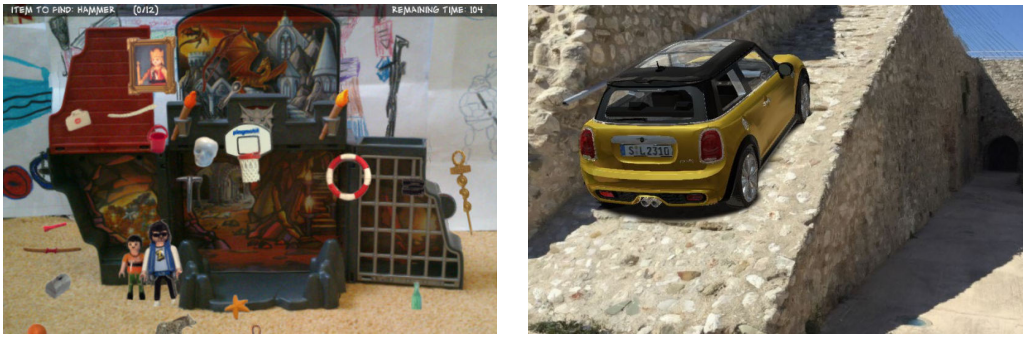
This thesis was funded by the EU project CR-PLAY (www.cr-play.eu), which was coordinated by a game developer company (*Testaluna/Miniclip*) and included the *TU Darmstadt* who provided an SfM and MVS solution. The results developed in this thesis had significant direct impact in the evolution and success of the project. The increase in rendering speed provided by the solution of Chapter 3 was critical in allowing the rendering algorithm to be used on mobile platforms, via a Unity3D port of our approach developed together with the game-developer partners of the project.

Two different prototypes games were developed using IBR technologies (in Fig. 6.2). The first one, name *SilverArrow* is a first-person-shotting game where an archer attempts to hit targets on a scene. The movement of the character is also restricted to avoid having the camera go outside the capture field. The background of the scene is rendered with our *Selective Rendering*. The second prototype is a street basketball game. The game-play happens in multiple urban scenarios give the ease capture of Image-based Modeled assets.




(a) (b)

Figure 6.2: **CR-PLAY IBR prototypes games.** (a) Silver Arrow. (b) IBR Basketball




(a) (b)

Figure 6.3: **Some prototypes games developed during the CR-PLAY workshop.** (a) Survive the weekend. (b) Rising car.

To engage game developers with the Image-based Rendering technology CR-PLAY organized a workshop in Patras that required the development of game prototypes through the whole process (capture, reconstruct, edit-and-play). More than 10 game developers participated creating prototypes games with image-based content and rendering. The game developers were able to use the system, including SfM and MVS reconstruction provided by TUD, and our selective IBR prepFig. 6.3. The quality and speed of the rendering was highly commended by the users, who all stated that they would like to be able to use the algorithms commercially.

## 6.3 Future Directions

The results we presented have advanced both the speed and the quality of IBR algorithms, and have also had impact in an applied setting. The ideas we have explored in this thesis open numerous possible future research directions. We list a few of these in what follows.

**Using 3D models for improved IBR.** The algorithm we presented in Chapter 4 is generic and can be directly applied on other object categories. We would like to test our approach with other classes, however the effectiveness of such solutions will depend on the availability of a sufficient large and diverse number of models for the given category.

We see our approach as a first step in a more general trend, in which traditional 3D models will be combined with image-based techniques to greatly simplify 3D content creation and interactive display. Additionally, we would like to incorporate the semantic knowledge of new databases as *ObjectNet3D* (Xiang *et al.*, 2016) to render retrieved objects based, for example, on their material properties.

**Reflections in IBR.** In order to render new viewpoints, current IBR techniques use approximate geometry to warp and blend colors from close viewpoints. They assume the scene materials are diffuse so geometry colors are independent of the viewpoint, which fails in the case of specular surfaces such as windows. Dealing with reflections in an IBR context requires being able to identify what are the diffuse and the specular color layers in the input images. Importantly, it requires a method to correctly warp the specular layers since their associated geometry is not available and since the normals of the reflective surfaces are not reliable. Our solution for placing an exact synthetic model to replace such objects could potentially allow the use of these high quality models to develop accurate solutions for reflective surface IBR.

**IBR for Head-Mounted Displays (HMD).** The improved quality and speed of our IBR algorithms has opened the possibility of their use in an HMD setting. In the context of CR-PLAY, and initial demonstrator using our IBR algorithm was created for an HMD. While the potential of such solutions was clear from the prototype, many problems need to be solved before this solution can be widely used. In particular, we need to improve poor depth perception in IBR for HMDs by alleviating the card-boarding effect and also to improve comfort by reducing visual fatigue. These can be done by matching the capturing, projection and display geometry, managing the Vergence - Accommodation conflict and rendering a Depth-of-Field effect consistent with the scene. Another interesting idea is to render content appropriately in order to match eye adaptation to light by properly rendering High Dynamic Range captures in IBR for HMDs. The increase in speed and quality resulting from the algorithms in this thesis are prerequisite for this type of future work.

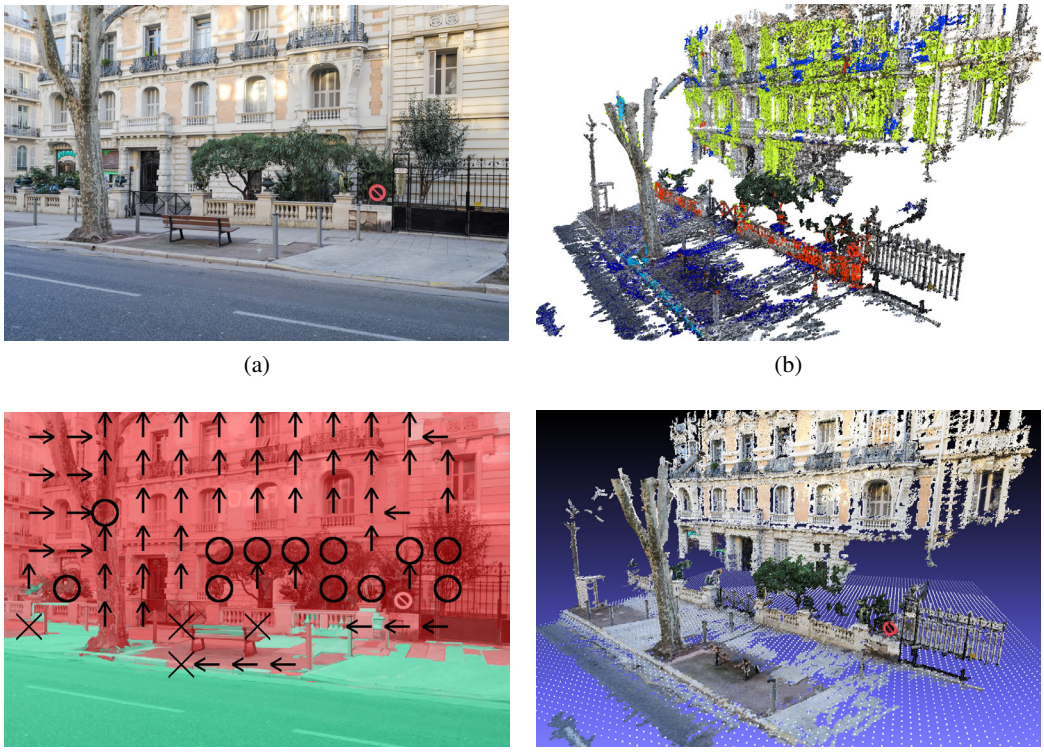
Appendix A

# Floor Plane Estimation

The floor in outdoors scenes is a particularly difficult case for reconstruction with MVS algorithms due to the fact that it is texture-less and the grazing angle of capture. We could augment urban scene reconstruction fitting a plane to the floor $\pi_{floor}$. We developed a first solution to this problem. With RANSAC we estimate principal planes in the point cloud. In Fig. A.1(b) we observe clusters of colors, corresponding to the main planes of the scene Fig. A.1(a). To select which of the planes represents the floor, we can simply define a heuristic based on location of planes in the images or use more elaborated machinery such as the *Geometric Context* proposed by Hoiem *et al.* (2005) where we can recover the surface layout. In Fig. A.1(c) we can see the detected ground surface in saturated green. With this we can select the 3D samples that generate the floor plane and extend it as in Fig.A.1(c).

Now, the problem consist to assign superpixels to the floor plane (floor label). We attempted to do so by casting the labeling problem as a MRF over the superpixels $S^i$ (see Eq.A.1). For each superpixel with MVS points, we fit a local plane following the procedure described in Section 3.3.2 and Section 3.3.2.1. The term $E_g$ encourages to assign the floor label to superpixels with a similar orientation to $\pi_{floor}$. We initialize global appearance model for the floor based on a Gaussian mixture model with the color of the superpixels with MVS points used to estimate $\pi_{floor}$. The term $E_a$ encourages superpixels to be assigned to the floor if they have a similar appearance. Sometimes the appearance of the floor cannot be separated from the poles or background. We add the term $E_p$ to regularize the solution with user scribbles. The user sparsely defines regions that should belong to the floor and elsewhere regions. We solve this with graph cut and propagate the results to other input images (see Fig. A.2).

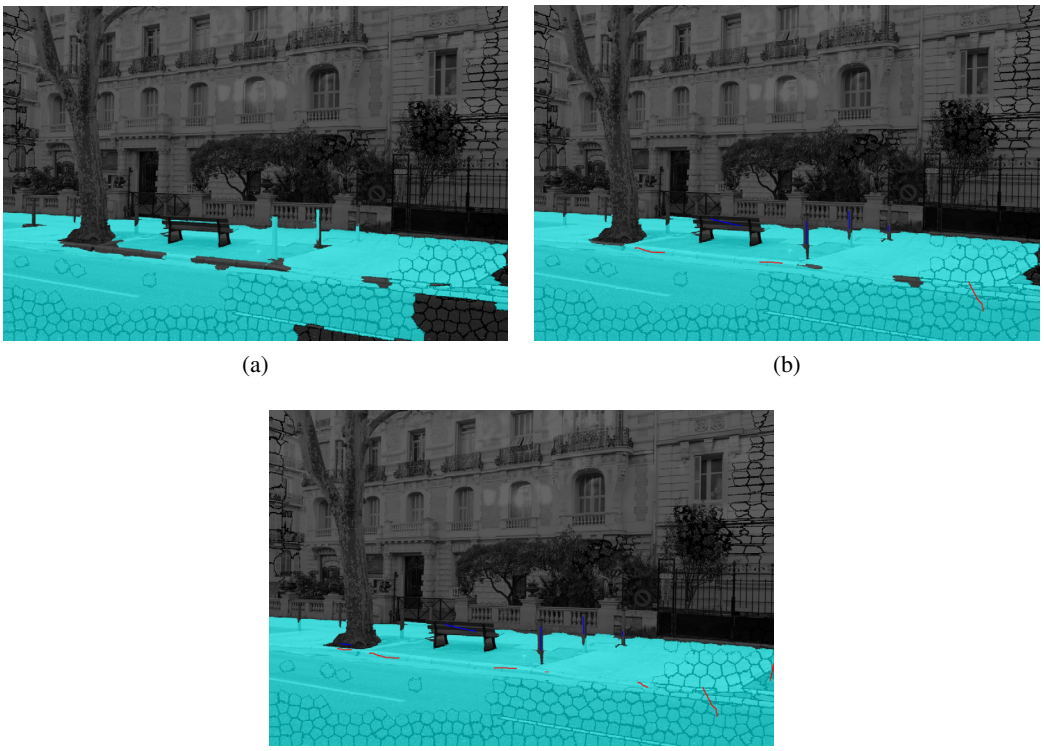
$$\sum E_g(S^i, l^i) + \sum E_a(S^i, l^i) + \sum E_p(S^i, l^i) \tag{A.1}$$



(a) (b)

(c) (d)

Figure A.1: **3D floor plane estimation.** (a) One of the input images of the scene. (b) Clusters of 3D points that belongs to different planes. (c) Geometric context layout (Hoiem *et al.*, 2005). Saturated in green is the ground floor, other planes in red. Vertical arrows represent a vertical surface while horizontal arrows represent a surface facing either left or right. Circles are porous regions like vegetation. (d) Recovered and extended floor plane.



(a) (b)

(c)

Figure A.2: **Assign superpixels to the floor plane.** Contours of superpixels with no depth marked. (a) Automatic results leave some superpixels of the sidewalk out of the floor plane but also mark some vertical poles as floor. (b) After one iteration, we can refine the result with strokes. (c) With two iterations we completely separate the floor from other elements. We propagate this result to the other input views.

Appendix B

# Derivatives for Automatic Pose Estimation

## B.1 Derivatives for Initial Pose Estimation

In Chapter 4, we presented a solution to the problem of setting an initial translation and scale. This can be represented as an energy minimization problem:

$$\Lambda^* = \arg\min_{\Lambda} E_{init\_align}(\Lambda) = \sum_{i=1}^{n} dist(x_1^i, d_1^i)^2 + dist(x_2^i, d_2^i)^2 \tag{B.1}$$

where $\{x_1, x_2\}$ and $\{d_1^i, d_2^i\}$ respectively define, for view $i$, the sets of upper left and bottom right corners of the 2D bounding boxes of the object and the detection. The distance function $dist$ is defined as

$$dist(x, x^d) = \begin{cases} 0 & \text{if } x \text{ or } x^d \text{ on the image border} \\ \|x - x^d\|_2 & \text{otherwise} \end{cases} \tag{B.2}$$

We use gradient descent to estimate these initial set of parameters $\Lambda \in \{\lambda | t_x, t_y, t_z, s\}$

$$\frac{\partial E_{init\_align}}{\partial \lambda} = \sum_{i=1}^{n} 2(x_1^i - d_1^i)\frac{\partial x_1^i}{\partial \lambda} + 2(x_2^i - d_2^i)\frac{\partial x_2^i}{\partial \lambda} \tag{B.3}$$

An example of derivative term:

$$\frac{\partial x^i}{\partial \lambda} = \frac{\partial x^i}{\partial x}\frac{\partial x}{\partial \lambda} + \frac{\partial x^i}{\partial y}\frac{\partial y}{\partial \lambda} \tag{B.4}$$

Remark that $x^i$ depends on the $(x, y)$ position. Thus $\frac{\partial x^i}{\partial x}$ and $\frac{\partial x^i}{\partial y}$ are equal to 1. In section B.3 we develop the term $\frac{\partial x}{\partial \lambda}$ and by extension $\frac{\partial y}{\partial \lambda}$.

## B.2 Derivatives for Multi-View Alignment

We follow the same strategy as related work in model alignment Dambreville *et al.* (2008b); Prisacariu & Reid (2012) where the derivatives of contour points are estimated using 3D points projecting on the contours. We use both closest and farthest 3D points for the following optimization problem:

$$\Lambda^* = \arg\min_{\Lambda} E_{fine\_align}(\Lambda) \tag{B.5}$$

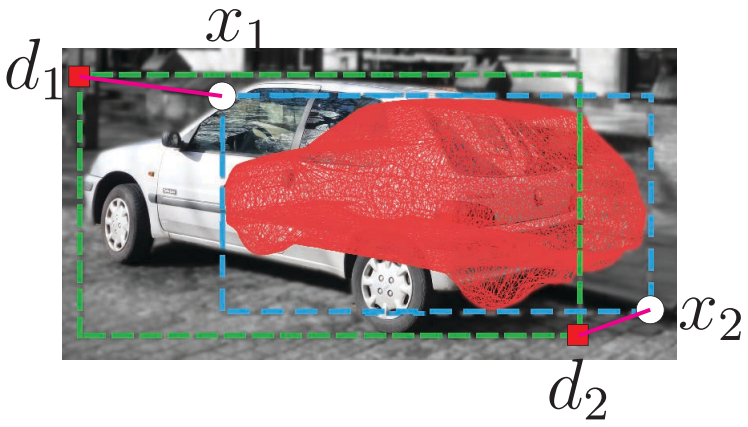



Figure B.1: **Constraints from detection of 2D boxes**. To have a better starting point for object alignment, we use the constraints from the detection bounding boxes. The bounding box corners $x_1$ and $x_2$ of the mesh should match the corners $d_1$ and $d_2$ of the R-CNN detection box.

Where:

$$E_{fine\_align}(\Lambda) = E_{3D}(\Lambda) + E_{edge}(\Lambda) \tag{B.6}$$

$$E_{3D}(\Lambda) = \sum_{i=1}^{n} \sum_{(v,v_{\text{mvs}})\in C^i} \|v - v_{\text{mvs}}\|^2 \tag{B.7}$$

$$E_{edge}(\Lambda) = \sum_{i=1}^{n} \sum_{(p,p')\in \mathcal{M}^i} \|p - p'\|^2 \tag{B.8}$$

$C^i$ is the set of valid $(v, v_{\text{mvs}})$ pairs obtained from view $i$. The set $\mathcal{M}^i$ contains the 2D matching points $(p, p')$ from silhouettes in view $i$. To solve equation B.5 we use gradient descent to estimate the set of parameters $\Lambda \in \{\lambda | t_x, t_y, t_z, q_x, q_y, q_z, q_w, s\}$:

$$\frac{\partial E_{fine\_align}}{\partial \lambda} = \frac{\partial E_{3D}}{\partial \lambda} + \frac{\partial E_{edge}}{\partial \lambda} \tag{B.9}$$

$$\frac{\partial E_{3D}}{\partial \lambda} = 2 \sum_{i=1}^{n} \sum_{(v,v_{\text{mvs}})\in C^i} (v - v_{\text{mvs}}) \cdot \left( \frac{\partial X}{\partial \lambda}, \frac{\partial Y}{\partial \lambda}, \frac{\partial Z}{\partial \lambda} \right)$$

$$\frac{\partial E_{edge}}{\partial \lambda} = 2 \sum_{i=1}^{n} \sum_{(p,p')\in \mathcal{M}^i} (p - p') \cdot \left( \frac{\partial x}{\partial \lambda}, \frac{\partial y}{\partial \lambda} \right)$$

Derivatives $\left(\frac{\partial X}{\partial \lambda}, \frac{\partial Y}{\partial \lambda}, \frac{\partial Z}{\partial \lambda}\right) r$ can be found in Table B.1 and derivatives $\left(\frac{\partial x}{\partial \lambda}, \frac{\partial y}{\partial \lambda}\right)$ can be found in section B.3

## B.3 Common Derivatives

In this section we present derivatives common to B.1 and B.2. Derivatives of 2D image projections with respect to pose parameters:

$$\frac{\partial x}{\partial \lambda} = \frac{\partial x}{\partial X}\frac{\partial X}{\partial \lambda} + \frac{\partial x}{\partial Y}\frac{\partial Y}{\partial \lambda} + \frac{\partial x}{\partial Z}\frac{\partial Z}{\partial \lambda} \tag{B.10}$$

$$\frac{\partial y}{\partial \lambda} = \frac{\partial y}{\partial X}\frac{\partial X}{\partial \lambda} + \frac{\partial y}{\partial Y}\frac{\partial Y}{\partial \lambda} + \frac{\partial y}{\partial Z}\frac{\partial Z}{\partial \lambda} \tag{B.11}$$

We can obtain a 3D point position in homogenious coordinates $W = (X, Y, Z, 1)^T$ from $(X_0, Y_0, Z_0)$ applying transformations (translation $t$, rotation $R$, scale $s$) given by the pose parameters $\Lambda$ (see equation B.12). $(X_0, Y_0, Z_0)$ represents the initial position of a 3D point of the mesh.

$$W = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = sR \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} + t \tag{B.12}$$

Where

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

We can relate $W$ to the 2D point $(x, y)$ in image coordinates using the camera $P$ as:

$$\begin{bmatrix} x & y & 1 \end{bmatrix}^T = P \cdot W^T \tag{B.13}$$

Where $P = \begin{bmatrix} P_1 & P_2 & P_3 \end{bmatrix}^T$ and $P_i = \begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \end{bmatrix}$. Thus $x$ and $y$ in terms of $(X, Y, Z)$ are:

$$x = \frac{P_1 \cdot W}{P_3 \cdot W} \qquad y = \frac{P_2 \cdot W}{P_3 \cdot W} \tag{B.14}$$

We derive equations B.14 w.r.t. $X$, $Y$ and $Z$ to obtain some of the terms of equations B.10 and B.11:

$$\frac{\partial x}{\partial X} = \frac{p_{11}P_3 \cdot W - p_{31}P_1 \cdot W}{(P_3 \cdot W)^2}$$

$$\frac{\partial x}{\partial Y} = \frac{p_{12}P_3 \cdot W - p_{32}P_1 \cdot W}{(P_3 \cdot W)^2}$$

$$\frac{\partial x}{\partial Z} = \frac{p_{13}P_3 \cdot W - p_{33}P_1 \cdot W}{(P_3 \cdot W)^2}$$

$$\frac{\partial y}{\partial X} = \frac{p_{21}P_3 \cdot W - p_{31}P_2 \cdot W}{(P_3 \cdot W)^2}$$

| $\lambda$ | $\frac{\partial X}{\partial \lambda}$ | $\frac{\partial Y}{\partial \lambda}$ | $\frac{\partial Z}{\partial \lambda}$ |
|---|---|---|---|
| $t_x$ | 1 | 0 | 0 |
| $t_y$ | 0 | 1 | 0 |
| $t_z$ | 0 | 0 | 1 |
| $q_x$ | $2q_yY_0 + 2q_zZ_0$ | $2q_yX_0 - 4q_xY_0 - 2q_wZ_0$ | $2q_zX_0 + 2q_wY_0 - 4q_xZ_0$ |
| $q_y$ | $2q_xY_0 - 4q_yX_0 + 2q_wZ_0$ | $2q_xX_0 + 2q_zZ_0$ | $2q_zY_0 - 2q_wX_0 - 4q_yZ_0$ |
| $q_z$ | $2q_xZ_0 - 2q_wY_0 - 4q_zX_0$ | $2q_wX_0 - 4q_xY_0 + 2q_yZ_0$ | $2q_xX_0 + 2q_yY_0$ |
| $q_w$ | $2q_yZ_0 - 2q_zY_0$ | $2q_zX_0 - 2q_xZ_0$ | $2q_xY_0 - 2q_yX_0$ |
| $s$ | $r_{11}X_0 + r_{12}Y_0 + r_{13}Z_0$ | $r_{21}X_0 + r_{22}Y_0 + r_{23}Z_0$ | $r_{31}X_0 + r_{32}Y_0 + r_{33}Z_0$ |

Table B.1: Derivatives of 3D point position with respect to pose parameters.

$$\frac{\partial y}{\partial Y} = \frac{p_{22}P_3 \cdot W - p_{32}P_2 \cdot W}{(P_3 \cdot W)^2}$$

$$\frac{\partial y}{\partial Z} = \frac{p_{33}P_3 \cdot W - p_{33}P_2 \cdot W}{(P_3 \cdot W)^2}$$

Other derivatives of equations are presented in Table B.1.

Appendix C

# Luminance Harmonization

At render time, IBR algorithms directly use content from multiple images to render novel views. However, color consistency is not always guarantee during the acquisition of images even when efforts are made to block the camera exposure parameters. This inconsistency produces blending artifacts. We implement an image harmonization procedure that consists in simply applying a global transformation $M_i$ from the input image $I_i$ color space to an output image $\tilde{I}_i$ in a median color space $\tilde{I}_i = M_i\, I_i$

The median color space $Y_i$ is built with the multi-view color information of the reconstructed points in $I_i$. The parameters of $M_i$ are found by solving $X_i$ in the linear system $Y_i = A_i X_i$, where $A_i$ contains the color of reconstructed pixels in $I_i$. This operation is performed independently for each color channel, resulting in the new set of images $\tilde{I}_i$ which have harmonized colors. We then use $\tilde{I}_i$ as input images instead of $I_i$ for the IBR algorithm, avoiding significant visual artefacts related to blending of inconsistent colors.

In the Fig.C.1 we see an example of the application of the harmonization approach, and the case of the red pixel and the differences between the two views before and after harmonization. In the original dataset, differences in RGB values were up to 14 color levels (average 11), while after harmonization, the maximum difference is 7 and the average is 5.6: we have reduced the difference by half. Even though the effect is subtle in the image shown above, the difference becomes much clearer during rendering where these image are blended. This can clearly be seen below, where we see that the severe artefacts in the sky on the right are greatly diminished after harmonization. The current method is restricted to a single transformation for the entire image, which has proven sufficient in some cases, but may need to be refined in more difficult cases. For these a per-region (e.g., superpixel) approach may be appropriate, followed by a filtering step (i.e., bi-lateral filtering), similar to the approach developed in Okura *et al.* (2015).


Figure C.1: **Example of color harmonization.**

# Bibliography


Achanta, Radhakrishna, Shaji, Appu, Smith, Kevin, Lucchi, Aurelien, Fua, Pascal, & Süsstrunk, Sabine. 2010. *Slic superpixels*. Tech. rept. (Cited on pages 15 and 42.)

Achanta, Radhakrishna, Shaji, Appu, Smith, Kevin, Lucchi, Aurelien, Fua, Pascal, & Süsstrunk, Sabine. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, **34**(11), 2274–2282. (Cited on pages 15, 22 and 84.)

Adelson, Edward H, & Bergen, James R. 1991. *The plenoptic function and the elements of early vision*. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology. (Cited on page 10.)

Alexa, Marc, Cohen-Or, Daniel, & Levin, David. 2000. As-rigid-as-possible shape interpolation. *Pages 157–164 of: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. (Cited on page 22.)

Arandjelović, Relja, & Zisserman, Andrew. 2011. Smooth object retrieval using a bag of boundaries. *Pages 375–382 of: 2011 International Conference on Computer Vision*. IEEE. (Cited on page 62.)

Argiles, Alberto, Civera, Javier, & Montesano, Luis. 2011. Dense multi-planar scene estimation from a sparse set of images. *Pages 4448–4454 of: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. (Cited on page 33.)

Aubry, Mathieu, & Russell, Bryan C. 2015. Understanding deep features with computer-generated imagery. *Pages 2875–2883 of: Proceedings of the IEEE International Conference on Computer Vision*. (Cited on pages 62 and 63.)

Aubry, Mathieu, Russell, Bryan C, & Sivic, Josef. 2014a. Painting-to-3D model alignment via discriminative visual elements. *ACM Transactions on Graphics (TOG)*, **33**(2), 14. (Cited on page 86.)

Aubry, Mathieu, Maturana, Daniel, Efros, Alexei A, Russell, Bryan C, & Sivic, Josef. 2014b. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. *Pages 3762–3769 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 62.)

Bay, Herbert, Tuytelaars, Tinne, & Van Gool, Luc. 2006. Surf: Speeded up robust features. *Pages 404–417 of: European conference on computer vision*. Springer. (Cited on page 12.)

Bishop, Christopher M, *et al.* 2006. *Pattern recognition and machine learning*. Vol. 4. springer New York. (Cited on pages 30 and 40.)

Bodis-Szomoru, Andras, Riemenschneider, Hayko, & Gool, Luc Van. 2014. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. *In: CVPR*. (Cited on pages 29, 33, 34 and 35.)

Boykov, Yuri, Veksler, Olga, & Zabih, Ramin. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, **23**(11), 1222–1239. (Cited on page 85.)

Buehler, Chris, Bosse, Michael, McMillan, Leonard, & Cohen, Steven Gortlerand Michael. 2001. Unstructured lumigraph rendering. *In: SIGGRAPH*. (Cited on pages 17, 18, 28, 42, 65, 73, 74 and 77.)

Čadík, Martin, Herzog, Robert, Mantiuk, Rafał, Myszkowski, Karol, & Seidel, Hans-Peter. 2012. New measurements reveal weaknesses of image quality metrics in evaluating graphics artifacts. *ACM Transactions on Graphics (TOG)*, **31**(6), 147. (Cited on page 50.)

Canny, John. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 679–698. (Cited on page 70.)

Chai, Jin-Xiang, Tong, Xin, Chan, Shing-Chow, & Shum, Heung-Yeung. 2000. Plenoptic sampling. *Pages 307–318 of: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. (Cited on page 10.)

Chang, Angel X., Funkhouser, Thomas A., Guibas, Leonidas J., Hanrahan, Pat, Huang, Qi-Xing, Li, Zimo, Savarese, Silvio, Savva, Manolis, Song, Shuran, Su, Hao, Xiao, Jianxiong, Yi, Li, & Yu, Fisher. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR*, **abs/1512.03012**. (Cited on pages 60, 62, 64 and 66.)

Chaurasia, Gaurav, Sorkine, Olga, & Drettakis, George. 2011. Silhouette-Aware Warping for Image-Based Rendering. *Comp. Graph. Forum*. (Cited on pages 21, 22, 43 and 46.)

Chaurasia, Gaurav, Duchene, Sylvain, Sorkine-Hornung, Olga, & Drettakis, George. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. on Graphics (TOG)*. (Cited on pages 4, 17, 21, 22, 23, 28, 29, 30, 33, 36, 42, 43, 44, 45, 46, 51, 60, 74, 80 and 95.)

Chen, Shenchang Eric. 1995. Quicktime VR: An image-based approach to virtual environment navigation. *Pages 29–38 of: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. (Cited on page 11.)

Chen, Shenchang Eric, & Williams, Lance. 1993. View interpolation for image synthesis. *Pages 279–288 of: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. (Cited on pages 10, 11 and 12.)

Chen, Tao, Zhu, Zhe, Shamir, Ariel, Hu, Shi-Min, & Cohen-Or, Daniel. 2013. 3-Sweep: extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, **32**(6), 195. (Cited on page 64.)

Cremers, Daniel, Rousson, Mikael, & Deriche, Rachid. 2007. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International journal of computer vision*, **72**(2), 195–215. (Cited on page 71.)

Dalal, Navneet, & Triggs, Bill. 2005. Histograms of Oriented Gradients for Human Detection. *In: CVPR*. (Cited on page 62.)

Dambreville, Samuel, Rathi, Yogesh, & Tannenbaum, Allen. 2008a. A framework for image segmentation using shape models and kernel space shape priors. *IEEE transactions on pattern analysis and machine intelligence*, **30**(8), 1385–1399. (Cited on page 63.)

Dambreville, Samuel, Sandhu, Romeil, Yezzi, Anthony, & Tannenbaum, Allen. 2008b. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. *Pages 169–182 of: European Conference on Computer Vision*. Springer. (Cited on pages 63 and 103.)

Debevec, Paul, Yu, Yizhou, & Borshukov, George. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. *In: Rendering Techniques 98*. Springer. (Cited on pages 11, 16, 17, 29, 30 and 36.)

Debevec, Paul E., Taylor, Camillo J., & Malik, Jitendra. 1996. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. *In: SIGGRAPH*. (Cited on pages 11, 12, 18 and 28.)

Eigen, David, & Fergus, Rob. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Pages 2650–2658 of: Proceedings of the IEEE International Conference on Computer Vision*. (Cited on pages 86, 89 and 90.)

Eigen, David, Puhrsch, Christian, & Fergus, Rob. 2014. Depth map prediction from a single image using a multi-scale deep network. *Pages 2366–2374 of: Advances in neural information processing systems*. (Cited on page 86.)

Eisemann, Martin, Decker, Bert De, Magnor, Marcus, PhilippeBekaert, de Aguiar, Edilson, Ahmed, Naveed, & Sellent, Christian Theobaltand Anita. 2008. Floating Textures. *Comp. Graph. Forum*. (Cited on pages 4, 19, 20, 21, 28 and 45.)

Ferryman, James M, Worrall, Anthony D, Sullivan, Geoffrey D, Baker, Keith D, *et al.* 1995. A Generic Deformable Model for Vehicle Recognition. *Page 2 of: BMVC*, vol. 1. Citeseer. (Cited on page 63.)

Fischler, Martin A, & Bolles, Robert C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**(6), 381–395. (Cited on page 33.)

Fitzgibbon, Andrew, Wexler, Yonatan, & Zisserman, Andrew. 2005. Image-based rendering using image-based priors. *IJCV*. (Cited on pages 24 and 28.)

Flynn, John, Neulander, Ivan, Philbin, James, & Snavely, Noah. 2015. DeepStereo: Learning to Predict New Views from the World's Imagery. *arXiv preprint arXiv:1506.06825*. (Cited on pages 24, 25, 45 and 56.)

Fuhrmann, Simon, & Goesele, Michael. 2014. Floating scale surface reconstruction. *ACM Transactions on Graphics (TOG)*, **33**(4), 46. (Cited on page 14.)

Furukawa, Yasutaka, & Ponce, Jean. 2007. Accurate, Dense, and Robust Multi-View Stereopsis. *In: CVPR*. (Cited on pages 14, 15, 43 and 47.)

Furukawa, Yasutaka, & Ponce, Jean. 2010. Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Trans. PAMI*. (Cited on pages 13, 14 and 41.)

Furukawa, Yasutaka, Curless, Brian, Seitz, Steven M, & Szeliski, Richard. 2009. Manhattan-world stereo. *Pages 1422–1429 of: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. (Cited on page 16.)

Gallup, D., Frahm, J.-M., & Pollefeys, M. 2010. Piecewise planar and non-planar stereo for urban scene reconstruction. *In: CVPR*. (Cited on page 16.)

Gidaris, Spyros, & Komodakis, Nikos. 2015. Object detection via a multi-region & semantic segmentation-aware CNN model. *CoRR*, **abs/1505.01749**. (Cited on pages 62, 63, 64, 65 and 66.)

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, & Malik, Jitendra. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *In: CVPR*. (Cited on pages 62 and 63.)

Godard, Clément, Mac Aodha, Oisin, & Brostow, Gabriel J. 2016. Unsupervised Monocular Depth Estimation with Left-Right Consistency. *arXiv preprint arXiv:1609.03677*. (Cited on page 93.)

Goesele, Michael, Snavely, Noah, Curless, Brian, Hoppe, Hugues, & Seitz, Steven M. 2007. Multi-view stereo for community photo collections. *In: ICCV*. (Cited on pages 13, 14, 15, 43, 47 and 69.)

Goesele, Michael, Ackermann, Jens, Fuhrmann, Simon, Haubold, Carsten, & Klowsky, Ronny. 2010. Ambient point clouds for view interpolation. *ACM Trans. Graph.* (Cited on pages 19, 21, 28 and 46.)

Gortler, Steven J., Grzeszczuk, Radek, Szeliski, Richard, & Cohen, Michael F. 1996. The Lumigraph. *In: SIGGRAPH*. (Cited on pages 10, 11 and 28.)

Guney, Fatma, & Geiger, Andreas. 2015. Displets: Resolving stereo ambiguities using object knowledge. *Pages 4165–4175 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 73.)

Hedman, Peter, Ritschel, Tobias, Drettakis, George, & Brostow, Gabrielr. 2016. Scalable Inside-Out Image-Based Rendering. **35**(6), to appear. (Cited on pages 51 and 80.)

Heigl, Benno, Koch, Reinhard, Pollefeys, Marc, Denzler, Joachim, & Van Gool, Luc. 1999. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. *Pages 94–101 of: Mustererkennung 1999*. Springer. (Cited on page 17.)

Hödlmoser, Michael, Micusik, Branislav, Pollefeys, Marc, Liu, Ming-Yu, & Kampel, Martin. 2013. Model-based vehicle pose estimation and tracking in videos using random forests. *Pages 430–437 of: 2013 International Conference on 3D Vision-3DV 2013*. IEEE. (Cited on page 63.)

Hoiem, Derek, Efros, Alexei A, & Hebert, Martial. 2005. Geometric context from a single image. *Pages 654–661 of: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE. (Cited on pages 99 and 100.)

Hueting, Moos, Ovsjanikov, Maks, & Mitra, Niloy J. 2015. CrossLink: joint understanding of image and 3D model collections through shape and camera pose variations. *ACM Transactions on Graphics (TOG)*, **34**(6), 233. (Cited on page 62.)

Huttenlocher, D. P., & Ullman, S. 1987. Object recognition using alignment. *In: ICCV*. (Cited on page 62.)

Isola, Phillip, Zoran, Daniel, Krishnan, Dilip, & Adelson, Edward H. 2014. Crisp Boundary Detection Using Pointwise Mutual Information. *In: ECCV*. (Cited on page 70.)

Jacobson, Alec, Panozzo, Daniele, *et al.* 2016. *libigl: A simple C++ geometry processing library*. http://libigl.github.io/libigl/. (Cited on page 73.)

Jancosek, Michal, & Pajdla, Tomás. 2011. Multi-view reconstruction preserving weakly-supported surfaces. *Pages 3121–3128 of: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. (Cited on pages 13, 14, 15, 41, 44, 47, 64, 65, 69, 80 and 88.)

Kalantari, Nima Khademi, Wang, Ting-Chun, & Ramamoorthi, Ravi. 2016. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, **35**(6). (Cited on page 93.)

Kazhdan, Michael, Bolitho, Matthew, & Hoppe, Hugues. 2006. Poisson surface reconstruction. *In: Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7. (Cited on page 14.)

Kholgade, Natasha, Simon, Tomas, Efros, Alexei, & Sheikh, Yaser. 2014. 3D object manipulation in a single photograph using stock 3D models. *ACM Transactions on Graphics (TOG)*, **33**(4), 127. (Cited on pages 60, 64 and 72.)

Klose, Felix, Ruhl, Kai, Lipski, Christian, & Magnor, Marcus. 2011. Flowlab-an interactive tool for editing dense image correspondences. *Pages 59–66 of: Visual Media Production (CVMP), 2011 Conference for*. IEEE. (Cited on page 17.)

Koch, Reinhard, Heigl, Benno, & Pollefeys, Marc. 2001. Image-based rendering from uncalibrated lightfields with scalable geometry. *Pages 51–66 of: Multi-Image Analysis*. Springer. (Cited on pages 17 and 18.)

Koller, Dieter. 1993. Moving object recognition and classification based on recursive shape parameter estimation. *In: Proc. 12th Israel Conf. Artificial Intelligence, Computer Vision*, vol. 2728. (Cited on page 63.)

Kopf, Johannes, Langguth, Fabian, Scharstein, Daniel, Szeliski, Richard, & Goesele, Michael. 2013. Image-based rendering in the gradient domain. *ACM Trans. Graph.* (Cited on pages 23, 24 and 28.)

Kopf, Johannes, Cohen, Michael F, & Szeliski, Richard. 2014. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)*, **33**(4), 78. (Cited on page 24.)

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey E. 2012a. Imagenet classification with deep convolutional neural networks. *Pages 1097–1105 of: Advances in neural information processing systems*. (Cited on pages 62 and 66.)

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey E. 2012b. Imagenet classification with deep convolutional neural networks. *In: NIPS*. (Cited on pages 86 and 89.)

Kutulakos, Kiriakos N, & Seitz, Steven M. 2000. A theory of shape by space carving. *International Journal of Computer Vision*, **38**(3), 199–218. (Cited on page 14.)

Lafarge, Florent, Keriven, Renaud, & Brédif, Mathieu. 2010. Insertion of 3-D-primitives in mesh-based representations: towards compact models preserving the details. *IEEE Transactions on Image Processing*, **19**(7), 1683–1694. (Cited on page 63.)

Lafarge, Florent, Keriven, Renaud, Brédif, Mathieu, & Vu, Hoang-Hiep. 2013. A hybrid multiview stereo algorithm for modeling urban scenes. *IEEE transactions on pattern analysis and machine intelligence*, **35**(1), 5–17. (Cited on page 63.)

Laveau, Stephane, & Faugeras, Olivier D. 1994. 3-D scene representation as a collection of images. *Pages 689–691 of: Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE. (Cited on page 10.)

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, & Haffner, Patrick. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324. (Cited on page 62.)

Lee, Jungjin, Kim, Younghui, Lee, Sangwoo, Kim, Bumki, & Noh, Junyong. 2015. High-quality depth estimation using an exemplar 3d model for stereo conversion. *IEEE transactions on visualization and computer graphics*, **21**(7), 835–847. (Cited on pages 60 and 64.)

Leotta, Matthew J, & Mundy, Joseph L. 2011. Vehicle surveillance with a generic, adaptive, 3d vehicle model. *IEEE transactions on pattern analysis and machine intelligence*, **33**(7), 1457–1469. (Cited on page 63.)

Levin, Anat, Rav-Acha, Alex, & Lischinski, Dani. 2008. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(10), 1699–1712. (Cited on page 36.)

Levoy, Marc, & Hanrahan, Pat. 1996. Light field rendering. *In: SIGGRAPH*. (Cited on pages 10, 11 and 28.)

Lim, Joseph J., Pirsiavash, Hamed, & Torralba, Antonio. 2013. Parsing IKEA Objects: Fine Pose Estimation. *In: ICCV*. (Cited on page 62.)

Lin, Yen-Liang, Morariu, Vlad I, Hsu, Winston, & Davis, Larry S. 2014. Jointly optimizing 3d model fitting and fine-grained classification. *Pages 466–480 of: European Conference on Computer Vision*. Springer. (Cited on page 63.)

Lipski, Christian, Klose, Felix, & Magnor, Marcus. 2014. Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video. *IEEE Transactions on Circuits and Systems for Video Technology*, **24**(6), 942–951. (Cited on pages 4, 16, 17, 20, 22, 28, 45, 46, 47 and 56.)

Lipski, Christian, Hilsmann, Anna, Dachsbacher, Carsten, & Eisemann, Martin. 2015. Image- and Video-based Rendering. *Pages 261–280 of:* Marcus Magnor, Oliver Grau, Olga Sorkine-Hornung Christian Theobalt (ed), *Digital Representations of the Real World: How to Capture, Model, and Render Visual Reality*. CRC Press. (Cited on pages 9 and 60.)

Lischinski, Dani, & Rappoport, Ari. 1998. Image-based rendering for non-diffuse synthetic scenes. *Pages 301–314 of: Rendering Techniques 98*. Springer. (Cited on page 23.)

Liu, Ce, Yuen, Jenny, & Torralba, Antonio. 2011. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, **33**(5), 978–994. (Cited on page 16.)

Liu, Feng, Gleicher, Michael, Jin, Hailin, & Agarwala, Aseem. 2009. Content-preserving Warps for 3D Video Stabilization. *Pages 44:1–44:9 of: ACM SIGGRAPH 2009 Papers*. SIGGRAPH '09. New York, NY, USA: ACM. (Cited on page 22.)

Lowe, D. 1987. The Viewpoint Consistency Constraint. *IJCV*, **1**(1), 57–72. (Cited on page 62.)

Lowe, David G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, **60**(2), 91–110. (Cited on page 12.)

Magri, Luca, & Fusiello, Andrea. 2014. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. *Pages 3954–3961 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 36.)

Mantiuk, Rafat, Kim, Kil Joong, Rempel, Allan G, & Heidrich, Wolfgang. 2011. HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. *Page 40 of: ACM Transactions on Graphics (TOG)*, vol. 30. ACM. (Cited on pages 49, 50 and 78.)

Mark, William R, McMillan, Leonard, & Bishop, Gary. 1997. Post-rendering 3D warping. *Pages 7–ff of: Proceedings of the 1997 symposium on Interactive 3D graphics*. ACM. (Cited on pages 11 and 12.)

Massa, Francisco, Russell, Bryan C., & Aubry, Mathieu. 2015. Deep Exemplar 2D-3D Detection by Adapting from Real to Rendered Views. *CoRR*, **abs/1512.02497**. (Cited on pages 62, 66 and 67.)

McMillan, Leonard, & Bishop, Gary. 1995. Plenoptic modeling: An image-based rendering system. *Pages 39–46 of: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. (Cited on page 10.)

Micusik, Branislav, & Kosecka, Jana. 2009. Piecewise planar city 3D modeling from street view panoramic sequences. *Pages 2906–2912 of: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. (Cited on page 4.)

Mičušík, Branislav, & Košecká, Jana. 2010. Multi-view superpixel stereo in urban environments. *International journal of computer vision*, **89**(1), 106–119. (Cited on page 33.)

Mottaghi, Roozbeh, Xiang, Yu, & Savarese, Silvio. 2015. A coarse-to-fine model for 3D pose estimation and sub-category recognition. *Pages 418–426 of: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. (Cited on page 63.)

Moulon, Pierre, Monasse, Pascal, & Marlet, Renaud. 2012. Adaptive structure from motion with a contrario model estimation. *Pages 257–270 of: Asian Conference on Computer Vision*. Springer. (Cited on page 13.)

Okura, Fumio, Vanhoey, Kenneth, Bousseau, Adrien, Efros, Alexei A, & Drettakis, George. 2015. Unifying color and texture transfer for predictive appearance manipulation. *Pages 53–63 of: Computer Graphics Forum*, vol. 34. Wiley Online Library. (Cited on page 107.)

Okutomi, Masatoshi, & Kanade, Takeo. 1993. A multiple-baseline stereo. *IEEE Transactions on pattern analysis and machine intelligence*, **15**(4), 353–363. (Cited on page 13.)

Ortiz-Cayon, Rodrigo, Djelouah, Abdelaziz, & Drettakis, George. 2015 (Oct.). A Bayesian Approach for Selective Image-Based Rendering using Superpixels. *In: International Conference on 3D Vision - 3DV*. (Cited on pages 65, 73, 74 and 77.)

Pharr, Matt, & Humphreys, Greg. 2004. *Physically based rendering: From theory to implementation*. Morgan Kaufmann. (Cited on page 1.)

Pollefeys, Marc, Van Gool, Luc, Vergauwen, Maarten, Verbiest, Frank, Cornelis, Kurt, Tops, Jan, & Koch, Reinhard. 2004. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, **59**(3), 207–232. (Cited on page 17.)

Pollefeys, Marc, Nistér, David, Frahm, J-M, Akbarzadeh, Amir, Mordohai, Philippos, Clipp, Brian, Engels, Chris, Gallup, David, Kim, S-J, Merrell, Paul, *et al.* 2008. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, **78**(2-3), 143–167. (Cited on page 43.)

Prasad, Mukta, & Fitzgibbon, Andrew. 2006. Single view reconstruction of curved surfaces. *Pages 1345–1354 of: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE. (Cited on page 64.)

Prisacariu, Victor A, & Reid, Ian D. 2012. Pwp3d: Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, **98**(3), 335–354. (Cited on pages 63 and 103.)

Prisacariu, Victor Adrian, & Reid, Ian. 2011. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. *Pages 2185–2192 of: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. (Cited on page 63.)

Prisacariu, Victor Adrian, Segal, Aleksandr V, & Reid, Ian. 2012. Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. *Pages 593–606 of: Asian Conference on Computer Vision*. Springer. (Cited on page 63.)

Pujades, Sergi, Devernay, Frédéric, & Goldluecke, Bastian. 2014. Bayesian View Synthesis and Image-Based Rendering Principles. *In: CVPR*. (Cited on pages 24, 28, 32 and 80.)

Pulli, Kari, Hoppe, Hugues, Cohen, Michael, Shapiro, Linda, Duchamp, Tom, & Stuetzle, Werner. 1997. View-based rendering: Visualizing real objects from scanned range and color data. *Pages 23–34 of: Rendering techniques 97*. Springer. (Cited on page 17.)

Reinhard, Erik, Pouli, Tania, Kunkel, Timo, Long, Ben, Ballestad, Anders, & Damberg, Gerwin. 2012. Calibrated image appearance reproduction. *ACM Transactions on Graphics (TOG)*, **31**(6), 201. (Cited on page 49.)

Rematas, Konstantinos, Ritschel, Tobias, Fritz, Mario, & Tuytelaars, Tinne. 2014. Image-based synthesis and re-synthesis of viewpoints guided by 3d models. *Pages 3898–3905 of: 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. (Cited on pages 4, 5 and 86.)

Reynolds, Malcolm, Doboš, Jozef, Peel, Leto, Weyrich, Tim, & Brostow, Gabriel J. 2011. Capturing time-of-flight data with confidence. *Pages 945–952 of: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. (Cited on page 85.)

Roberts, L. 1965. Machine perception of 3-D solids. *In: PhD. Thesis*. (Cited on page 62.)

Rosenhahn, Bodo, Brox, Thomas, & Weickert, Joachim. 2007. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, **73**(3), 243–262. (Cited on page 63.)

Ruhl, Kai, Klose, Felix, Lipski, Christian, & Magnor, Marcus. 2012. Integrating approximate depth data into dense image correspondence estimation. *Pages 26–31 of: Proceedings of the 9th European Conference on Visual Media Production*. ACM. (Cited on page 17.)

Rusinkiewicz, Szymon, & Levoy, Marc. 2001. Efficient variants of the ICP algorithm. *Pages 145–152 of: 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE. (Cited on page 70.)

Sandhu, Romeil, Dambreville, Samuel, Yezzi, Anthony, & Tannenbaum, Allen. 2011. A nonrigid kernel-based framework for 2D-3D pose estimation and 2D image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, **33**(6), 1098–1115. (Cited on page 63.)

Scharstein, Daniel, & Szeliski, Richard. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, **47**(1-3), 7–42. (Cited on page 13.)

Seitz, Steven M, Curless, Brian, Diebel, James, Scharstein, Daniel, & Szeliski, Richard. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. *Pages 519–528 of: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1. IEEE. (Cited on page 13.)

Shade, Jonathan, Gortler, Steven, He, Li-wei, & Szeliski, Richard. 1998. Layered depth images. *Pages 231–242 of: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM. (Cited on pages 11, 12 and 20.)

Shum, Harry, & Kang, Sing B. 2000. Review of image-based rendering techniques. *Pages 2–13 of: Visual Communications and Image Processing 2000*. International Society for Optics and Photonics. (Cited on page 9.)

Shum, Heung-Yeung, & He, Li-Wei. 1999. Rendering with concentric mosaics. *Pages 299–306 of: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. (Cited on page 10.)

Shum, Heung-Yeung, Chan, Shing-Chow, & Kang, Sing Bing. 2008. *Image-based rendering*. Springer Science & Business Media. (Cited on page 9.)

Simonyan, K., & Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, **abs/1409.1556**. (Cited on pages 62, 63 and 67.)

Sinha, Sudipta N., Steedly, Drew, & Szeliski, Richard. 2009. Piecewise planar stereo for image-based rendering. *In: ICCV*. (Cited on pages 15 and 16.)

Sinha, Sudipta N, Kopf, Johannes, Goesele, Michael, Scharstein, Daniel, & Szeliski, Richard. 2012. Image-based rendering for scenes with reflections. *ACM Trans. Graph.* (Cited on page 23.)

Snavely, Noah, Seitz, Steven M., & Szeliski, Richard. 2006. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph.* (Cited on page 13.)

Sorkine, Olga, & Alexa, Marc. 2007. As-rigid-as-possible surface modeling. *In: Symposium on Geometry processing*, vol. 4. (Cited on pages 72 and 73.)

Stich, Timo, Linz, Christian, Albuquerque, Georgia, & Magnor, Marcus. 2008. View and time interpolation in image space. *Pages 1781–1787 of: Computer Graphics Forum*, vol. 27. Wiley Online Library. (Cited on page 16.)

Stich, Timo, Linz, Christian, Wallraven, Christian, Cunningham, Douglas, & Magnor, Marcus. 2011. Perception-motivated interpolation of image sequences. *ACM Transactions on Applied Perception (TAP)*, **8**(2), 11. (Cited on page 16.)

Su, H., Huang, Q., Mitra, N.J., Li, Y., & Guibas, L. 2014. Estimating image depth using shape collections. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, **33**(4). (Cited on page 62.)

Tola, Engin, Lepetit, Vincent, & Fua, Pascal. 2010. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, **32**(5), 815–830. (Cited on pages 12 and 82.)

Tomasi, Carlo, & Kanade, Takeo. 1992. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, **9**(2), 137–154. (Cited on page 13.)

Vangorp, Peter, Chaurasia, Gaurav, Laffont, P-Y, Fleming, Roland W, & Drettakis, George. 2011. Perception of Visual Artifacts in Image-Based Rendering of Façades. *Pages 1241–1250 of: Computer Graphics Forum*, vol. 30. Wiley Online Library. (Cited on page 25.)

Vangorp, Peter, Richardt, Christian, Cooper, Emily A, Chaurasia, Gaurav, Banks, Martin S, & Drettakis, George. 2013. Perception of perspective distortions in image-based rendering. *ACM Transactions on Graphics (TOG)*, **32**(4), 58. (Cited on page 25.)

Wang, Zhou, Bovik, Alan C, Sheikh, Hamid R, & Simoncelli, Eero P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, **13**(4), 600–612. (Cited on page 49.)

Woodford, Oliver, & Fitzgibbon, Andrew W. 2005. Fast image-based rendering using hierarchical image-based priors. *In: BMVC*. (Cited on page 24.)

Wu, C., Agarwal, S., Curless, B., & Seitz, S. M. 2011 (June). Multicore bundle adjustment. *Pages 3057–3064 of: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. (Cited on pages 13, 41, 43 and 64.)

Wu, Changchang. 2007. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). (Cited on page 13.)

Wu, Changchang. 2013. Towards linear-time incremental structure from motion. *Pages 127–134 of: 2013 International Conference on 3D Vision-3DV 2013*. IEEE. (Cited on page 64.)

Xiang, Yu, Kim, Wonhui, Chen, Wei, Ji, Jingwei, Choy, Christopher, Su, Hao, Mottaghi, Roozbeh, Guibas, Leonidas, & Savarese, Silvio. 2016. ObjectNet3D: A Large Scale Database for 3D Object Recognition. *In: European Conference Computer Vision (ECCV)*. (Cited on pages 60, 77 and 98.)

Xiao, Jianxiong, Fang, Tian, Tan, Ping, Zhao, Peng, Ofek, Eyal, & Quan, Long. 2008. Image-based façade modeling. *Page 161 of: ACM transactions on graphics (TOG)*, vol. 27. ACM. (Cited on page 4.)

Xiao, Jianxiong, Fang, Tian, Zhao, Peng, Lhuillier, Maxime, & Quan, Long. 2009. Image-based street-side city modeling. *ACM Transactions on Graphics (TOG)*, **28**(5), 114. (Cited on page 4.)

Xu, Kai, Zheng, Hanlin, Zhang, Hao, Cohen-Or, Daniel, Liu, Ligang, & Xiong, Yueshan. 2011. Photo-inspired model-driven 3D object modeling. *Page 80 of: ACM Transactions on Graphics (TOG)*, vol. 30. ACM. (Cited on page 64.)

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, & Lipson, Hod. 2014. How transferable are features in deep neural networks? *Pages 3320–3328 of: Advances in neural information processing systems*. (Cited on page 63.)

Zhang, Cha, & Chen, Tsuhan. 2004. A survey on image-based rendering - Representation, sampling and compression. *In: Signal Processing: Image Communication*. (Cited on page 9.)

Zheng, Ke Colin, Colburn, Alex, Agarwala, Aseem, Agrawala, Maneesh, Curless, B, Salesin, D, & Cohen, M. 2009a. *A consistent segmentation approach to image-based rendering*. Tech. rept. Technical Report CSE-09-03-02, University of Washington. (Cited on pages 21 and 22.)

Zheng, Ke Colin, Colburn, Alex, Agarwala, Aseem, Agrawala, Maneesh, Salesin, David, Curless, Brian, & Cohen, Michael F. 2009b. Parallax photography: creating 3d cinematic effects from stills. *Pages 111–118 of: Proceedings of Graphics Interface 2009*. Canadian Information Processing Society. (Cited on page 21.)

Zheng, Youyi, Chen, Xiang, Cheng, Ming-Ming, Zhou, Kun, Hu, Shi-Min, & Mitra, Niloy J. 2012. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph.*, **31**(4), 99–1. (Cited on page 64.)

Zhou, Tinghui, Tulsiani, Shubham, Sun, Weilun, Malik, Jitendra, & Efros, Alexei A. 2016. View Synthesis by Appearance Flow. *arXiv preprint arXiv:1605.03557*. (Cited on pages 4, 5 and 25.)

Zitnick, C Lawrence, & Kang, Sing Bing. 2007. Stereo for image-based rendering using image over-segmentation. *IJCV*. (Cited on pages 4, 15, 20, 28, 30 and 60.)

Zitnick, C. Lawrence, Kang, Sing Bing, Uyttendaele, Matthew, Winder, Simon, & Szeliski, Richard. 2004. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.* (Cited on pages 17, 20, 30, 33 and 36.)