UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Édition et rendu à base d'images multi-vues par apprentissage profond et optimisation

Julien Philip

Inria Sophia Antipolis-Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur
Dirigée par : George Drettakis
Soutenue le : 25 Septembre 2020

Devant le jury composé de :
Céline Loscos, Professeure, Université de Reims Champagne-Ardenne
Jean-François Lalonde, Professeur, Université Laval
Jaakko Lehtinen, Professeur, Aalto University
Florent Lafarge, Chercheur, Inria Sophia Antipolis - Méditerranée
George Drettakis, Directeur de Recherche, Inria Sophia Antipolis - Méditerranée
Michaël Gharbi, Chercheur, Adobe Research

Inria

Édition et rendu à base d'images multi-vues par apprentissage profond et optimisation

---

Multi-view image-based editing and rendering through deep learning and optimization

Jury:

Présidente du jury / President of the jury
Céline Loscos, Professeure, Université de Reims Champagne-Ardenne

Rapporteurs / Reviewers
Jean-François Lalonde, Professeur, Université Laval
Jaakko Lehtinen, Professeur, Aalto University

Examinateurs / Examiners
Florent Lafarge, Chercheur, Inria Sophia Antipolis - Méditerranée

Visiteurs / Visitors
Michaël Gharbi, Chercheur, Adobe Research

Directeur de thèse / Thesis supervisor
George Drettakis, Directeur de Recherche, Inria Sophia Antipolis - Méditerranée

Résumé

Les images de synthèse (CGI) prennent une place grandissante dans notre environnement. Que ce soit dans les jeux vidéos ou les films, leur qualité ne cesse de s'accroître nécessitant la création fastidieuse de contenus artistiques. L'émergence de la réalité virtuelle et augmentée, entraine la nécessité de rendre des environnements existants. Pour permettre l'utilisation généralisée des images de synthèse dans des applications telles que la télé-présence ou les visites virtuelles, la digitalisation manuelle des contenus par des artistes se doit d'être évitée. Une des solutions peut provenir des techniques de Rendu à Base d'Images (IBR) qui permettent de rendre des scènes, depuis un point de vue libre, à partir d'un ensemble de photographies parcimonieux. Bien que ces méthodes ne nécessitent que peu de travail artistique, elles n'autorisent cependant pas le contrôle ou l'édition du contenu. Dans cette thèse, nous explorons l'Edition et le Rendu d'Images Multi-vues. Afin de permettre à des scènes, capturées avec le moins de contraintes possibles, d'être rendues avec des altérations telles que la suppression d'objets, l'édition d'éclairage, ou la composition de scènes, nous exploitons les techniques d'optimisation et d'apprentissage profond. Nous concevons nos méthodes afin qu'elles tirent pleinement avantage de l'information présente dans le contenu multi-vues, tout en en respectant ses contraintes spécifiques. Pour la suppression d'objets, nous introduisons un algorithme de remplissage automatique, multi-vues cohérent, utilisant une représentation planaire. Les plans sont des objets simples et efficaces pour combler la géométrie, dont la cohérence multi-vues émerge naturellement lorsque le remplissage est effectué dans un espace texture rectifié et partagé. Ils permettent aussi le respect des effets de perspective. Nous démontrons la capacité d'enlever des objets, à grande l'échelle, dans des scènes contenant plusieurs centaines d'images. Nous traitons ensuite le problème du rééclairage des scènes extérieures par une méthode d'apprentissage profond. Elle permet de modifier l'illumination, en enlevant et synthétisant les ombres portées, pour une position du soleil quelconque, tout en tenant compte des variations d'illumination globale. Une représentation géométrique approximative, reconstruite en utilisant la stéréo multi-vues, est utilisée pour générer des images tampons d'illumination et d'ombres qui guident un réseau de neurones. Nous entrainons ce réseau sur un ensemble de scènes

synthétiques, permettant une supervision complète. Une augmentation des données minutieuse permet à notre réseau de généraliser aux scènes réelles et de produire l'état de l'art en terme de résultats. Nous démontrons ensuite, la capacité du réseau à être utilisé pour composer des scènes réelles, capturées dans des conditions d'orientation et d'éclairages différentes. Nous présentons ensuite des contributions à la qualité de l'IBR. Nous introduisons un algorithme de maillage de cartes de profondeur et de leur simplification. Nous démontrons son impact sur la qualité et les performances d'une nouvelle méthode d'IBR utilisant l'apprentissage. Enfin, nous introduisons une méthode qui combine rééclairage, IBR, et analyse de matériaux. Afin de permettre un rendu à base d'images, rééclairable et tenant compte des effets spéculaires, nous extrayons du contenu multi-vues les variations d'apparence des matériaux et l'information de texture haute résolution, sous la forme de plusieurs rendus IBR heuristiques. Nous les combinons ensuite avec des rendus d'irradiance, obtenus par lancer de rayons, qui spécifient les conditions d'éclairage initiales et désirées. Cette combinaison permet d'entrainer un réseau de neurones à extraire implicitement les propriétés des matériaux et à produire des points de vues rééclairés réalistes. La séparation de la supervision entre composante diffuse et spéculaire fut démontrée cruciale dans l'obtention de résultats haute-qualité.

Mots-clés: Rendu Basé Images, Multi-vue, Inpainting, Rééclairage, Rendu Neuronal

---

Abstract

Computer-generated imagery (CGI) takes a growing place in our everyday environment. Whether it is in video games or movies, CGI techniques are constantly improving in quality but also require ever more qualitative artistic content which takes a growing time to create. With the emergence of virtual and augmented reality, often comes the need to render or re-render assets that exist in our world. To allow widespread use of CGI in applications such as telepresence or virtual visits, the need for manual artistic replication of assets must be removed from the process. This can be done with the help of Image-Based Rendering (IBR) techniques that allow scenes or objects to be rendered in a free-viewpoint manner from a set of sparse input photographs. While this process requires little to no artistic work, it also does not allow for artistic control or editing of scene content. In this dissertation, we explore Multi-view Image Editing and Rendering. To allow casually captured scenes to be rendered with content alterations such as object removal, lighting editing, or scene compositing, we leverage the use of optimization techniques and modern deep-learning. We design our methods to take advantage of all the information present in multi-view content while handling specific constraints such as multi-view coherency. For object removal, we introduce a new plane-based multi-view inpainting algorithm. Planes are a simple yet effective way to fill geometry and they naturally enforce multi-view coherency as inpainting is computed in a shared rectified texture space, allowing us to correctly respect perspective. We demonstrate instance-based object removal at the scale of a street in scenes composed of several hundreds of images. We next address outdoor relighting with a learning-based algorithm that efficiently allows the illumination in a scene to be changed, while removing and synthesizing cast shadows for any given sun position and accounting for global illumination. An approximate geometric proxy built using multi-view stereo is used to generate illumination and shadow related image buffers that guide a neural network. We train this network on a set of synthetic scenes allowing full supervision of the learning pipeline. Careful data augmentation allows our network to transfer to real scenes and provides state of the art relighting results. We also demonstrate the capacity of this network to be used to compose real scenes captured under different

lighting conditions and orientation. We then present contributions to image-based rendering quality. We discuss how our carefully designed depth-map meshing and simplification algorithm improve rendering performance and quality of a new learning-based IBR method. Finally, we present a method that combines relighting, IBR, and material analysis. To enable relightable IBR with accurate glossy effects, we extract both material appearance variations and qualitative texture information from multi-view content in the form of several IBR heuristics. We further combine them with path-traced irradiance images that specify the input and target lighting. This combination allows a neural network to be trained to implicitly extract material properties and produce realistic-looking relit viewpoints. Separating diffuse and specular supervision is crucial in obtaining high-quality output.

Keywords: Image Based Rendering, Multi-view, Inpainting, Relighting, Neural Rendering

---

Acknowledgements

The work presented in this thesis would not exist without the dedication, enthusiasm and work of my advisor **George Drettakis**. I want to thank him for the guidance and the expertise he provided during these four years. I also want to thank him for his trust and, above all, his love for science that he passed on to me.

I would like to thank my coauthors for their investment in the different projects and their valuable advice. They provided insight without which the present work would not exist. A specific thanks to **Michaël Gharbi**, for all the good advice and discussions, for helping me when I was in doubt and for giving me the opportunity to intern at Adobe Research. To **Peter Hedman**, many thanks for trusting me to work on his project, for showing me how to properly handle a deadline and for all the guidance and tips later on. I would also like to thank **Alexei A. Efros** for welcoming me to Berkeley. Thanks also to the anonymous reviewers for their useful insights that helped improved the quality of my work.

Going through PhD studies is also being part of a team, I would like to thank all my colleagues at the GraphDeco group. Through our many arguments and discussions they all impacted positively my thesis and my experience as a student. A special thanks to **Valentin Deschaintre** and **Simon Rodriguez** for facing the doctoral studies together from day one and for their very useful help. To **Bastien Wailly** thanks for sharing with me his love for science, rockets and learning new things, and for all these gaming nights. Behind the work presented in this thesis, many engineering challenges were solved thanks to **Sébastien Morgenthaler**, thanks for staying late with me during the deadlines, thanks also for tolerating my stress, for your constant positiveness and for always being enthusiastic. To my good old friend **George Koulieris**, many thanks for your support and mentoring at the beginning of my PhD studies, I'll always be grateful to you for supporting my weird sense of humor and for all the good laughs.

Thanks to all my friends for helping me get out of my computer scientist life from time to time and for pretending to understand what I was doing. To **Valentin, Lucas, Jerôme, Pierre, Mouss, Flo, Charlène, Matthias, Kenny, Clément** and all the **Bedfas** thanks!

For helping me be more confident during hard times and for some of the best pieces of advice I could ever get thank you **Philippe Boneff**.

I spare a deep thought for all my past teachers who taught me how to learn, how to be curious and the beauty of science. A special thank to **Alexandre Marino** for sharing his love for math.

I am immensely grateful to my parents, **Brigitte** and **Hervé** for pushing me to do my best, for supporting me, for providing the best education I could dream of during all of those years and for the love and care they gave me. To my sister **Maga**, thanks for showing me the way to go, for being the calm one and for always listening to me when I needed so. Finally I would like to thank **Camille Bey**, my girlfriend for her support, for her tremendous help during deadlines and for doing everything she could to make these four years as sweet as possible.

À Marcel, Huguette, Émile et Henri

Contents

Chapter 1

Introduction

Over the last decades, digital technology occupies a growing part of our daily environment. Nowadays, many of us barely spend a day without using a screen. While the technology was being democratized, the initial text-based interfaces were replaced with more user-friendly graphical user interfaces. Computer graphics techniques started in the entertainment industry and now have a significant impact in many domains. Video games and movies are the most obvious ones, but Computer Graphics (CG) are also used extensively in advertising, design, architecture or even scientific visualization and healthcare. At the heart of CG lies the need to create and display content. While hardware capabilities increased for display and mathematical approximations were developed to accelerate computation, content creation remained a time-consuming process. Even with massive improvements in 3D asset generation software, the pace at which computation improved could not be followed. Designing a 3D scene or object requires different steps depending on its final target usage. In the case of photo-realistic imagery, very precise geometry and textures must be created by artists. Materials have to be designed and applied to this geometry before lighting can be set up. In the case of non-static scenes, animation adds another layer of work. All these steps let all the imagination and talent of artists be expressed in a very flexible manner, often leading to beautiful images and photorealistic renderings such as the ones visible in figure 1.1. For high budget movies, this creation process is an acceptable time and financial constraint, but for consumer applications such as digital double, telepresence, or virtual visits it is an impediment. To be able to integrate CG with our everyday environment, for instance in mixed or augmented reality, the replication process of existing assets must be automated and the acquisition setup simplified. This thesis studies ways of conciliating CG quality and flexibility with real environment capture and rendering. We present techniques that allow users to navigate through casually captured scenes while giving them back some of the flexibility and editability inherent to classical computer graphics techniques.

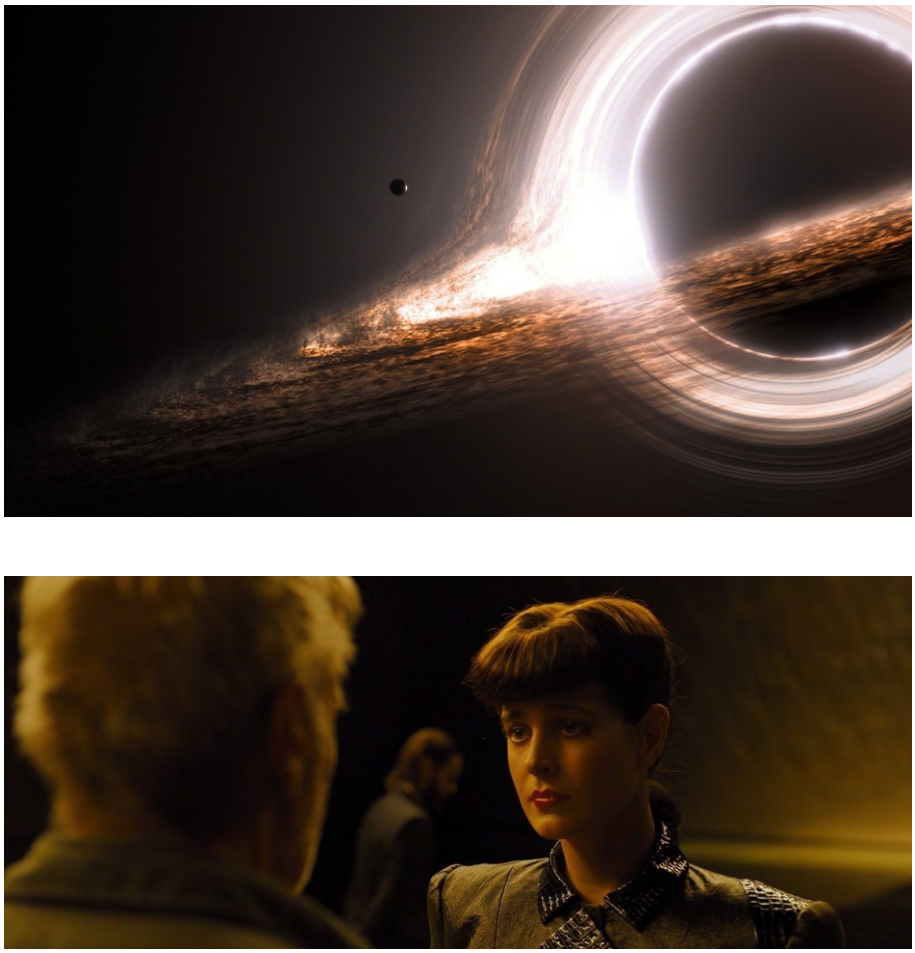


Figure 1.1: Top: CGI of a black hole taken from the movie interstellar[a]. The effects of gravity on light propagation were simulated to render this image. Bottom: Image of a rendering of a digital double of Sean Young created from the original Blade Runner movie and used in the Blade runner 2049 sequel[b].

[a] http://dans-la-lune.fr/2015/11/08/la-science-dinterstellar-2-le-trou-noir-gargantua/
[b] https://www.youtube.com/watch?v=724JhpqKEmY

1.1 Rendering and Captured scenes

Rendering scenes is traditionally done using one of two different techniques depending on the context. The first method is light transport simulation through the use of path tracing. In that setup, rays are cast from a virtual camera toward the scene and then bounces of these rays are used to integrate the incoming irradiance recursively. This allows the computation of an unbiased estimation of light transport in a given environment which

means it allows to realistically simulate complex lighting effects. This process has two main drawbacks though, the first one is that it is a very computationally expensive process. The recursivity of the process and the number of samples required to get noise-free images, make this approach unsuited for rendering on consumer hardware in realtime. The second drawback is that the realism of the rendering heavily relies on the underlying scene description that is used. Even with perfect light simulation, the complexity of our world and the level of detail required to produce plausible images is reflected in the assets used for the simulation. The importance of asset quality is visible in figure 1.2.

The second method used for real-time rendering in most game engines is called rasterization. Geometry is projected to the screen and then shaded directly with approximations. This process relies on simplifying assumptions and precomputation to render realistic-looking images in realtime. This method –while practical– cannot reproduce some lighting effects easily such as glossy reflections on complex surfaces or caustics. It shares the second drawback of path tracing, relying heavily on asset quality. It is also adapted to specific lighting configuration and effects on a per scene basis.

Figure 1.2: Two scenes rendered using Mitsuba's path tracer [87]. Left: the Cornell Box[a], a very simple scene with only diffuse materials. Right: The GT rendering of the glossy kitchen scene from Diolatzis et al. [38]. While the same light transport engine was used for both scenes, the second one looks a lot more realistic due to its complex geometry and materials.

[a] http://www.graphics.cornell.edu/online/box/

As we can see, with the traditional pipeline, asset creation has a huge impact on the outcome. In the context of real asset renderings such as people, objects, or full scenes,

one would need to model, with a very high level of fidelity, geometry, materials, and lighting to be able to use the aforementioned methods and get realistic results. This would make the process impossible to generalize and scale up to the potential billions of users and assets.

There exist different ways of rendering captured assets. One approach that has gained popularity is to take multiple photos of a scene, leverage structure from motion (SfM), and multi-view stereo (MVS) to obtain a 3D proxy and texture it automatically. While this can give decent results as we can see in figure 1.3, it has several limitations: first, the *quality of geometry* may vary depending on the density of capture and lead to strong visual artifacts (see fig 1.3 middle), second, the *view-dependent effects* such as glossy surfaces, mirrors, and specular highlight are either removed or baked into the texture (see fig 1.3 right). Finally, the lighting, geometry, and materials are constrained by the capture conditions.

Figure 1.3: Left: Rendering of an interior scene using a textured mesh. The overall quality is acceptable. Middle: illustrations of two of the visible artifacts with the textured approach. The geometry around the chair leg is very noisy leading to visual artifacts, the highlight caused by the lamp is not visible and residual highlights are baked in the texture. Right: inset of a input view with a visible highlight for comparison.

To overcome the first and second issues, *i.e.*, artifacts due to geometry and missing view-dependent effects, image-based rendering methods have been proposed. Instead of baking a single texture from the images, images are reprojected on the geometry and blended with weights depending on the viewer's position and its orientation with respect to the surface [19]. Many methods have improved this basic approach but artifacts due to geometry error remain an issue.

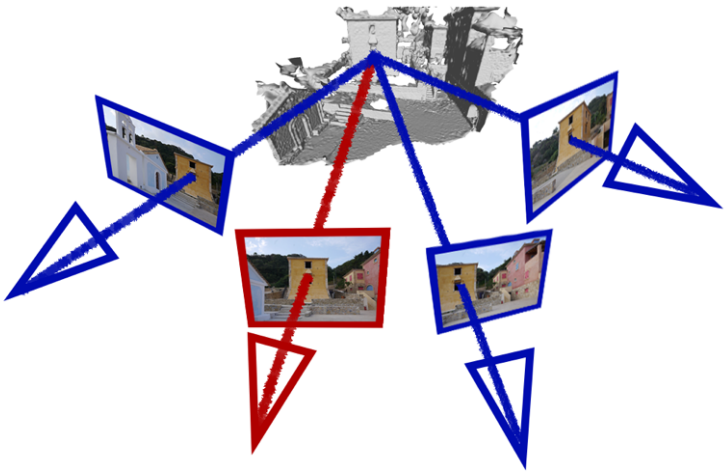


Figure 1.4: Illustration of the traditional IBR pipeline. Selected input views (in blue) are reprojected in a novel view (in red) using a proxy geometry and then blended to form the rendering.

1.2 Need of flexibility

As mentioned, in Image-Based Rendering (IBR), captured scenes are usually rendered using reprojection of photographs. The usual pipeline involves taking photos of a scene, building a 3D representation, for instance using 3D meshes, and a method to select views, reproject, and blend them. It allows rendering content without light transport simulation nor the need for manual asset creation.

While IBR is a good direction to overcome geometry issues and some view-dependent effects artifacts, it does not address the last limitation mentioned previously: the scene content is completely fixed. When capturing a scene or an object, one does not necessarily control all the capture conditions such as the surroundings, the presence of people in large-scale scenes or the lighting. Depending on the weather and time of day, outdoor scenes may only be captured under different lighting conditions than the one desired. Moreover, it can be useful to be able to display the same content with varying lighting for instance in the case of a virtual visit of an apartment. With IBR only, this would require capturing the place under all the desired lighting conditions which defeats the purpose of avoiding the time-consuming manual creation process. Methods have been developed to edit the lighting of pictures. This process is referred to as *relighting*. While some methods give good results on single images there are very few that work on high-quality

multi-view datasets that are required for real asset rendering and never with a high level of control from the end-user. Another issue that can occur is the presence of undesired content when capturing the scenes. There can be moving people or cars in public places that are either undesired or can even impair rendering quality with ghosting artifacts or broken geometry (see figure 1.5). Being able to remove content can also be useful, *e.g.*, in the context of refurnishing with synthetic assets, where existing furniture must be removed before superimposing synthetic ones. Removing part of images, is referred to as *inpainting*. It has been widely studied in the context of single images but these methods do not apply to multi-view content as treating each frame independently leads to multi-view incoherencies. Recent methods investigated inpainting in the same context as ours [183] but only softly enforce coherency without respecting strong 3D cues such as perspective.

Figure 1.5: Unstructured Lumigraph rendering[19] of a scene captured with moving people and specular objects. When sampling the input images, moving people are blended with the background leading to ghosting artifacts. The specular parts of the car are badly reconstructed leading to severe visual artifacts.

1.3 Contributions

The need for better, more flexible Image-Based techniques motivated the research presented in this thesis. We explore new ways of editing and rendering multi-view data that are a step toward bringing together the flexibility of traditional computer graphics and the ease of capturing assets with images. We work on unstructured sets of pictures of real-world environments, from which we obtain a proxy geometry of the scenes using MVS. We apply optimization and deep learning algorithms to obtain novel, high-quality

edited renderings. We believe that improving the flexibility of image-based methods has the potential to increase their adoption and the number of use-cases.

Through the five projects presented in this thesis, we went from treating isolated issues, toward a more general neural rendering approach that integrates and generalizes some of our findings. The contributions of this thesis are presented as follow:

- Chapter 3: a new multi-view inpainting method that can handle up to several hundreds of images of large scale scenes. To this end, we introduce a shared rectified piecewise planar space in which the inpainting is done using a resolution aware patch-match approach. This space enforces multiview coherency while respecting perspective effects. The new patch-match approach saves computation by only performing high-resolution inpainting when it is required.

- Chapter 4: a novel deep learning-based multi-view relighting solution for outdoor scenes with a high level of user control. We train a deep neural network to directly produce a relit image from an input photo and image-space buffers generated by computer graphics. Because of their non-local nature and their importance for outdoor scenes, shadows are carefully treated by introducing RGB shadow images in the network. RGB shadow images, that are refined by the first stage of the network, allow to correctly remove and synthesize shadows while overcoming MVS geometry inaccuracies. We train our network on synthetic data allowing full supervision of both relighting and shadow refinement. To avoid a domain gap we use a dual representation of the training scenes, with ground truth geometry for supervision and MVS like geometry to generate the inputs to the network. We also present a novel application of this network to captured scene composition. This application was mostly implemented by Baptiste Nicolet, based on the original code of the project, while he was interning in our group.

- Chapter 5: a novel depth map meshing strategy that has a significant positive impact on the quality of the deep blending image-based rendering method. We introduce an occlusion edge detection method and a simplification scheme that is adapted to Image-Based rendering, adapting the rate of simplification in image space. While we discuss the full deep blending pipeline, developed by Peter Hedman, the contribution to this thesis is limited to the meshing algorithm.

- Chapter 6: a relightable neural renderer for indoor scenes. Motivated by our results for outdoor, we worked on the more challenging indoor setup. In this method, we mix physically based rendering, IBR, and material analysis to treat global illumination and specularities realistically. Our neural network takes as input several observations of surface behavior, thanks to image reprojection akin to IBR. Target lighting conditions are described as an approximate irradiance map computed using PBR, while the source ones are computed as an image-based final gathering. Reflexions are correctly synthesized with the help of a mirror image buffer. View-dependent effects are produced and supervised separately from diffuse ones, allowing temporal stability and better final quality. We again use synthetic training data and the same dual representation as in Chapter 4.

1.4 Funding and Publications

The work in this thesis was funded by a European Union's Horizon 2020 research and innovation program under grant agreement No 727188 [1] and the ERC Advanced Grant No. 788065 FUNGRAPH [2]. The Neural Relightable Rendering project was partially conducted when the author was interning at Adobe Research.

The work in this thesis has led to four publications in international venues, out of which two are first author publications, and a first author project still under review:

- Plane-based multi-view inpainting for image-based rendering in large scenes.
  Philip and Drettakis [146]
  Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games

- Deep Blending for Free-viewpoint Image-based Rendering.
  Hedman, Philip, Price, Frahm, Drettakis, and Brostow [72]
  ACM Transactions on Graphics (TOG)

- Multi-view Relighting Using a Geometry-aware Network.
  Philip, Gharbi, Zhou, Efros, and Drettakis [148]
  ACM Transactions on Graphics (TOG)

[1] https://emotiveproject.eu/
[2] https://project.inria.fr/fungraph/

- Repurposing a Relighting Network for Realistic Compositions of Captured Scenes.
  Nicolet, Philip, and Drettakis [138]
  Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games

- Relightable Neural Rendering of Multi-view Indoor Scenes
  Philip, Gharbi, Morgenthaler, and Drettakis
  In preparation

Chapter 2

Previous Work

The focus of this thesis is on improving and augmenting Image-Based techniques both in terms of quality and editability. This requires to deeply merge existing IBR concepts with ones from image editing and image analysis. We will first describe the basics of computer graphics and synthetic scene representations. Then we will show how these representations have been used to simulate light transport leading to classical physically based-rendering. While on one side computer graphics focused on rendering these environments created by artists, there has been growing interest in extracting representations of the real world leading to what is commonly referred to as *inverse rendering*. Since IBR techniques are at the crossroads of both rendering and inverse-rendering we will review both before discussing classical IBR methods that allow rendering real scenes as-is. Finally, we review methods related to re-rendering with content alteration and image editing.

2.1 Computer Graphics through explicit modeling

2.1.1 Basic representation in computer graphics concepts

A core element of classical computer graphics (CG) is the representation of the synthetic scenes one wants to render. This representation is a necessary brick, on which algorithms are built to obtain images. While this is not the only option, most scenes are described with three core elements. The first one is lighting, often described as an emissive point, a surface or volume, describing *how much* light is emitted from *where* and in *which* form. Without light, our images would be black. The second one is geometry, representing the matter with which light interacts, *i.e.*, *where* the interactions take place. Most often it is represented as a mesh of triangles and can be augmented by bump maps, displacement maps and normals. There exist many different representations and variations but throughout this thesis this is the principal representation we considered.

The third element is materials, describing *how* the light interacts with matter. They are usually represented with a bidirectional reflectance distribution function (BRDF) [137].

2.1.2 Light transport equation and CG

The light emitted by a point x at the wavelength λ and at time t towards a direction ω_o, if we neglect light propagation time, was described by Kajiya [92] as follows:

$$L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_\Omega f_r(x, \omega_i, \omega_o, \lambda, t) L_i(x, \omega_i, \lambda, t)(\omega_i \cdot n) d\omega_i \quad (2.1)$$

It is interesting to link this equation to the different elements we mentioned before. Simplifying this equation for a static scene and a given wavelength :

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_i, \omega_o) L_i(x, \omega_i)(\omega_i \cdot n) d\omega_i,$$

we can identify all the previously mentioned elements in this equation. L_e is the emissivity of a point in the given direction; it is null except on light sources for the given wavelength. x and n respectively represent the position and normals of the point considered hence requiring knowledge of the geometry. Finally f_r is the aforementioned BRDF that represents the material at point x and its interaction with light. This equation is recursive by nature, is untractable in the general case and is traditionally estimated with Monte Carlo methods [105]. This allows us to simulate light transport in a physically-based manner and to generate photorealistic images. An example of such a rendering is shown in figure 2.1.

Figure 2.1: Physically based rendering of a synthetic scene rendered using the Mitsuba [87] path-tracer.

2.2 Explicit property estimation in the world

While computer graphics is traditionally more oriented towards image generation from created content, computer vision (CV) tries to tackle the inverse problem that is, extracting information from existing images. While a large body of work tries to interpret or classify images, in the context of this thesis our main focus is on the link between CG and CV, seen as a back-and-forth process. More specifically, we will show how CG can be used to help train CV algorithms and how CV can be used to extract pieces of information that are then useful for CG tasks. Here we make a parallel with the previous section and review methods that try to extract classical computer graphics representations from images. We start by discussing existing lighting estimation methods, then we quickly review the vast body of work focusing on geometry estimation from images, and finally we present material estimation techniques.

2.2.1 Estimating lighting

Estimating the lighting environment in an image is an important step for many tasks related to content alteration such as relighting. There exist many proposed solutions,

Debevec [33] presents image-based lighting where he shows that an environment can be captured from photos. The captured environment can be represented as a light probe and can be used as a light source in the same manner as an environment map (see figure 2.2). Stumpfel et al. [173] later described how to capture HDR environments.

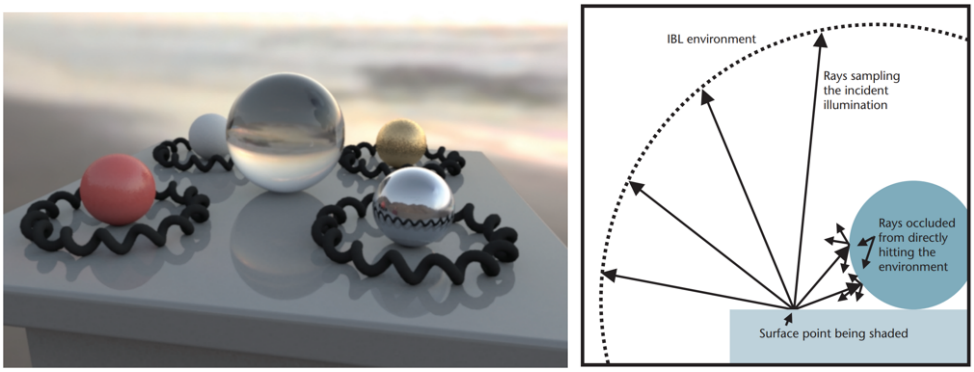


Figure 2.2: Left: Result of a traditional physically based rendering of a synthetic scene lit by a measured lighting environment. Right: Illustration of how the environment is used in the rendering. Illustrations taken from [33].

Researchers later focused on estimating lighting from a single photo [107] which is a very challenging task as the direct illumination is only partially available. More recently deep learning has been leveraged to incorporate learned priors in the lighting estimation process [54, 75, 77, 114]. LeGendre et al. [114] create a training set using physical probes attached to a mobile phone. They thus capture pairs of pictures and ground-truth probe images. They then train a network to predict the probe images from the corresponding photo. Hold-Geoffroy et al. [77] first train a sky panorama encoder-decoder, then train an image encoder to produce the same latent representation as the one of the corresponding sky, from a crop of the panorama. They obtain their full pipeline by encoding an image with the second network and decoding with the sky decoder. Some methods on the other hand estimate the lighting from the appearance of a specific object. Weber et al. [190] used an approach similar in spirit to Hold-Geoffroy et al. [77] but trained on images of objects instead of crops of panoramas. These methods are often used to composite virtual objects in a real image, which we discuss in more detail in section 2.4.1.

In this thesis, our goal is to enable scene editing in the context of IBR. One core editing that we discuss is relighting; for such applications, the estimation of initial lighting conditions is often crucial. We do not use advanced lighting estimation techniques

for our methods, thanks to multi-view data. It is however interesting to note that lighting estimation works fairly well with a small number of input images, which is more problematic for geometry estimation that we discuss later.

2.2.2 Estimating geometry

Estimating geometry from images is a long-lasting challenge in computer vision. A wide variety of approaches to geometry estimation exist, varying from laser-scans with time-of-flight sensors [111], Multi-View Stereo rigs, unstructured Multi-View Stereo (MVS) [159], stereo depth estimation, to single view depth estimation [58]. In this thesis we focus on unstructured photos that are easy to capture and suited for Image-Based editing as well as on human-made environments that exhibit structured geometry such as planar surfaces. Multi-view stereo algorithms (e.g., [50, 59, 89]) perform automatic 3D geometry reconstruction from unstructured photo datasets with diverse viewpoints. They first calibrate cameras using structure from motion (SfM), then estimate a dense point cloud and finally compute a 3D mesh from that point cloud. Examples of input images and a reconstruction can be seen in figure 2.3.

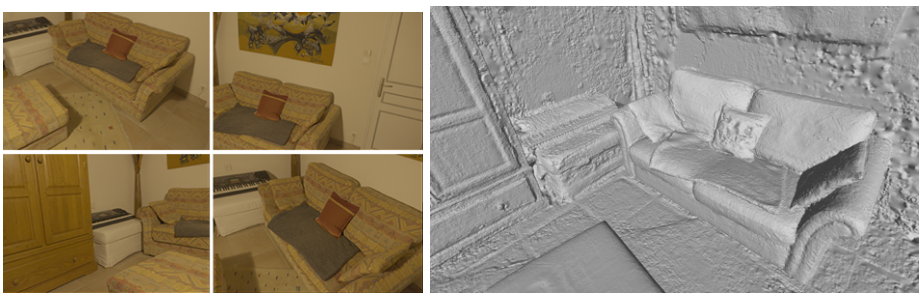


Figure 2.3: Left: Four example views used for SfM+MVS. Right: Reconstruction obtained from 253 images of the scene using Reality Capture [152].

Approaches based on Delaunay tetrahedralization (*e.g.*, , [89, 103, 152]) are able to generate impressive 3D models from photos, even in the presence of traditionally hard cases such as large textureless regions. Similarly, Ummenhofer and Brox [185] show that it is possible to generate dense meshes from noisy multi-view stereo point clouds using a regularized signed distance field. While these methods lead to globally satisfactory results, it is often at the cost of coherency between depth and detailed features in individual images. This is the reason why other methods improve the quality of individual depth maps by

discretizing the scene depths and enforcing smoothness in image-space [74, 158]. These methods are able to produce edge-aligned geometry, which is smooth for textureless regions, albeit with visible staircasing artifacts due to discretization. Recently, PatchMatch based algorithms such as COLMAP [161] have been demonstrated to create the most accurate geometry in benchmark tests [100, 162]. In human-made environments, plane estimation is a central component of many 3D reconstruction algorithms, including for image-based rendering [169]. Several methods [16, 53] use Markov Random Field (MRF) solutions to estimate planes in a multi-view scene, often using higher-level structures. Sinha et al. [168] introduce plane intersections to represent corners. In areas where traditional MVS algorithms fail such as textureless regions, which are frequent in urban environments, some methods can leverage planar priors to obtain qualitative 3D reconstruction [112].

Geometry estimation is the cornerstone of many IBR methods that we discuss in section 2.3. It is also crucial for light transport simulation, as such it is a core component of the Image-Based Rendering and Editing methods we discuss in this thesis.

2.2.3 Intrinsic images and material estimation

Estimating materials from images has many applications. It can be used to previsualize the appearance of manufactured objects using physical samples of the actual material used or for example it can be a way to create realistic content quickly for video games and movies. When estimating a material explicitly one has first to choose a representation for it. There exist a wide variety of representations that can reproduce different sets of materials more or less faithfully (eg.[15, 29, 187]). One of the simplest cases is to assume purely diffuse materials, meaning that the light emitted in any direction is the same: $f_r(x, \omega_i, \omega_o) = f_r(x, \omega_i)$. One can then assume that images are the product of diffuse reflectance and shading, which is often referred as intrinsic image decomposition. The classic Retinex work [110] inspired the intrinsic decomposition method of Weiss [191], which used time-lapse sequences to compute shadow-free reflectance images. Single image decomposition methods [178] initially needed user assistance [18] and now can achieve impressive results automatically. Bonneel et al. [17] recently reviewed the state of the art and discussed the direct applications to image editing. An example taken from Bousseau et al. [18] can be seen in figure 2.4.

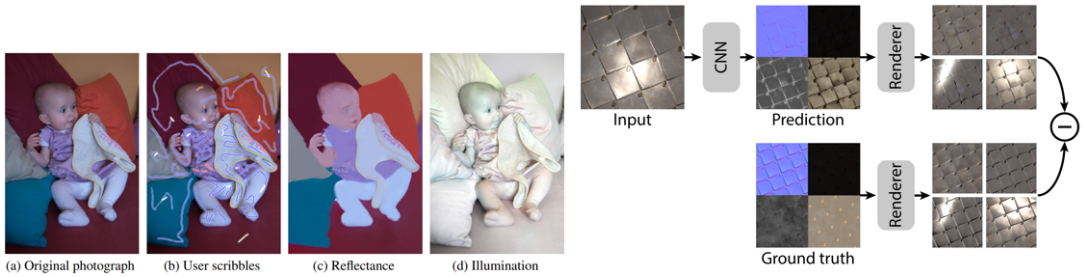


Figure 2.4: Left: Exemple of user-assisted intrinsic image decomposition taken from [18]. Right: Single-shot SVBRF training strategy, with rendering loss. Image from [37].

However, the scenes we encounter in the real world contain glossy materials that violate the "diffuse-only" intrinsic image decomposition assumption. For realistic re-rendering one needs more powerful models. Many solutions exist to estimate spatially-varying bi-directional distribution functions (SVBRDFs). Early optimization-based methods were quite successful for individual objects [116] but required specific capture conditions. Elaborate hardware setups like the Light Stage [36] use multiple lights and/or cameras to record highly detailed accurate representations of complex materials like human skin. Lighter-weight methods, *e.g.*, based on flash/no flash photos [2] can extract complex SVBRDFs under certain assumptions such as repetitive texture. Recently Neural material estimation [37, 119] enabled one-shot SVBRDF estimation from a patch of materials; they are typically trained on synthetic data displaying a first application of "Graphics for Vision for Graphics". Recent methods can even handle full objects [120, 129]. Using mirror renderings, Meka et al. [129] can recover sharp reflections by explicitly supervising a network to produce *mirror images*. Sengupta et al. [164] propose a residual appearance renderer to estimate albedo and normals from a single image, but do not explicitly output glossy BRDF parameters. Li et al. [121] also estimate materials and light from a single image. They use spatially-varying spherical Gaussians as their lighting model.

Finally, Barron and Malik [9, 10] jointly estimate lighting, geometry and materials and shows that all three estimations are closely related and that one task can help the others. Whether it is to differentiate between shading and albedo in the context of shadow removal and relighting, or accurately re-render specularities for view-point interpolation, we will show in this thesis, that understanding materials behaviors is crucial for many multi-view image rendering and editing tasks.

2.3 Rendering the real World from images

Figure 2.5: Stitched panorama from four input views.

We discussed how previous methods try to extract diverse elements of scenes to achieve several goals such has better understanding, scene-editing or re-rendering. What is interesting to note is that in the context of static re-rendering, *i.e.*, rendering a scene from a different viewpoint without modifying its properties, under certain circumstances, some of the three key elements (lighting, geometry, materials) need not be estimated. For example, if the novel-view optical center is the same as that of a captured photo, the image transformation for the overlapping region is a homography. This principle is leveraged for panorama stitching [177]. An example of panorama stitching is presented in figure 2.5. There exist many other ways to use images to re-render a scene from a different view-point; also referred as "virtual camera". We first describe the plenoptic function and introduce light fields, then discuss the first IBR methods before describing in more detail recent IBR algorithms that work from an unstructured set of images, which is the body of work most closely related to this thesis.

2.3.1 Plenoptic Function, Light Fields, First Blending Methods

The plenoptic function [73] can be thought as the dual of the left-hand part of the rendering equation (2.1) for a static scene: $L_e(x, \omega_o)$. As such it is a 5D function, where three dimensions are the 3D position of x' and two dimensions are used to describe ω'_o which is a direction *i.e.*, a unit vector that can be described by two angles, traditionally, θ' and ϕ'. An illustration of the duality between the plenoptic function and $L_e(x, \omega_o)$ is given in figure 2.6.

In practical terms, the plenoptic function describes the incoming radiance to a point

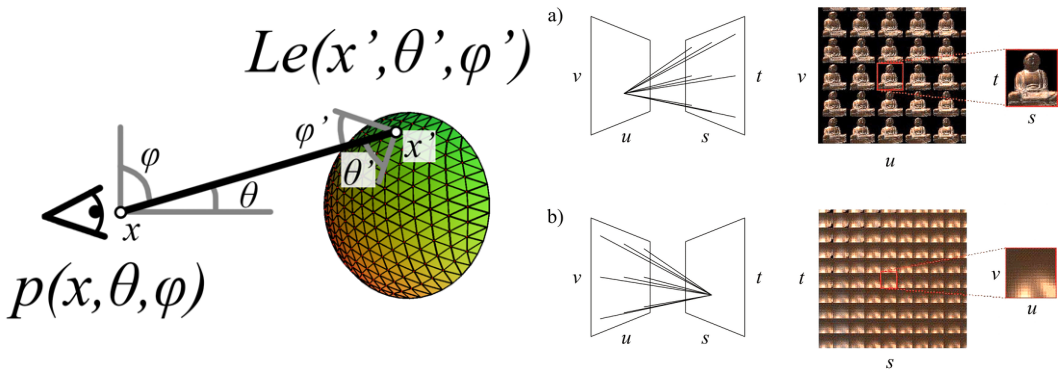


Figure 2.6: Left: illustration of the duality between the plenoptic function and $L_e(x, \omega_o)$. Right: Illustrations of the same duality for light fields. Taken from Levoy and Hanrahan [117].

from a certain direction. With that in mind, capturing a photo is actually sampling this function for each of the visible directions visible from the optical center of the camera. One can notice that in the case where the optical center remains static, some of these directions stay unchanged, leading to the trivial homography transform mentioned earlier. On the other hand, if the points are at a seemingly infinite distance from the camera, the dimensionality is reduced to two, as the point position does not matter anymore, which leads to the two-dimensional environments maps commonly used in computer graphics and previously mentioned with IBL. As we can see, in the general case, capturing only one direction has some useful applications but can not directly be used for view synthesis with a novel camera position. To perfectly reconstruct the plenoptic function one should sample it respecting Shannon's Theorem [165], meaning that the rate of sampling must be at least twice the highest frequency present in the Fourier transform of the plenoptic function. With discontinuities in the signal, this becomes untractable, but with dense capture very good approximations can be made. Early approaches [60, 117] required complex capture setups, making them impractical for widespread use. For instance, Light Fields use an array of cameras, that locally sample the plenoptic function at a high rate, that can then be interpolated. Light Fields reduce the plenoptic function to four dimensions as the geometry is described as a surface. They can be represented from the viewer's point of view, as the plenoptic function, or from the object point of view, as the rendering equation. This is illustrated in figure 2.6.
The Unstructured Lumigraph [19] uses a globally consistent geometric *proxy* and blends reprojected input images, *i.e.*, mixes samples for varying ω_o in the novel view. Assump-

tions about source image positioning have also been considered. For example, floating textures [42] use optical flow in short-baseline video sequences to correct for inaccurate geometry and thus correct the estimation of the sampling. Davis et al. [32] performed bilinear blending of viewpoints located approximately on the surface of a sphere around a captured subject; this assumes the source viewpoints vary smoothly along a 2D manifold. This restricts the sets of viable interpolated camera positions. Most following methods focus on removing the impact of errors in measurements, and lack of sampling. Among other special capture configurations, Arikan et al. [4, 5] present a fast rendering and seam-hiding method for the case of high-quality diffuse scenes imaged using laser scanners. This is also an assumption that reduces the dimensionality of the function, as diffuse materials lead to no variation with respect to the observed direction ω_o.

2.3.2 Superpixels, Per-view geometry, Volumetric approaches

Recently, commercial systems [3][1] [2] deliver high-quality results by capturing data with multi-camera rigs and constraining the virtual viewpoint. As the work in this thesis aims at developing methods to help the spread, usability and number of use cases of Image-Based techniques, we focus on methods that can use an unstructured set of photos to facilitate ease-of-capture. As accurately rendering view-dependent effects often requires dense sampling, methods that focus on free view-point navigation and unstructured inputs often assume more or less diffuse environments which means that the main challenge for them is to have accurate geometry. Global proxy IBR methods (*e.g.*, [19, 42, 73]) are inherently limited in realism by the accuracy of the 3D reconstruction. To address this issue, *per-view representations* have recently been used to maintain accurate image edges during rendering and overcome geometry estimation related issues. These include superpixels [24, 139] or per-view meshes [70]. Illustrations of two of these methods can be found in figure 2.7.

In these solutions, different blending strategies have been used, most of which are based on heuristics [24, 70, 101]. Volumetric representations have also been proposed, Soft3D [143], is based on a regular discretization of space using the input images and a sophisticated blending approach using a *soft* estimation of visibility. Most of the

[1] https://facebook360.fb.com/facebook-surround-360/
[2] https://www.blog.google/products/google-vr/experimenting-light-fields/

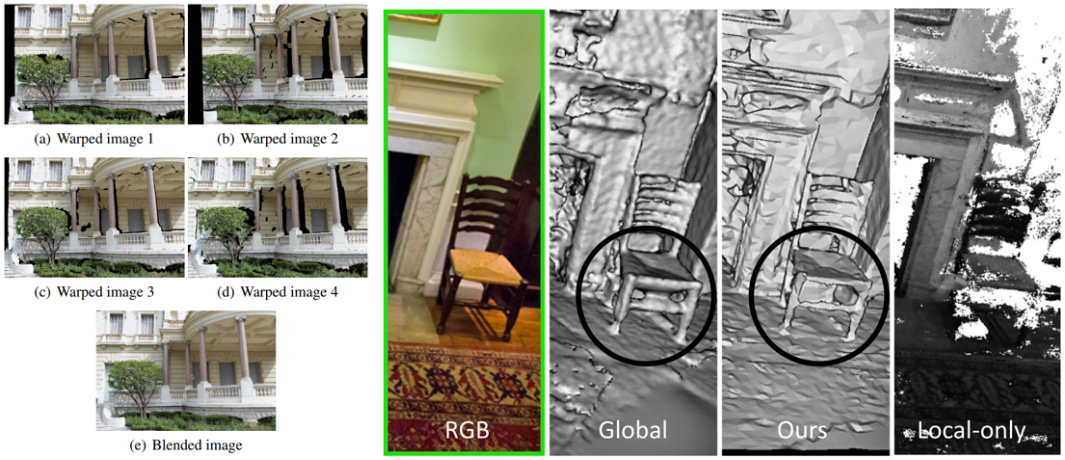


Figure 2.7: Left: Illustration from Chaurasia et al. [24].They use planar superpixel to warp for input viesw and blend them. Right: Hedman et al. [70] refine geometry in a per-view manner using RGB-D images as input. Illustration taken from [70].

time these methods have limited free-viewpoint capabilities, *e.g.*, due to discretization [143], an implicit fronto-parallel superpixel assumption [24, 139], or due to a variety of rendering artifacts that occur for many existing methods, including InsideOut [70].

2.3.3 Learning to render

The early work on image-based priors for IBR [46, 197–199] used a form of learning to synthesize novel views, based on a dictionary of patches from the input images. More recently, Convolutional Neural Networks (CNNs) and deep learning have been applied to the novel view synthesis problem. DeepStereo [47] learns to predict depth and colors using separate "towers" in the network, building on traditional plane-sweep algorithms. Zhou et al. [209], use an encoder-decoder approach to predict the flow field transforming an input image to the novel view. Like many deep learning methods, wide-baseline CNN solutions [47, 209] suffer from visual artifacts that do not provide a sufficient level of realism. There has also been interesting work in learning for view synthesis in the context of Light Fields and small-baseline approaches [94, 172]. More recently, multi-plane images have been used with impressive results [48, 131, 210]. However, the constraints of the narrow-baseline inputs result in very different design choices and it is difficult to see how to directly apply these to our scenario of wide-baseline capture and to the free-viewpoint navigation applications we target in this thesis.

Another neural representation, deep neural textures [182] allows view synthesis for glossy objects, by optimizing deep features in texture space but at the cost of very dense capture. Mildenhall et al. [132] have a different approach to a similar problem. They use a multi-layer perceptron to encode a light field, with Fourier features, from a set of input images as an optimization procedure.

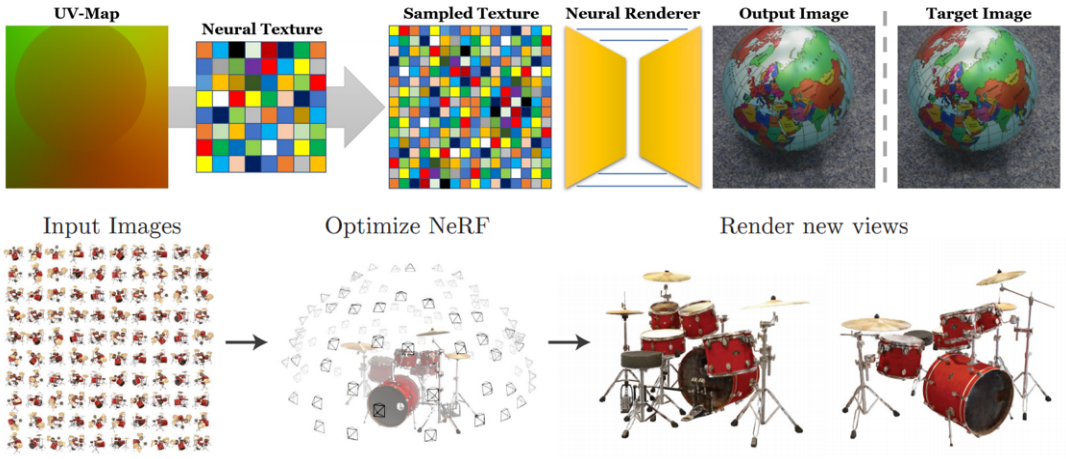


Figure 2.8: Top: Thies et al. [182] use neural textures optimized to reproduce input images allowing to re-render the scene from novel view-points. Bottom: NeRF [132] optimize a neural network to output a density and color for each point and each direction.

Both methods, illustrated in figure 2.8, are closer to what we aim at as they can faithfully render full objects and small scenes. They are still limited, regarding our goals, by the density of capture required, the scale of the rendered scenes and the limited types of motion for which they were designed.

2.4 Re-rendering

We discussed previous methods that try to extract representations from images and the state of the art in Image-Based Rendering. Both of these domains have to be taken into account to achieve the goal of this thesis which is to provide more flexible and editable Image-Based methods. But together they miss a key component which is content editing. While they teach us about the scene properties and how to render them, they do not necessarily give answers on how to modify the properties. In this section we discuss previous work on what we call *re-rendering*; we define this process as *generating a new*

image from existing ones with one or several intrinsic property modifications, such as lighting, geometry, or materials.

A direct solution to re-rendering would arise naturally if we obtained perfect estimations of lighting, geometry, and materials while having a dense enough sampling of the plenoptic function. We could then use the rendering equation and modify the content at wish using traditional asset creation software. While the previously discussed methods often give good enough results for many applications, they are not perfect and the errors over the different elements add up. For instance, the quality of MVS geometry is not good enough to use directly in a path tracer, the defects in geometry are visible and the precision is not good enough. Instead, one needs to factor in the fact that the results of estimations are noisy when designing editing methods. We thus review how researchers tackled image editing problems orienting the discussion towards the goals of this thesis. First, we show how previous work treated *content* editing, discussing object removal and inpainting as well as object insertion in captured scenes. Then we discuss methods that aim at editing the *lighting* of scenes, also known as relighting methods. Finally, we see how recent deep-learning algorithms have been used to tackle similar problems, providing a powerful tool for scene manipulation.

2.4.1 Object Insertion, Object Removal and Inpainting

We previously discussed lighting estimation techniques [55, 76, 107, 134]; most of them work on single images and have as a final goal to composite virtual objects in a real image, which is the first instance of content manipulation, and is a major ingredient of many *augmented reality* applications. Most methods target realistic object editing or compositing in single images but they do not address major lighting changes, such as editing cast shadows and in complex setups, they either require significant effort from the user to annotate the scene images [96, 98] or use information recovered from inserting specific objects into the scene [35]. Illustrations of object insertion from Karsch et al. [96] can be seen in figure 2.9.

In this thesis, we focus on captured content manipulation and do not address virtual object insertion. Some methods to manipulate real-world scenes have been proposed, but operate in a restricted context [206] or rely on drastic simplifications of the scene's geometry [82]. Other solutions are limited by the computational power of the devices

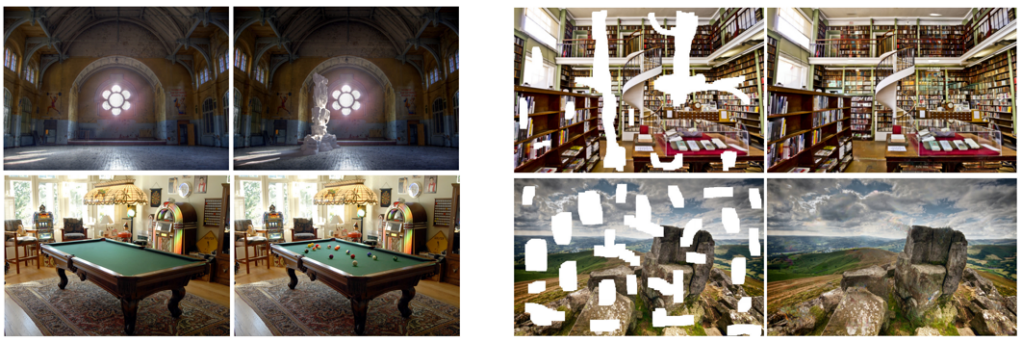


Figure 2.9: Left: Example of user-assisted single image object insertion. Inputs are in the left column, outputs on the right [96]. Right: Results of inpainting taken from Iizuka et al. [83]. Left column: input with removed regions in white. Right column: corresponding outputs.

they use [205] to generate photorealistic images. To the best of our knowledge, no methods exist that allow compositing captured content in captured scenes.

Manipulating captured scenes is a notoriously difficult problem. Another focus in this area has been on removing objects, followed by *inpainting* the regions revealed by removal. This process is sometimes referred to as *decreased-reality* in the context of video feeds or multi-view content. Inpainting is a vast research domain; a good survey can be found in Guillemot and Le Meur [62]. The seminal work of Bertalmio et al. [12] and Criminisi et al. [30] have greatly influenced subsequent work in the fields of computer graphics and vision. Sun et al. [174] use user inputs to better propagate structures during inpainting. More recently, the PatchMatch algorithm [7] introduced efficient solutions for texture synthesis and inpainting. Several improvements have been proposed to the basic algorithm, including Image Melding [31], that identifies and exploits transformations during matching, leading to improved quality. He and Sun [64] further exploit statistics of patch offsets to better guide inpainting. Recently deep neural networks (DNNs) and machine learning have been used for inpainting [83, 203] leading to impressive results visible in figure 2.9. These methods combine global and local context information to achieve good quality results, but have limitations on image resolution and sizes of regions to complete. They are also generally agnostic to the 3D content of the underlying scene, inducing errors in inpainting such as incorrect perspective or errors in planar structures. Previous methods [81, 157] use image analysis to find vanishing lines and induce approximate planar structure or perspective. Video completion is also an

active field of research, that is closer to the multi-view context of this thesis. The work of Wexler et al. [195] introduced the methodological basis for many of the subsequent Expectation-Minimization methods. The recent video-based solution of Newson et al. [136] proposes texture features which improve inpainting quality in many cases. Video-based methods have *dense, small baseline* sequences of frames with rich redundant information, in contrast to the sparse, wide-baseline capture we target in this thesis. Depth information and multi-view data have been used to improve inpainting. The DCSH approach [44] operates on RGBD images, while Howard et al. [80] operate on stereo pairs, as opposed to wide-baseline data. DCSH is based on a local planar approximation of the surface at each pixel that is sensitive to the noise in depth images and is not applicable to missing geometry. Whyte et al. [196] used several photographs of a scene, typically taken from the internet and simple registration between images to improve inpainting. Baek et al. [6] jointly estimate depth and color on an image sequence, but do this progressively from one image to the next. The resulting depthmaps are thus not adapted to a free-viewpoint IBR context. Thonat et al. [183] introduce the first method for multi-view inpainting with output suitable for free-viewpoint IBR. Their approach imposes soft multi-view coherence while inpainting separately in each input image. Finally, inpainting in a multi-view context has some similarities to texture mapping of scenes captured with multi-view stereo (*e.g.*, [11, 21, 52, 186, 208]). A recent approach [13] proposes a patch-based optimization for texture mapping from multiple images. Some of these methods show limited inpainting of small regions on object surfaces, but do not inpaint geometry, which is crucial when removing significant parts of scenes. In this thesis we address two types of editing regarding geometry manipulation, we first introduce a method to remove objects in large scenes allowing to clean IBR environments. We then show how a relighting method can be used to composite realistically different parts of captured scenes.

2.4.2 Relighting, Lighting transfer and shadow removal

Removing or adding objects in scenes mostly involves minor changes to their global appearance, allowing to modify the content in a way that would be impractical in the real world, such as removing cars from a street. Another application that we target in this thesis is to be able to edit the lighting conditions of captured scenes. This is a parameter that is often complex to control when capturing and that is baked in the images. Being

able to capture a scene under only one lighting condition and to render it under many others would give a lot more flexibility to image-based methods.

Image-based relighting methods try to change the lighting conditions of an input image or a set of images. Early work relied on acquiring the intrinsic parameters of the scene either by computing a reflectance model [204] and estimated geometry segmentation [125], or used multiple photographs of the same viewpoint with varying lighting conditions [41, 124]. Marschner and Greenberg [128], used laser scans to estimate geometry. Other methods aim at decomposing images in their intrinsic appearance parameters [178] before computing a new rendering of the viewpoint, with changed illumination. Wu and Saito [200] provide good results on single images, but at the cost of manual scene annotation and geometry estimation. More involved capture setups such as the Light Stage [36, 193], shown in figure 2.10. The Light Stage allows for production-quality relighting, with wide-ranging applications in the film industry by leveraging the linear behavior of light transport and building a basis of lighting.

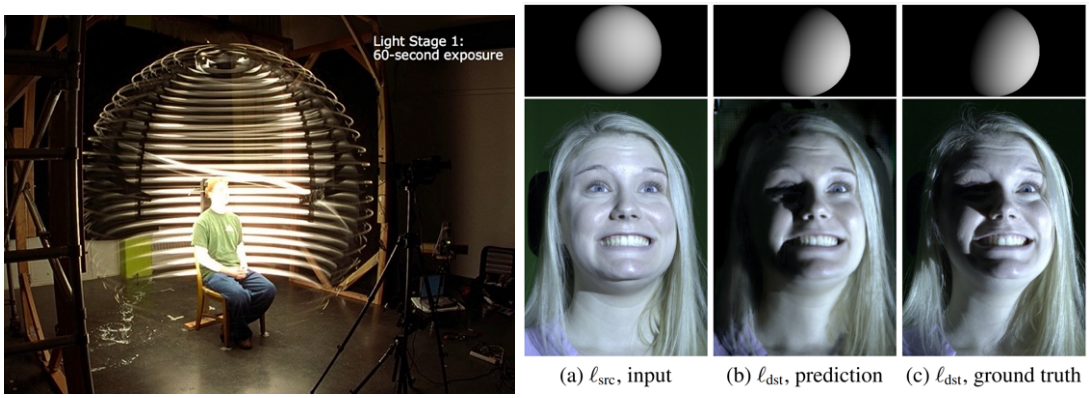


Figure 2.10: Left: The light stage [36]. This hardware helps capture the face of humans by rotation independently a camera and a light source. Right: Illustration of Nestmeyer et al. [135]. Their method allows for realistic face lighting editing.

In this thesis, we target more casual capture with a single camera (DSLR, phone or drone), providing approximate 3D geometry, which is most often unsuitable for inverse rendering methods. Several methods on multi-view image relighting have been developed, both for the case of multiple images sharing single lighting conditions [39], and for images of the same location with multiple lighting conditions (typically from internet photo collections) [104, 201]. The multi-view setting provides additional information such as geometry estimation and multiple viewpoints of each surface as discussed previously. For

the single lighting condition, Duchêne et al. [39], first perform shadow classification and intrinsic decomposition using separate optimization steps. Despite impressive results, artifacts remain especially around shadow boundaries and the relighting method fails beyond limited shadow motion. Webcam sequences have also been used for relighting [108, 176], although cast shadows often require manual layering. Hard shadows are inherently problematic for relighting as they create strong discontinuities, their detection and removal, which is closely linked to relighting has been studied extensively; see Sanin et al. [156] for a survey. Most such methods operate on a single image, for example the work of Finlayson et al. [45], which works well on shadows of relatively simple isolated objects. Other approaches include Lalonde et al. [109] which uses Conditional Random Fields to detect the shadow, or Mohan et al. [133] which is a gradient-based solution for shadow removal.

Recently relighting methods have relied on convolutional neural network architectures to estimate intrinsic images [163], or directly generate the relit images [130], thus avoiding the ambiguous and under-constrained model of intrinsic images. Deep learning also powers object relighting techniques that use multiple lighting conditions as input [201]. Although they provide many interesting intuitions, these methods focus on single images, which means they are not directly compatible with our goal of free-viewpoint 3D navigation which inherently requires multi-view consistency. Another widely developed area of image relighting focuses on images of faces (*e.g.*, [142, 175, 189, 192]). Nestmeyer et al. [135] present a physics-guided approach that incorporate traditional graphics pipeline elements with deep learning leading to very accurate results, visible in figure 2.10. Nonetheless, the specific nature of face geometry and reflectance result in solutions that are not well adapted to the type of scenes we target in this thesis. As we can see relighting was extensively studied for many setups and types of input data, but since the early work of Loscos et al. [124] very few methods tried to tackle the problem of full scene relighting, that has to be solved to give control over lighting in IBR.

2.4.3 Learning to edit

We finally discuss learning methods for image editing as we leverage learned priors for several methods presented in this thesis. Even before the massive adoption of deep CNNs, learning methods were proposed to edit images, for instance, to remove shadows

from images. The method of Guo et al. [63], detects pairs of points in shadow/light using a learning approach, and subsequently removes shadows with an optimization. Recently deep learning strongly impacted image manipulation providing a tool that allowed a large increase in quality and the number of applications. Neural networks that were introduced in the late sixties [86], are optimized using stochastic gradient descent optimization techniques. Instead of computing the gradient over a full dataset, which would be impractical, it is done repeatedly over a small subset. The backpropagation algorithm [113, 194] allows to compute the gradient of all the parameters with respect to a *loss function*, using the chain-rule from the last layers to the first. Even though the theory was developed more than 20 years ago, neural networks only regained popularity in 2012 when AlexNet [102] outperformed all other image classification solutions on the ImageNet competition. More recently, the *Pix2Pix* method [85], illustrated in figure 2.11 used a U-net [155] to perform many different image transformation tasks with remarkable success, even though the quantity of training data is quite low compared to other methods. Similarly, ResNet-like architectures [66] have been particularly successful in large image transformation tasks [211], thanks to the residual blocks that preserve useful information in the network.

Figure 2.11: Left: Results of Isola et al. [85]. Right: Illustration of Meshry et al. [130]. Left column: input buffers. Right column: output renderings. They use SfM and MVS to build a dataset per-scene and then first train an encoder network to represent specific appearance that is an input to a GAN renderer.

There has been a body of work on transforming images, including day-to-night [123] changes, that is akin to relighting. While impressive, the results of these methods typically generated by GANs [150] are lacking in consistency and ease of control. With the advances of deep learning techniques and their capabilities, more and more researchers focused on merging it within graphics pipelines. Tewari et al. [180] review the state of the art of this new domain referred to as *Neural rendering*. They define it as: *"a new class of deep image and video generation approaches that enable explicit or implicit control of scene properties such as illumination, camera parameters, pose, geometry, appearance,*

and semantic structure. It combines generative machine learning techniques with physical knowledge from computer graphics to obtain controllable and photo-realistic outputs." This definition and the goal of this thesis are very much aligned, while having a more specific focus on IBR and multi-view data in the methods we present. We thus review neural rendering methods.

Work on relighting using deep learning (*e.g.*, [95, 167]) fits in this category. Xu et al. [202] can relight single objects from multi-view captures but their acquisition setup requires multiple illumination conditions. Neural re-rendering [130] also takes varying lighting as input using internet images, allowing transitions between different conditions. Chen et al. [26] perform neural rendering based on neural textures for view synthesis and relighting of a single object. Their image formation model consists of environment lighting, intrinsic object attributes and the light transport function, all implemented as trainable networks. Deep Neural Textures [181] allow the user to copy and translate an object in a *single* multi-view dataset. Lightshop [78] allowed compositing of light fields, while recent advances in neural rendering [49] allow compositing of light field videos [40]. Despite impressive advances, neural rendering still struggles with large baselines, single lighting setups available in IBR captures, both in terms of reproducing glossy effects and for free-viewpoint navigation more generally.

2.5 Summary

In this chapter, we reviewed the different domains that we build upon in this thesis. Each of them represents decades of work and could not be described exhaustively. We presented the basics of graphics and physically-based rendering, that are useful for re-rendering. We linked them with their estimation counterparts, describing how lighting, geometry, and materials can be extracted from images. Finally, we presented how these estimations can be used for view-point interpolation, image-based rendering and editings such as object removal, object composition and relighting.

Geometry. We saw that many IBR methods leverage geometric information estimated using SfM and/or MVS, this is also the path we will follow in this thesis as this approach provides consistent, graphics friendly estimates. In Chapter 3, we show that this representation can be used to estimate planar structures that are used both for geometry and

texture inpainting allowing to remove objects in IBR scenes. In Chapter 4 we describe how the noise of the estimation can be overcome using graphics generated image buffers in the context of outdoor relighting. These buffers are more deep-learning friendly than the mesh itself and allow geometry errors to be interpreted locally. Next in Chapter 5 we show how globally consistent geometry can be mixed together with carefully refined depth maps to improve IBR quality. Finally, in Chapter 6 we use this geometry along with PBR to guide rendering and relighting of indoor scenes.

Lighting and materials. Lighting and materials estimations while of good quality, do not address directly the problem of their editing. In this thesis, we do not address material editing which was partially studied by the intrinsic image community. We focus our research on lighting editing which cannot be disentangled from material estimation especially for complex indoor scenes that we consider in Chapter 6. Instead of having an explicit material estimation, for instance of albedo, or glossiness, that could impair relighting quality we opt for an implicit learned representation, targeting finally rendering quality rather then interpretability of the models. In Chapter 4 we explicitly refine CG generated shadow masks for accurate shadow removal and synthesis but without explicit materials estimates. Finally, in Chapter 6, we train a deep neural network to analyze material behaviors to better render specularities for existing and added light sources in our relightable neural image-based render.

This last chapter contains most of the elements discussed in this chapter namely, PBR for lighting simulation through the use of MVS geometry, material analysis for improved specularities, input lighting estimation to guide a neural network for relighting tasks and IBR blending and reprojection for the free-viewpoint aspect. As such, while being only a first step toward more flexible IBR, it builds heavily on of the research work presented in this thesis.

Chapter 3

Plane-Based Multi-View Inpainting for Image-Based Rendering in Large Scenes

3.1 Introduction

We saw in Chapter 2 that recent Image-Based Rendering solutions [22, 70, 144] provide high-quality free-viewpoint navigation, using only a *multi-view dataset* of photos of a 3D scene as input. However, the scene displayed is limited to the content in the input photographs. As mentioned in Chapter 1 this is a major drawback of IBR approaches: capture is easy and they make rendering assets simple but at the cost of the flexibility inherent to traditional approaches. One of the directions we explore in this thesis is content editing as in removing or compositing captured objects.

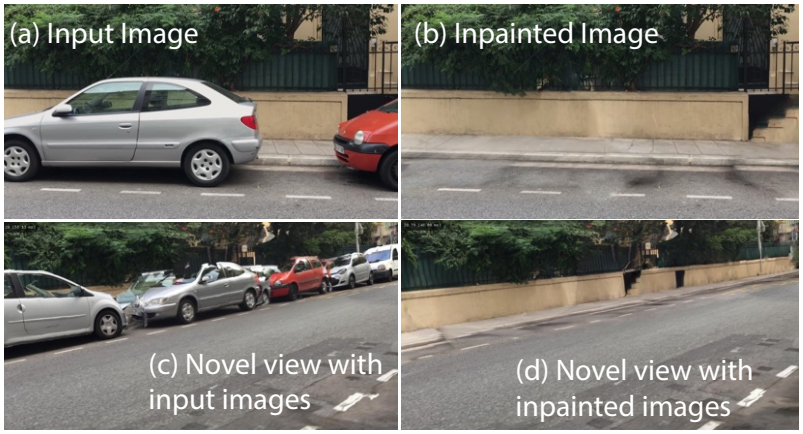


Figure 3.1: Our multi-view inpainting method can remove objects such as cars (b) from all input images (a) of a multi-view dataset for Image-Based Rendering (IBR). This allows more flexible usage of IBR: by removing the cars, we avoid problems due to bad reconstruction which are more visible in novel views (c-d). Our method preserves perspective cues and provides clean separation between different planes (*e.g.*, wall and sidewalk).

For 3D graphics applications, it is often necessary or convenient to remove objects from the scene, *e.g.*, parked cars or furniture in a room; this enables applications such as refurnishing and city exploration but it can also help remove artifacts due to the captured content, like moving people or badly reconstructed objects. This removal can be achieved by *multi-view inpainting*, *i.e.*, by removing the undesired objects in each input image and filling corresponding pixels and depth using inpainting or completion [81, 183]. Single-image inpainting methods [12, 30] are not designed to treat multi-view datasets. The lack of good 3D information inherently limits these single-image methods, which cannot enforce multi-view coherence, nor fully respect perspective during inpainting. When they handle effects such as perspective [81], the results are often impressive, but the lack of good quality 3D information limits their applicability at the scale required for IBR in complex scenes. Recent work provides initial solutions to this problem [6, 183], but suffers from four limitations: 1) multi-view coherence is applied progressively across neighboring images and is often inaccurate or incomplete, 2) perspective effects are not correctly reproduced during inpainting, resulting in visual artifacts and blurring, 3) the quality of depth synthesis is insufficient and 4) the methods are not designed to handle large datasets, since they often use expensive algorithmic solutions operating on all images in the dataset. We target scenes containing man-made structures, corresponding to city blocks or apartments, containing up to hundreds of input images.

The key to overcoming these limitations is to perform inpainting in *intermediate, locally planar* spaces shared between the input images. Our method uses such common rectified planes for inpainting a given region visible in several input images, and thus naturally provides multi-view consistency, strongly encourages correct perspective and provides consistent depth completion. We fit planar segments to each 3D region to be completed and perform inpainting in a plane which can be seen as a fronto-parallel image of each region. Use of planes provides a common reference between images, allowing us to develop a clustering approach for efficient treatment of large scenes. Use of intermediate rectified planar spaces is intuitive and appealing, since it involves locally inpainting in an undistorted image space. However this approach poses two difficult problems that we address with our method. First, we need to identify local planes and create planar regions which have two important properties for multi-view inpainting: the regions must be well oriented and have well-defined edges with respect to the underlying structure of the object being inpainting (*e.g.*, wall, floor, sidewalk). This step needs to be very

efficient, since our goal is to treat hundreds of images. Second, we need to carefully handle inpainting resolution and image resampling, since our algorithm operates in two distinct spaces: the input image and rectified planar space. Our method addresses these challenges in two main steps. The first step is a new planar region extraction algorithm, that finds a small set of planes for multi-view inpainting and efficiently assigns input image pixels to planar regions. The second step uses an intermediate rectified planar space for multi-view inpainting. Our approach performs inpainting in the intermediate planes using a *cascade* of progressively larger resolutions based on constraints in each of the input images in the multi-view dataset. We carefully handle image resampling guided by 3D information in all steps. An example result is shown in Fig. 3.1.

In summary, our contributions are:

- An efficient planar region extraction method that facilitates multi-view inpainting for large datasets.

- A multi-view inpainting method using an intermediate rectified planar space and cascaded resolution. Our inpainting method matches resolution to that in the input images, uses high-quality resampling, structure-preserving initialization and a resolution-dependent distance metric. Together, these elements result in significant improvement in quality compared to previous methods.

Our method includes a clustering approach allowing us to handle large datasets. We present results of our method on indoors and outdoors scenes, and demonstrate significant improvement over previous work, especially in the context of IBR (Fig. 3.1(d), Fig. 3.20, 3.21, supplemental video).

3.2 Overview

Fig. 3.2 presents an overview of our method, with references to the corresponding sections in the text.

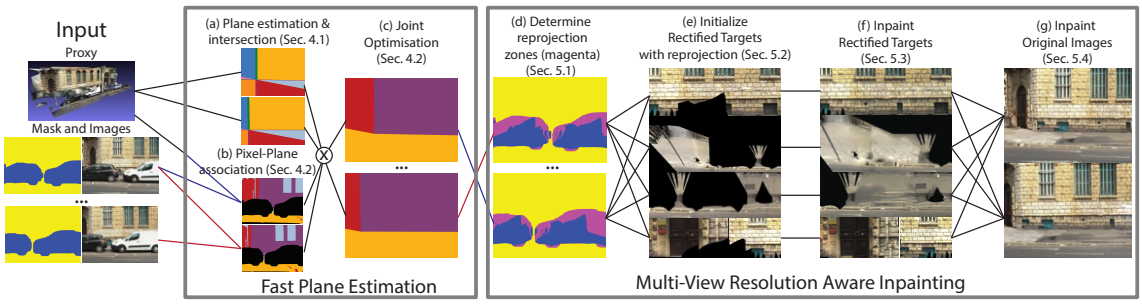


Figure 3.2: Overview of our method. We use a multi-view dataset and the corresponding 3D proxy as input. The first step is a fast algorithm to estimate local planes and assign them to pixels in the input images. In the second step we identify regions to inpaint, and create rectified targets with reprojection. We then inpaint the rectified target images and finally resample them to inpaint the original images.

3.2.1 Input data

The input to our method is a set of images from different viewpoints of a scene containing man-made structures. These are typically photographs, taken with a reasonable "up" vector. We also require masks to identify the zones to inpaint; These can be obtained automatically using *e.g.*, a CNN detector to find bounding boxes for cars, motorbikes or people [57, 122]. Alternatively, an interactive interface can be used to define objects to remove (see Sec. 3.6.1 and video). We use structure from motion (SfM) [170] and multi-view stereo (MVS) [89], to calibrate the input cameras and obtain an approximate 3D mesh or *proxy* of the scene, which is correctly scaled and stored in meters; Fig. 3.2, left. Our algorithm identifies planar regions in the images and operates in *rectified plane space*. We illustrate images, 3D reconstruction and rectified plane space in Fig. 3.3. We call the masked regions in the images *image targets*.

Our main goals are to provide good quality multi-view coherence, preservation of perspective during image inpainting and depth completion, overcoming the limitations of previous methods. We achieve this by using an intermediate planar space for inpainting. The first step is to identify locally planar structures in the scene and create rectified planes π for inpainting. *Image sources* are the pixels of the input images that are not contained in image targets, and the corresponding pixels in the rectified plane are *planar sources*. We process images in clusters, allowing our method to scale to large datasets. Our approach correctly handles interdependencies between clusters, by reprojecting

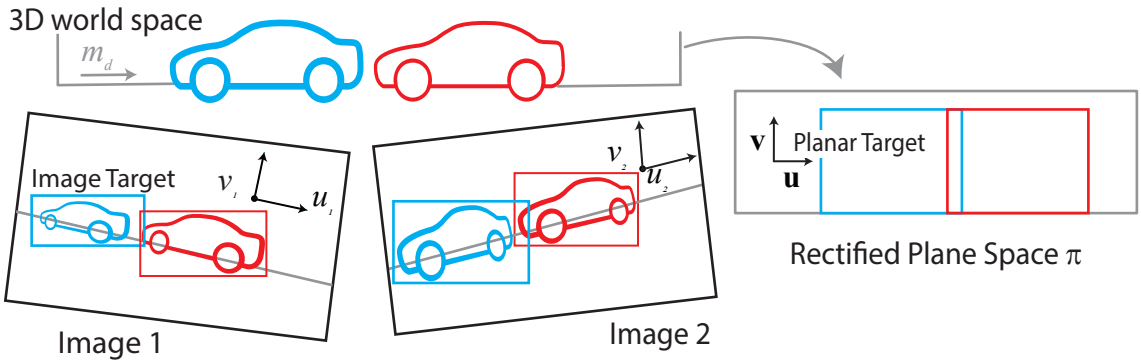


Figure 3.3: Our input is set of images and a multi-view stereo 3D reconstruction, together with the image targets, shown here as bounding boxes. We identify planar structures, *e.g.*, the wall, and inpaint in the rectified plane (right); the inpainting result is then reprojected into the input images.

already inpainted regions into following clusters. For each cluster, our method has two main steps: plane estimation and inpainting.

3.2.2 Plane Estimation

Our main goals for the first step are speed, to allow treatment of large multi-view datasets and quality, especially to provide "clean edges" for corners between man-made locally planar structures. We introduce a fast method to identify planes in the scene and assign pixels of each input image to the corresponding planes; Fig. 3.2(a)-(c).

3.2.3 Multi-View Inpainting

In the second step, we use the intermediate rectified planar space to perform multi-view coherent inpainting that preserves perspective, and provides consistent inpainted depth using the planes; Fig. 3.2(d)-(g). Our approach exploits the multi-resolution nature of our algorithm to inpaint input images only at the required resolution, by introducing a *cascaded resolution* structure. Our inpainting method introduces a structure-aware initialization step and a resolution-dependent term, improving overall quality. Careful treatment is required for resampling and reprojection between the rectified planes and the input images. These are the central components of our efficient and coherent multi-view inpainting algorithm.

We show results of our approach on scenes containing up to hundreds of input images, both for indoors and outdoors scenes containing man-made structures. In Sections 3.3 and 3.4 we present our multi-view inpainting method for a single cluster which can be seen as an independent scene; we present our clustering method in Sec. 3.5.

3.3 Fast Plane Estimation for Inpainting in Rectified Planes

Good quality inpainting requires sharp plane boundaries and careful orientation of the planes, to avoid blur and "bleeding" artifacts between structures. We first estimate a small number of planes that serve as intermediate rectified spaces for multi-view inpainting. We then use our new fast approach to assign pixels to planes while respecting corners or equivalently *plane intersections*.

3.3.1 Estimation and Clustering of Planes

We start with a standard RANSAC plane estimation step, using the 3D points of the reconstruction, similar to previous work (*e.g.*, [16, 53]). We then perform fast hierarchical clustering based on the following distance between two planes π_1 and π_2:

$$d = (1 - \vec{n}_1 \cdot \vec{n}_1) + \chi_E(|d_1 - d_2|) \tag{3.1}$$

where $\vec{n}_i$ and d_i are respectively the normal and the distance of plane π_i to the origin, χ_E is the characteristic function of $E = \{x \in \mathbf{R}^+ | x < \alpha_\pi\}$, and α_π represents a maximum threshold for the variation of the distance to the origin between two planes. In our experiments α_π is set to 30cm in outdoors scenes and 10cm for indoors, which are reasonable thresholds to distinguish different objects for each scene category. The combination of plane estimation and clustering allows us to have a small number of planes while preserving good precision on plane position and orientation.

In the man-made scenes we target, structures are often locally parallel or perpendicular to intersections between planes, *e.g.*, the ground and a wall or the corner of a building. This is similar to Manhattan world assumptions made in some 3D reconstruction algorithms [51, 82]. Our goal is to orient the rectified planes to follow these directional structures; this simplifies the inpainting task, since the patch search and match steps become more reliable. We orient the planes, and define a local basis $\vec{\mathbf{u}}$, $\vec{\mathbf{v}}$ in the rectified plane, in world

coordinates. Given that we focus on man-made structures, inpainting quality depends heavily on the orientation of these vectors. We first determine the best candidate for a "ground" plane, based on the assumption that photos are taken with a reasonable **up** vector. To do so we compute a normalized median vector over the **up** vectors of the input cameras. The ground is the plane whose normal has the highest dot product with this median vector. We can now compute a *main direction* vector $\vec{m_d}$ of the scene which we consider to be the intersection of the ground plane with a predominant vertical structure (*e.g.*, a wall, see Fig. 3.3, top left). We search the set of planes with close-to-vertical orientation, and use the plane with the highest number of points to compute this intersection.

The basis of each plane is then obtained by reprojecting the main direction vector $\vec{m_d}$ on the plane. We use this projection as the first vector to build an orthonormal basis. If this projection norm is too small to be numerically stable, we use the ground normal instead.

The two vectors of the oriented bases are thus:

$$\vec{\mathbf{u}} = \frac{\vec{m_d} - (\vec{n} \cdot \vec{m_d})\vec{n}}{|\vec{m_d} - (\vec{n} \cdot \vec{m_d})\vec{n}|} \text{ and } \vec{\mathbf{v}} = \frac{\vec{n} \times \vec{\mathbf{u}}}{|\vec{n} \times \vec{\mathbf{u}}|} \tag{3.2}$$

We now have the main planar structures of the scene and basis for each plane. The basis vectors can be projected into each image I_i; we denote the image space basis vectors as $\vec{u_i}$ and $\vec{v_i}$. These are defined as follows, for a pixel (x, y):

$$\vec{u}(x, y) \;=\; C(P(x, y) + \vec{\mathbf{u}}) - (x, y) \tag{3.3}$$

where $P(x, y)$ is the 3D point in the plane corresponding to the pixel (x, y) and C is the projection operator for this camera. These three spaces are illustrated in Fig. 3.3.

3.3.2 Assigning Pixels to Planes

We now need to assign pixels to planes to create the intermediate rectified planes for multi-view coherent inpainting. If we compute the 3D position of each source pixel using the camera pose and approximate 3D proxy, and then associate the pixel to its closest plane, we have noisy results in several regions, and in particular at plane boundaries, see Fig. 3.4. For good quality inpainting, it is essential to have clean boundaries, avoiding content being mixed between distinct surfaces.

Figure 3.4: Left: original image. Right: pixel-plane association.

We observe that if we keep a small number of planes, these subdivide the input images into a limited number of zones K, where $K \leq 50$ in our tests. The labelling problem of assigning each zone to a plane can thus be solved efficiently, and we can use the accurate plane intersections for clean boundaries. Plane intersections have been used in the different context of image-based modelling [168]; our solution benefits from the quality of the clean intersection edges, and provides a very efficient solution, avoiding the need for more expensive pixel-based MRF methods [16, 53, 168].

We compute the intersections between all clustered planes in 3D and reproject these intersection lines into the different images to inpaint. For efficiency, we use a bitwise encoding: for each reprojected intersection line, each pixel receives a 0 or 1 code if it is on the left or the right side of the line respectively. This bitwise code separates the images into zones, and two zones are connected if and only if their bitwise code differs by one bit. We can see the intersections and zones in Fig. 3.5.

Each image is now separated into zones and we want to associate a plane to each one of them. To do this we define an energy function. We have the set of zones $Z = \{z_0, z_2 \dots z_n\}$, the set of planes Π and a given plane labeling $L(z_i) \in \Pi$. We first introduce a compactness constraint given by the pixel-plane association for source pixels. If a zone has a majority of pixels associated to one plane we want to encourage the association of the entire zone to this plane. We express this with the following term:

$$e_{\mathbf{z}}(z_i) = \max_{\pi \in \Pi} P_{\mathbf{N}}(z_i, \pi) - P_{\mathbf{N}}(z_i, L(z_i)) \tag{3.4}$$

where $P_{\mathbf{N}}(z, \pi)$ is the number of pixels associated to plane π or any plane parallel to π in zone z. This term implicitly treats visibility, since closer points tend to cover a larger

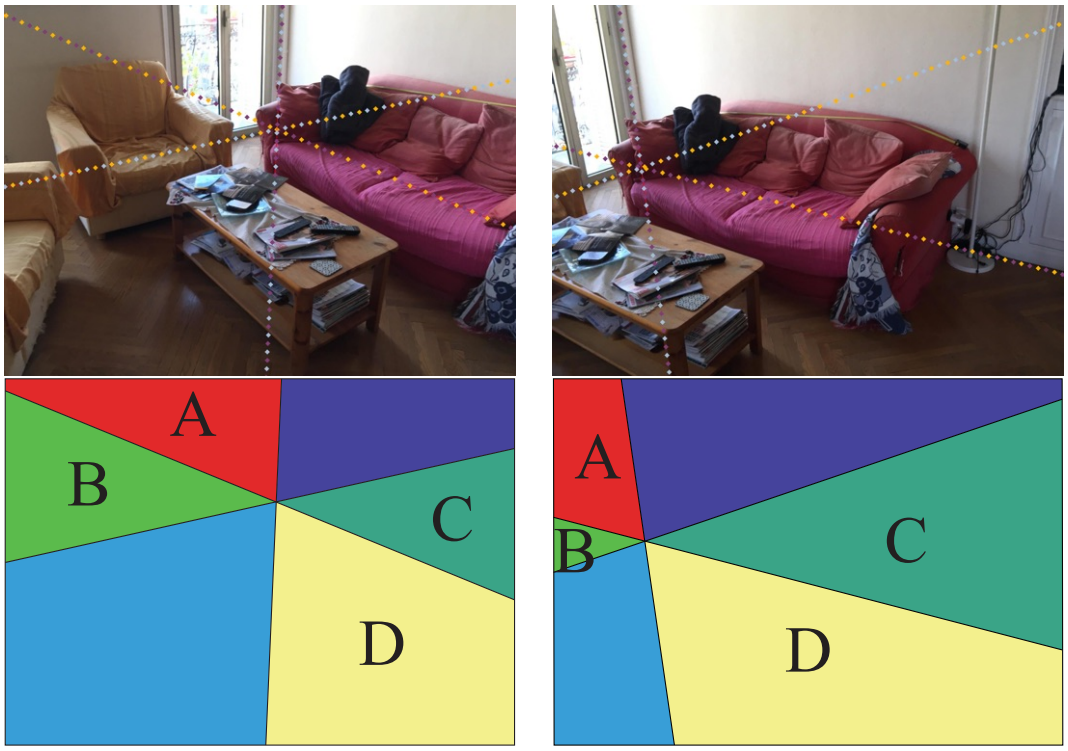


Figure 3.5: Two images of the living room dataset. We see that several zone intersections (*e.g.*, of the zones A and B) are incorrect and need to be removed.

area.

We also encourage zones to be consistent with the plane intersections, while still being able to discard intersections in some cases, *e.g.*, when we have occlusions between planes, the intersections are present both on the visible and occluded side. In Fig. 3.5 the plane intersection line that separates A,B and C,D is relevant only for C and D and should be discarded for A and B. This is expressed with the term:

$$e_c(z_i, l_k, z_j, l_m) = \begin{cases} 0, & \textbf{if } l_k \in C(z_i, z_j) \text{ and } l_m \in C(z_i, z_j) \\ \infty, & \textbf{otherwise} \end{cases} \tag{3.5}$$

where $C(z_i, z_j)$ is the set of the two planes that intersect, forming the intersection line connecting z_i and z_j. Note that label l_k may be equal to l_m in the case where e_c is 0. In Fig. 3.5, this term encourages points in both zones A and B to be assigned the cyan plane (label), while encouraging points in D and C to be assigned to the orange and magenta

label respectively, using the color coding of Fig. 3.4 and 3.6. The final energy is thus:

$$E_z = \sum_{z_i \in Z} e_{\mathbf{z}}(z_i, L(z_i)) + \sum_{\{z_i, z_j\} \in Z_c} e_c(z_i, L(z_i), z_j, L(z_j)) \tag{3.6}$$

where $Z_c = \{\{z_i, z_j\} \mid z_i \text{ and } z_j \text{ are connected }\}$. Since the number of zones is small, less than 100 in all our examples, we use a greedy search in the tree of solutions to optimize. In practice this step is very fast, taking less than 50 ms in all our datasets. Once complete, we have an approximate planar representation of the scene suitable for high-quality rectified inpainting. Results are shown in Fig. 3.6.

Figure 3.6: The result of our fast pixel plane assignment step. Incorrect zones have been removed.

3.4 Multi-View, Resolution-Aware Inpainting

We now have a set of planes associated to a set of images and we can proceed with rectified multi-view inpainting in each plane. A major advantage of having estimated

planes is that we have synthesized 3D positions for all target pixels. These 3D positions allow us to identify target regions that are connected between the images, for example corresponding to the same object seen from different viewpoints. Inpainting is performed in three steps: 1) Creation of the resolution-aware rectified images with target regions, 2) Reprojection to create source pixels for inpainting and 3) Rectified multi-view inpainting.

3.4.1 Creating Resolution-aware Rectified Images for Inpainting

Inpainting in rectified space can be wasteful if done naively, since the final goal is to inpaint the *input* images, which have a given resolution for each image target. Consider the car in Fig. 3.8: due to strong perspective each part of the car occupies a progressively smaller region in the input image I_1. Inpainting at high resolution everywhere in the rectified plane would be wasteful, both in computation and storage of the inpainted images.

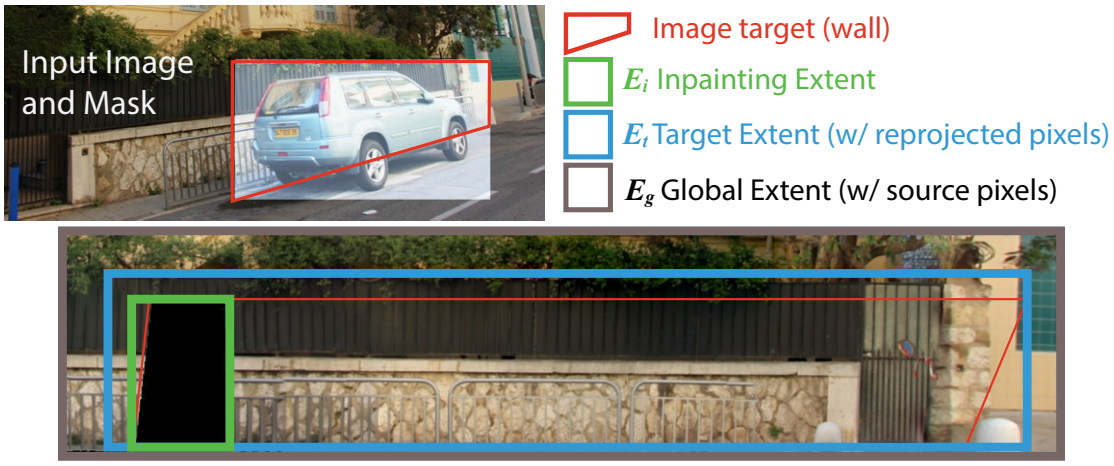


Figure 3.7: The different extents. The red region (wall) is projected into the plane and used to define the three extents: pixels to inpaint E_i, target E_t including reprojected pixels and E_g including the source pixels.

We distinguish three *extents* in each rectified plane π. The inpainting extent E_i, *i.e.*, pixels $\mathcal{P}_i$ in the planar target, corresponding to 3D points in π that are not covered by a source pixel in any input image. We also have the remaining pixels $\mathcal{P}_r$ of the planar target, that can be filled with pixels reprojected from input images. The *target extent* E_t contains E_i and pixels $\mathcal{P}_r$. Finally, we have pixels $\mathcal{P}_s$ taken directly from image sources as *source* pixels for inpainting: E_t and pixels $\mathcal{P}_s$ define the global extent E_g of the rectified planar

image to be inpainted. Example extents are shown in Fig. 3.7. We first explain how we obtain E_i, E_t and E_g and then how we build a *cascade* of different resolutions.

3.4.1.1 Creating the Inpainting and Target Extents.

We first segment each image target into regions corresponding to each plane; *e.g.*, in Fig. 3.7 we split into a region for the wall (shown in red) and for the ground. We determine the pixels to be inpainted for this region in image I_i by reprojecting the pixels into all other images using the corresponding 3D position in the plane. If a point projects outside an image target in another image I_j, the pixel in I_i can be filled by reprojection and belongs to set $\mathcal{P}_r$. Otherwise the pixel is marked to be inpainted and is in $\mathcal{P}_i$. We also check for overlap between all pairs of image inpainting regions in different images. If there is more than 20% overlap, we mark these regions as a group. This ensures that we do not merge regions that barely overlap, and account for mask inaccuracy.

For each group of regions to inpaint, we project pixels to inpaint $\mathcal{P}_i$ into the rectified plane, and find the corresponding bounding box which defines the inpainting extent E_i. We then associate each pixel $\mathcal{P}_r$ to the closest inpainting extent. The bounding box of the union of E_i and associated pixels $\mathcal{P}_r$ is the target extent E_t that is clipped to the boundaries of the reprojected input images (Fig. 3.7).

3.4.1.2 Cascaded Resolution for Rectified Inpainting.

Our inpainting approach is based on PatchMatch [7] which is inherently multi-resolution: inpainting starts at a coarse resolution, followed by upscaling and inpainting at progressively higher resolutions. Since we will adapt resolution to the regions of the input images, we create a *cascade* of rectified planar images, each of which is inpainted only at the required resolution: *e.g.*, in the example of Fig. 3.8, only Level 0 (black) is inpainted at the highest resolution, Level 1 (mid grey) at half resolution and Level 2 (light gray) at the coarsest resolution.

To do this, for each pixel in E_t we determine the required resolution using an approach similar to a mipmap lookup in texture mapping [67], using the directional derivatives $\nabla\vec{u}(P(x,y))$ and $\nabla\vec{v}(P(x,y))$.

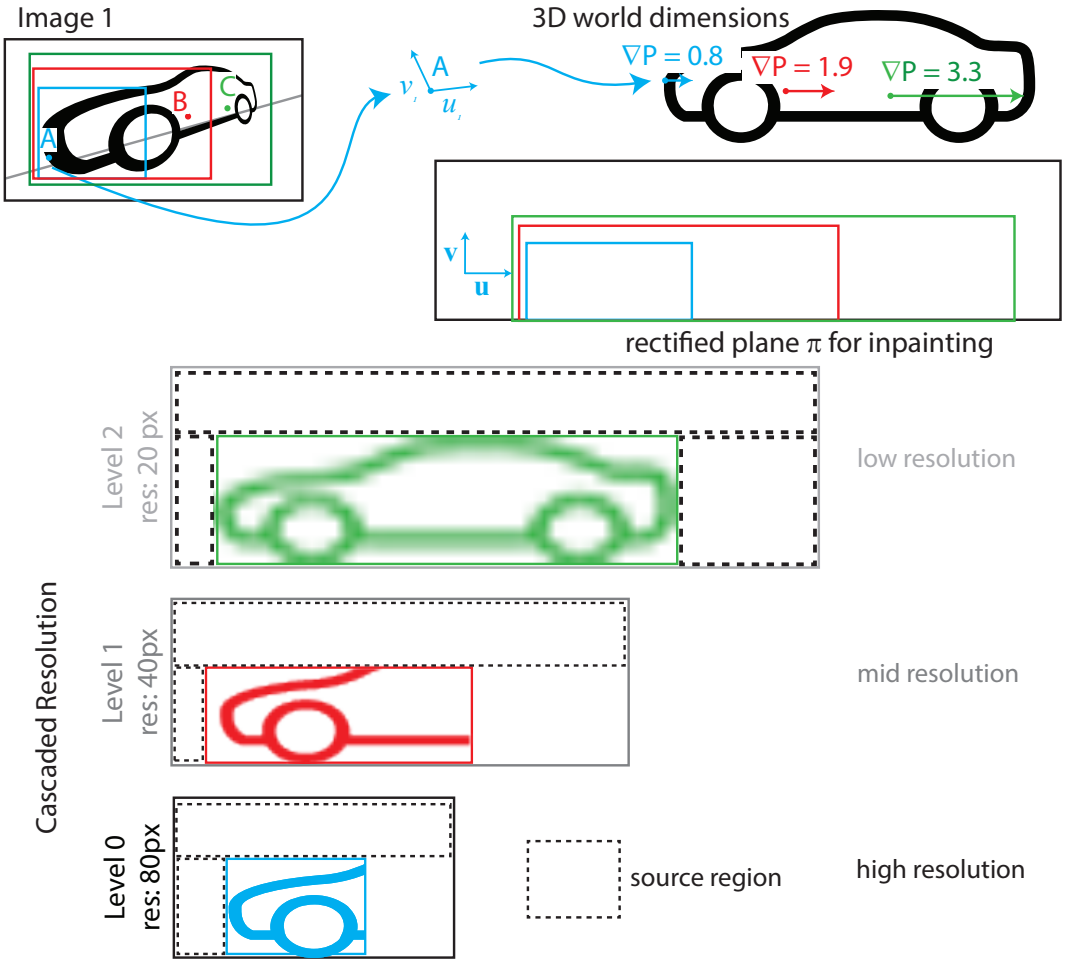


Figure 3.8: Cascaded Resolution: the maximum resolution required is found using the gradient of the rectified basis vectors u an v.

For each pixel $P(x, y)$, we consider two vectors of unit length in pixels, in the directions of the image basis vectors $\vec{u}$ and $\vec{v}$. The normalized vector is: $\vec{u}_n = \frac{\vec{u}}{|\vec{u}|}$. We have the directional derivative $\nabla \vec{u}_n P(x, y)$ (and equivalently $\nabla \vec{v}_n P(x, y)$):

$$\nabla \vec{u}_n P(x, y) = \frac{1}{|\vec{u}|} \nabla \vec{u} P(x, y) \tag{3.7}$$

This provides the magnitude in world space units of a unit pixel displacement in the image in the direction of $\vec{\mathbf{u}}$. This measure is used to choose the corresponding resolution for inpainting each pixel. We first find the pixel with the minimum value of $P_{\min} = \min \nabla P$ which gives the maximum required resolution for inpainting in rectified space. In the example of Fig. 3.8, this value is 0.8 at pixel A.

The specific resolution of each rectified image will be determined based on the value of ∇P and the resolution of the input images. We first determine the maximum resolution required (blue region in Fig. 3.8), and we then create n levels for inpainting, by halving the maximum resolution, corresponding to the minimum value $P_{\min} = \min \nabla P$. In the example of Fig. 3.8, level 0 corresponds to maximum resolution, and will include pixels with ∇P between $P_{\min}$ and $2\ P_{\min}$. Pixels with ∇P between $P_{\min}$ and $4\ P_{\min}$ will be in level 1 etc. (see Fig. 3.8). At the end of this process we perform a region expansion step, and find the region including only level 0 pixels, then the region with level 1, then level 2 etc. resulting in the cascaded resolution structure shown in Fig. 3.8. We ensure that E_t is fully contained in E_g, so that all reprojected pixels are present in the source. We take 50% of the inpainting region size on each side as source pixels $\mathcal{P}_s$. If there is not enough source on one side to reach 50% we increase the amount of source on the other side correspondingly. Example source regions are shown as dashed boxes in Fig. 3.8.

3.4.2 Resampling and Reprojection

We now have the mapping from input images to the target planar regions. Given the projective transformation between the input images and the rectified plane, we need to carefully resample the input images to provide good quality source pixels for inpainting. Each pixel in a rectified image is backprojected into the original images, using Elliptical Weighted Average (EWA) filtering [67] to sample the input images and provide source colors. We call valid rectified pixels those with coordinates within E_g and that are not target pixels. For input image pixels that re-project on valid rectified pixels in more than one image we have to select which image to sample. We formulate this as a labeling problem. We want each pixel to be sampled with a high quality kernel, but also from pixels that correspond to 3D content close to the plane. We thus introduce the following quality term for a pixel p sampled in image I^l, with label l:

$$D_p(I^l) = A_p(I^l) + \gamma D_M(I^l), \tag{3.8}$$

where $A_p(I^l)$ is the inverse of the area of the EWA filter to fill pixel p coming from image I^l [67]. D_M is the distance between the point given by the plane equation and the 3D position obtained with the original depth image associated to I^l, given by the proxy. The first term favors sampling images with large elliptical kernels, which means that we have more accuracy on the sampled values while the second term favors pixels corresponding

to points in the 3D reconstruction closer to the plane (see Fig. 3.9). The factor γ is used to balance the two terms. Ideally $A_p(I^p)$ should be 1 or less, *i.e.*, the sampling kernel for one pixel in rectified space has a surface of one pixel in the image. Since our scenes are in meters, we set $\gamma = 10$ so that a few centimeters deviation from the plane does not have a big impact on the term in Eq. 3.10.

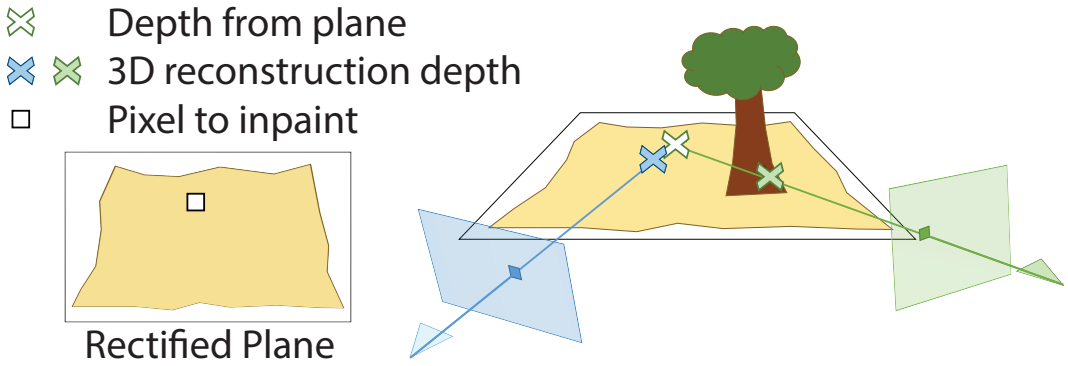


Figure 3.9: The Unary term D_M encourages the use of the depth from the images with depth closer to the plane.

For each pixel we find l which minimizes $D_p(I^l)$. If we only had $A_p(I^l)$, this per pixel minimum would lead to homogeneous source label regions. However the noise in the original proxy breaks this regularity through $D_M(I^l)$, leading to noise in label selection at the boundary of low energy zones. We overcome this issue by solving a multi label MRF only for pixels that do not have homogeneous minima in a 10×10 neighborhood. This occurs for a small number of pixels – less than 10% of the image pixels in our tests – limiting the impact on computation time compared to solving the MRF on the entire image. Similar steps exist in previous work [183, 196], but in contrast to those approaches we have well-defined geometry, allowing us to define the following energy:

$$E_s = \sum_{p \in R_{im}} D_p(I^l) + \sum_{\{p,q\} \in N} V_{p,q}(I^l, I^k) \tag{3.9}$$

with:

$$D_p(I^p) = A_p(I^p) + \gamma D_M(I^p) \tag{3.10}$$

and

$$V_{p,q}(I^p, I^q) = \begin{cases} 0, & \textbf{if } I^p = I^q \\ |I^p(p) - I^q(q)|, & \textbf{otherwise}, \end{cases} \tag{3.11}$$

where V is a standard smoothness term, that helps borders of the zones to be coherent in terms of visual content, and N represents a 4-neighborhood. $A_p(I^p)$ is the inverse of the area of the EWA filter to fill pixel p coming from image I^p [67]. D_M is the distance between $P_{3D}(p)$ given by the plane equation and the 3D position obtained with the original depth image associated to I^p. The first term favors sampling images with large elliptical kernels, which means that we have more accuracy on the sampled values while the factor γ is used to balance the two terms. Ideally $A_p(I^p)$ should be 1, *i.e.*, the sampling kernel for one pixel in rectified space has a surface of one pixel in the image. Since our scenes are in meters, we set $\gamma = 10$ so that a few centimeters deviation from the plane does not have a big impact on the term in Eq. 3.10.

The result of the graph-cut and the local minima of the homogenous zones provides the source for each rectified pixel. We sample these sources by reprojecting the input images into the rectified image. Because pixels coming from different images may have illumination variations we run a Poisson blending step [145] at the border defined by source changes. A planar source and its source labels are shown in Fig. 3.10.

Figure 3.10: Rectified plane initialization. Inset: each color label corresponds to a different input source image.

3.4.3 Inpainting rectified images

For each global extent corresponding to a rectified planar region, we now have the cascaded resolution structure and we can perform inpainting. We use an Expectation-Maximization algorithm similar to previous work ([7, 183, 195]): we first initialize then proceed with several iterations of PatchMatch [7] followed by voting.

3.4.3.1 Initialization

To initialize our images at the lowest scale, we strive to preserve major structures. We represent these structures by strong gradients in the images, and encourage the initialization step to preserve these boundaries.

We first compute straight lines in all filled areas of the image at the highest resolution using a Canny edge detector and a Hough transform. The total number of lines found is N_L. At the lowest resolution, we associate each target pixel p_t with coordinates t to a source pixel p_s with coordinates s randomly sampled according to the following distribution, that discourages sampling pairs (s,t) crossing many lines and thus respects structures in the sources. The sampling density associating a target t to a source s is defined as:

$$p(p_t \leftrightarrow p_s) = \frac{1}{Z_t}\, G(s,t) \quad \text{and} \quad Z_t = \sum_{s \in S} G(s,t) \tag{3.12}$$

with

$$G(s,t) = e^{-0.5 \frac{\|t-s\|_2^2}{((w+h)/2)^2}}\, e^{-0.5(\frac{N_l(p_t,p_s)}{\sigma_l})^2}$$

where w, h are the width and height of the rectified image at lowest resolution, $N_l(X_t, X_s)$ is the number of lines intersecting (t,s) and $\sigma_l = 0.05\ N_L$.

We finally initialize each pixel in the target area E_i by transferring the patch at the associated source area to the target patch and performing mean blending of all patches per pixel.

3.4.3.2 Inpainting

We use the following distance between patches t (target) and s (source) for both the PatchMatch step and voting:

$$d(t,s) = \|t-s\|_2^2 + \lambda_{\text{occ}} \frac{\Omega(s)}{\omega_{\text{best}}} + \lambda_{\text{tfeat}} T(t,s) + \lambda_{\text{res}} R(t,s) \tag{3.13}$$

where $\frac{\Omega(s_i)}{\omega_{\text{best}}}$ is a spatial uniformity ("occurence") term described by Kaspar et al. [97] and $T(t_i, s_i)$ is the texture feature descriptor distance from Newson et al. [136]. We

found that images with many structured features need high uniformity to prevent blur, while imposing uniformity when in less structured regions leads to inconsistent copies of blocks of content (*e.g.*, a piece of one structure in the middle of another).

Instead of setting λ_{occ} manually as in Kaspar et al. [97], we automatically choose λ_{occ} with respect to the mean texture feature norm:

$$\lambda_{\text{occ}} = 0.01(\frac{1}{|S|}\sum_{s\in S}\|Tf(s)\|_2)^2 \tag{3.14}$$

We introduce the term $R(t,s)$ that is a measure of the correspondence of the original resolution between two pixels in the source image. To compute R we reason on rectified source pixels p_r in $\mathcal{P}_s$ and the corresponding pixels p_s in the input images. For each pixel p_r in the initialized rectified image we have the corresponding original input image pixel p_s, and the elliptical sampling kernel e_{s} used to sample p_s. Recall that $A_p = \frac{1}{\text{Area}(e_{\text{s}})}$. For a target pixel we compute A_p for all the input images in which it reprojects and we take the one with the largest area $A_t(I_{\min}(t))$:

$$R(t,s) = \max(0, A_s(I_s) - A_t(I_{\min}(t))) \tag{3.15}$$

given:

$$I_{\min}(t) = \underset{I\in\mathcal{I}}{\text{argmin}}\,\|A_t(I)\| \tag{3.16}$$

where $\mathcal{I}$ is the set of all input images.

This ensures coherent resolution of the patches used to inpaint a specific zone, and also provides sufficient resolution for all the images when we reproject back into the original image space. λ_{res} is set to 0.1 and λ_{tfeat} to 0.001 for all our tests. The different maps for resolution, texture feature and uniformity are shown in Fig. 3.11.

3.4.3.3 Proxy and Depth synthesis

IBR algorithms require a coherent geometric proxy. Since we have removed objects from the scene, the original reconstructed proxy cannot be used. We first create a clean version of the mesh by removing all vertices corresponding to pixels contained in target regions in all images. We do this by projecting all vertices of the mesh into the input images; if a vertex reprojects in a target region in one image it is marked as potentially invalid and if

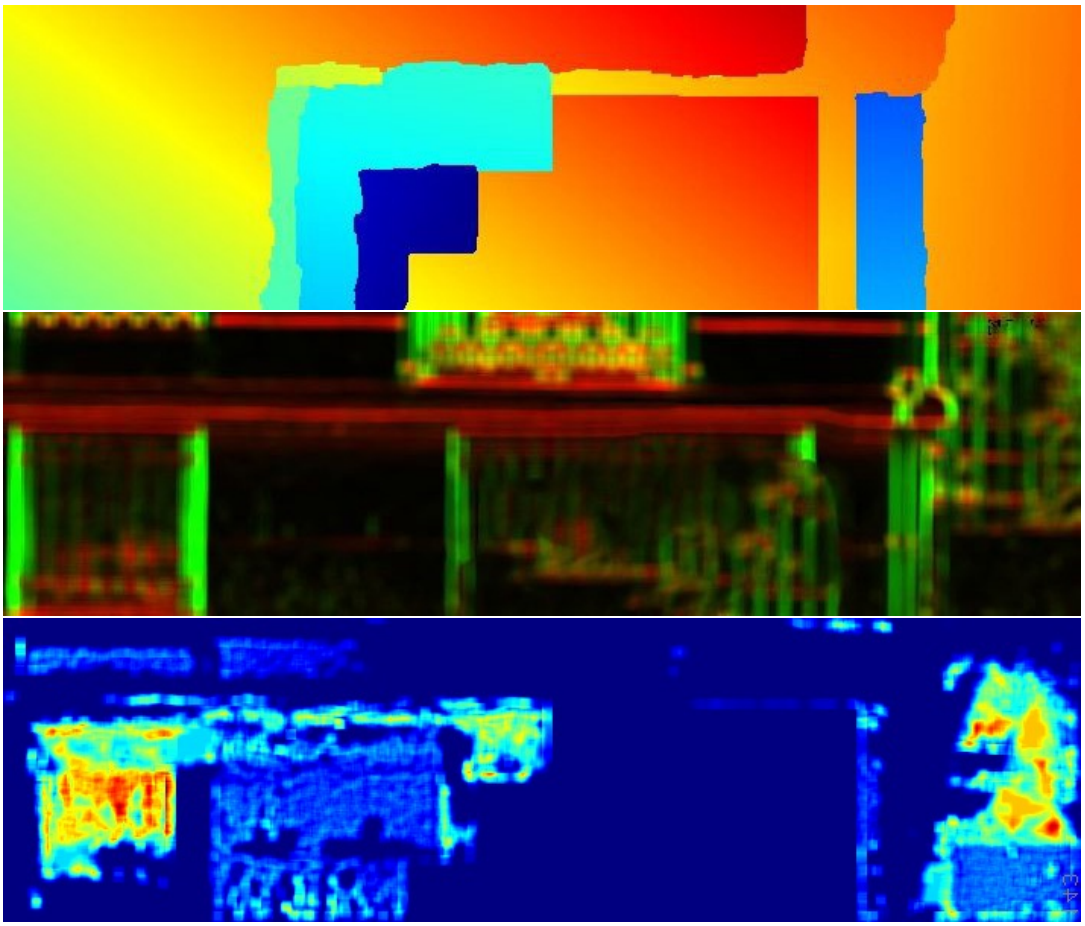


Figure 3.11: The maps corresponding to resolution term, texture feature term and uniformity term. The resolution has only small variations for this rectified target and has almost no impact on the distance term. Texture feature and Uniformity Map are shown at the end of the inpainting. The uniformity map shows how much a source pixel is used.

it reprojects into a source region in one image it is marked as visible. Vertices are that are potentially invalid and not visible are marked as invalid. The clean mesh consists of the triangles that have three valid vertices, and contains holes in regions corresponding to the E_i of the input images.

We fill these holes using the planes of the corresponding rectified images. We create a mesh by inserting a vertex every 10 pixels in the rectified image.

3.4.4 Inpainting input images

After inpainting the rectified planar image, we inpaint the input images by sampling the plane using the same resampling and reprojection approach as for the creation of E_t (Sec. 3.4.2). We compute the elliptical kernels but this time from rectified space to image space. We then apply Poisson blending on the border of the zones to inpaint, to compensate for view-dependent color differences. As a final step we propagate structures that do no not belong to any estimated plane when they are in a source region in at least one image. Specifically, if an inpainted pixel projects on a source pixel in a nearby image with original proxy-based depth closer than the depth of the plane by at least α_π, we sample this source pixel directly.

3.5 Handling Large Datasets

We treat large datasets using a clustering step and reprojection between overlapping clusters. We set a target size of a cluster N_C, depending on the available memory and computing resources. We first compute the clean version of the proxy; this provides a common reference so we can we can treat each cluster separately and thus avoid loading all images in memory at the same time. Next we cluster input cameras with k-means (k = N_C) by 3D position and angle with two-thirds/one-third weights respectively. This creates a set of independent clusters which can be seen as separate scenes. We start by inpainting one of these sets. After treating a set, we reproject the rectified inpainted images of all treated sets into each image of the next set. If a rectified inpainted image reprojects into zones to inpaint, the pixels are sampled in the corresponding rectified image and considered inpainted. For the rectified plane initialization step we add in the original source images, *i.e.*, all images whose cameras see the polygons defined by the rectified images. This maintains and propagates multi-view coherence across sets and additionally avoids inpainting the same 3D zone twice while taking into account all the available information.

Scene	N	Res	N_C	T_p	T_i
Living Room	40	1517x1110	na	16'50"	8'46"
Large Street	290	1828x984	70	45'	28'
Fig. 3.14 & 3.20	16	2016x1512	na	1'49"	2'52"
Fig. 3.15	15	2016x1512	na	2'21"	3'24"
Fig. 3.16	12	1728x1152	na	1'37"	1'26"
Street 1	20	1864x1390	na	2'03"	3'38"
Fig. 3.22 & 3.18	30	2546x1672	na	5'25"	7'40"

Table 3.1: For each scene we list the number N of input images, their resolution Res, the cluster value N_C where applicable, the time T_p for plane processing and T_i for inpainting in minutes.

3.6 Implementation, Results and Experiments

We implemented our method in C++, parallelizing over images and planar targets when possible. All running times are on a PC with a dual Intel Xeon E5-2650 CPU and 64Gb of memory. Plane processing and inpainting are run offline, and create the data that can be used in our IBR system. All IBR results are with a per-pixel Unstructured Lumigraph [19] implementation with soft visibility [42].

3.6.1 Results

We show results for outdoors and indoors scenes. All running time statistics are shown in Table 3.1. In Fig. 3.1, 3.12, we show a Street scene with 290 images, using masks automatically computed with RefineNet [122]. In row 3 of Fig. 3.12 we show pixels filled by reprojection in magenta and by inpainting in blue, illustrating how our method successfully inpaints very large image regions. We also show results for another Street scene from [183] (Street 2) and two additional scenes in figures 3.14 to 3.18 and the accompanying video. In Fig. 3.13, we show a living room scene where the sofa was removed using a interactive tool: the object is removed with one selection in the proxy using a simple 3D viewer such as Meshlab [28] (please see video) and the masks are automatically created in all input images through reprojection. Please see the accompanying video which shows free-viewpoint IBR on all scenes, before and after inpainting.

Figure 3.12: Results from the Large Street dataset for 4 images. First row: input images. Second row: inpainting result. Third row: inpainting result with overlay for reprojected regions (magenta) and inpainted regions (blue). Notice how perspective effects on the sidewalk tiles and the separation between structures (wall, sidewalk) are preserved during inpainting.

Figure 3.13: Results for the living room dataset. Top: Two input photos. Bottom: The corresponding inpainted results. Note that the inpainting respects multi-view coherency.

Figure 3.14: First row: input images. Second row: our results.

Figure 3.15: First row: input images. Second row: our results.

Figure 3.16: YellowHouse dataset from Thonat et al. [183]. First row: input images. Second row: inpainted images by Thonat et al. Third row: our results.

Figure 3.17: Street 1 Dataset. First row: input images. Second row: our results.

Figure 3.18: Inpainting on a curved surface. First row: input images. Second row: our results.

3.6.2 Comparisons

All comparisons shown were generated either using the original authors' code under their guidance ([203]) or by the respective authors themselves (all others). We show comparisons to single image methods in Fig. 3.20. Evidently, our approach has an advantage over these solutions since we have use multiple input images; the main goal of this comparison is to demonstrate that such methods are not well suited to our goal of multi-view inpainting for IBR.

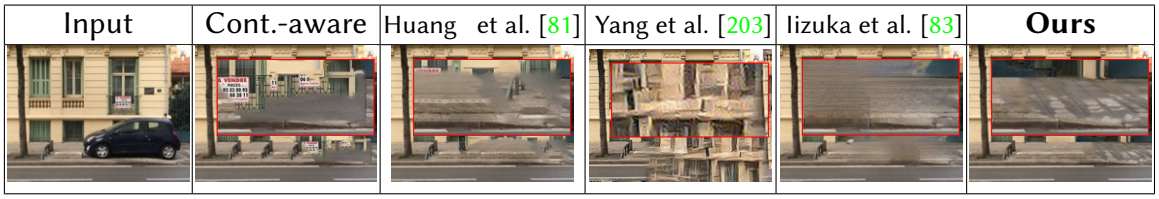


Figure 3.19: Comparison to single image methods. Cont.-aware is the result of Content-Aware fill in Photoshop CC (2016 edition); note that the method of Yang et al. [203] operates on much lower resolution images.

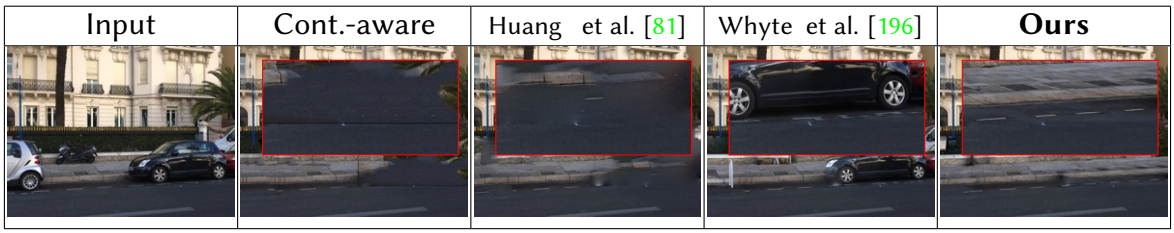


Figure 3.20: Comparison to single image methods and Whyte et al. [196] that does not explicitly synthesize depth. Cont.-aware is the result of Content-Aware fill in Photoshop CC (2016 edition).

We see that our approach preserves perspective and reduces blur significantly compared to these methods. In Fig. 3.21, we show comparisons to methods that treat multi-view datasets, *i.e.*, Whyte et al. [196], Thonat et al. [183] and Baek et al. [6]. We see that our approach overcomes the artifacts related to perspective by better preserving slanted lines (*e.g.*, on the sidewalk) and greatly reduces blur. Our approach also achieves better overall multi-view consistency, *e.g.*, the sidewalk and road in scenes from Thonat et al. [183].

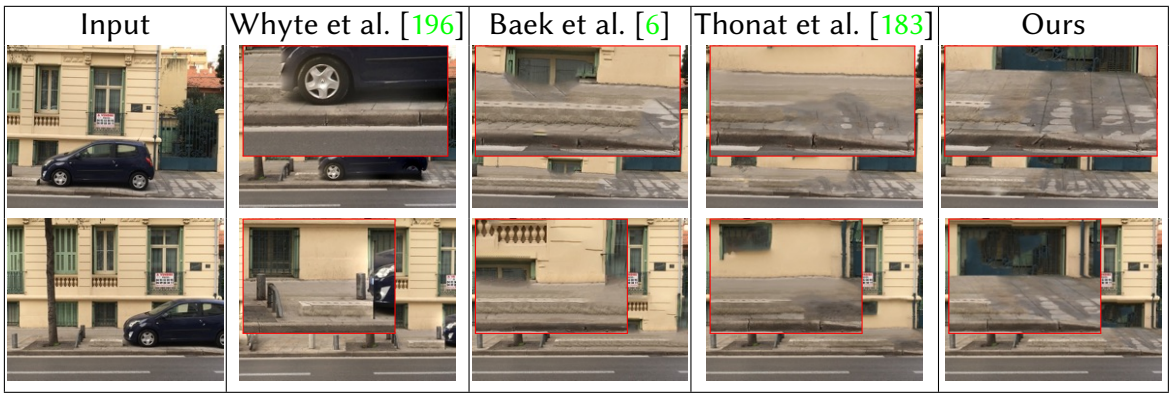


Figure 3.21: Comparison to multi-view methods.

3.7 Limitations and Future Work

Limitations. In Fig. 3.22 we show inpainting on a non-planar surface. The method performs reasonably well overall, but non planar structures are incorrectly copied to a plane, *e.g.*, the curved stairs copied onto the wall.

Figure 3.22: Inpainting on a curved surface.

Another limitation of our approach is related to the number of planes we find during plane estimation. For example, in the Living Room scene, we do not identify the plane of the coffee table with the default parameter settings. This is a user tunable parameter, and can be adapted depending on the application needs.

Future work. In future work, we envisage the use of semantic information to improve the selection of sources for inpainting, thus further improving final image quality overall.

Our clustering approach can be extended to provide good quality source content even when there is no overlap between sets; semantic information will probably be required here as well to ensure good quality transfer of sources between such disjoint clusters.

3.8 Conclusion

In this chapter, we have presented a scalable inpainting method for multi-view datasets, suitable for large-scale IBR, which greatly improves quality compared to previous work. We first present a fast approach to identify planes for inpainting, and assign input image pixels to these planes. Our algorithm enforces clean boundaries between planar regions, encouraging high-quality inpainting. Our approach carefully handles sampling and resolution between the input image and rectified planar spaces, by introducing a cascaded resolution structure used for inpainting. Our inpainting method introduces structure-aware initialization and a resolution term improving overall quality. We also present a clustering approach to handle large datasets. We demonstrated our results on indoors and outdoors scenes with up to hundreds of input images. Our results show significant overall improvement in inpainting quality, providing a usable solution for manipulating IBR environments. Our method also provides a a way to overcome difficulties related to certain capture conditions, *e.g.*, moving people or cars that cause ghosting artifacts, specular objects for which IBR has issues; these can all be removed from the original content after the fact.

Another element of capture that is often not controlled is the lighting setup. In traditional IBR we are limited to the lighting condition at the time of capture; as mentioned in the introduction, this is a major limitation of such methods. The following chapter introduces a method to relight outdoor multi-view data after the fact allowing more flexible captures and even more content editing, providing a first step towards removing this severe limitation.

Chapter 4

Multi-view Relighting and Scene Compositing using a Geometry-Aware Network

4.1 Introduction

We have seen previously that content modification is crucial to add flexibility to image-based rendering methods. In the context of multi-view data editing we now explore lighting modifications for outdoor scenes. As discussed in Chapters 1 & 2 lighting is one of the main components that would benefit from being edited after capture. Indeed, lighting is hard to control when capturing a scene or simply taking a photo. Professionals use specific hardware and know when to capture specific landmarks to obtain the best shots. This is not compatible with the casual capture we target in this thesis. We thus propose a method allowing the user to alter the lighting of captured

(a) our algorithm can relight a single-illumination drone video dynamically to synthesize a "time-lapse" effect

(b) single-view input

(c) three relit outputs: here we built the *proxy* geometry using internet photos of the same location

Figure 4.1: Two applications of our multi-view relighting system.(a) We show five different frames from a drone video (copyright Namyeska youtu.be/JHeDP7_YBos used with permission) relit with a "time-lapse" effect of a rotating sun (full video at youtu.be/C_V8UpZYAQU). A user can also relight a photograph of a known landmark (b) to different target lighting conditions (c). For this, we applied our algorithm to a collection of 50 internet images of the same location.

pictures after the fact. Here we focus on outdoor scenes. Changing the illumination of an outdoor image is a notoriously difficult problem that requires the lighting to be modified consistently across the image, and shadows to be removed and resynthesized for the new sun position [39, 179, 204]. Cast shadows are particularly challenging because an occluder can be arbitrarily far from the point it shadows, or even out of view.

The basic premise of our approach is to use multi-view information and approximate 3D geometry to reason about non-local lighting interactions and guide the relighting task. In Chapter 3, multi-view imposed strong constraints inpainting, such as multi-view coherence, and was one of the core problems the method deals with. For relighting, on the contrary, we argue that having access to multi-view and geometry is one of the key elements allowing us to solve the problem. We introduce the first learning-based algorithm that can relight multi-view datasets of outdoor scenes (Fig. 4.1), which have become a commodity thanks to smartphone cameras, large-scale internet photo collections and drone cinematography. Our model uses a neural network designed to exploit geometric cues. It includes a careful treatment of cast shadows and is trained solely on realistic synthetic renderings.

Our method has several applications: it allows automatic creation of a "time-lapse" effect by dynamically relighting drone videos (Fig. 4.1(a)). Or, if we only have a single photo, we can access online photos of the same place to relight the input photo (Fig. 4.1(b)). We can also relight images in traditional multi-view pipelines, *e.g.*, Image-Based Rendering (IBR) or photogrammetry (Fig. 4.20). In section 4.6 we also show how the relighting network can be used to composite real scenes captured under different lighting conditions. We introduce a two step method to do this compositing which has the advantage of using the already trained relighting network.

Previous methods have difficulty with the type of input we target. Inverse-illumination methods [124, 204] cannot handle the approximate geometry of the proxy, while single-image relighting solutions struggle with cast shadows [126, 166]. Finally, our solution significantly outperforms neural-network baselines (Sec. 4.5.2).

Contributions. In summary, in this chapter we present the following contributions:

- An end-to-end learning method for multi-view relighting of outdoor scenes, guided by image-space buffers, namely shadow masks and illumination buffers, that are computed from a geometry proxy.
- A learning-based shadow refinement solution to remove and resynthesize shadows. It uses the input images as well as our newly-introduced RGB shadow images to overcome reconstruction errors in the proxy.
- A training procedure that uses realistic synthetic scenes to flexibly generate multiple lighting conditions. Critically, we create a stereo-based proxy for each training scene which, together with the ground truth geometry, enables supervised learning for shadow refinement.
- A compositing algorithm that leverages the trained relighting network to realistically insert parts of captured scenes into other captured scenes.

Although it is entirely trained on synthetic images, our algorithm generalizes to real multi-view datasets, and can modify the lighting in a much wider range of illumination conditions than previous methods (*e.g.*, [39]). We evaluate our approach on real multi-view datasets, and show a variety of applications (Fig. 4.1,4.17,4.18. Sec. 4.6).

4.2 Overview

Given a set of images captured from multiple viewpoints (Fig. 4.2a), we start by building an approximate representation of the scene's geometry — a *proxy* — using off-the-shelf stereo [152, 171] (Fig. 4.2b). We can relight any *reference* view of that scene (Fig. 4.2c) — this could be one of the input images or a novel view obtained by IBR. The user provides a target illumination by specifying a sun direction vector and a scalar "cloudiness" level (or a sequence of such parameters for "time-lapse" effects). From the proxy, we then compute image-space *buffers* (Fig. 4.2d: normal maps, specular reflection direction, etc.) and shadow masks for the source and target illuminations. We perform relighting by training a neural network to map from the reference image, with extra buffers and shadow masks, to the novel lighting condition.

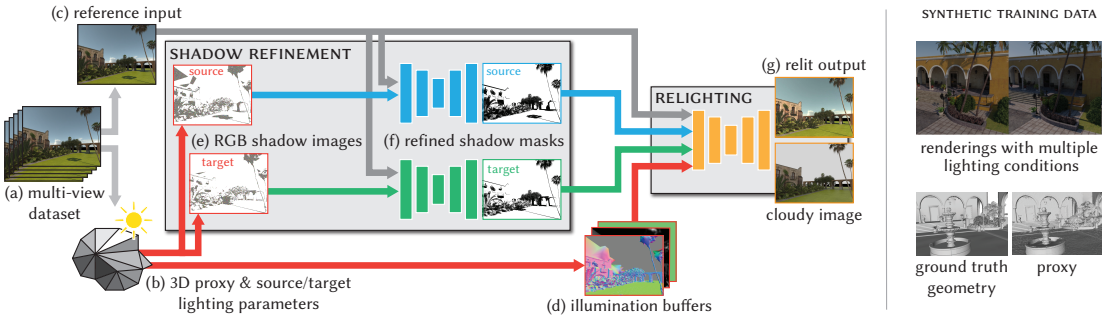


Figure 4.2: Overview of our approach. **Left:** We use off-the-shelf stereo to create a 3D geometric proxy of the scene (b). The geometry is encoded as illumination buffers (d) and used to create RGB shadow images (e) that are independently refined by two networks (f), helping the final relighting network remove and re-synthesize shadows, and change the illumination (g) according to the desired novel lighting condition. **Right:** We train our model with synthetic data, including accurate ground truth geometry and renderings and an approximate proxy, created using synthetic renderings instead of photos. These two representations of the training scene allow the network to accurately refine shadows, enabling plausible relighting.

The importance of *accurate* shadow estimation for shadow removal has been previously demonstrated [39, 61, 63]. But reconstruction errors in the proxy often lead to inacurrate masks a network cannot trust. This motivates our network design: we decompose our model into three sub-networks (Fig. 4.2). Two modules *refine* the source (resp. target) shadow masks (Fig. 4.2f) while the third implements the final relighting (Fig. 4.2g). The sub-networks are trained jointly but with different supervision: respectively ground truth shadow masks and ground truth relit images. Furthermore, instead of computing standard shadow masks from the proxy, we introduce *RGB shadow images* (Fig. 4.2e). These shadow images re-project colors from the shadow-casting geometry from all viewpoints into pixels in shadow, helping the network identify erroneously reconstructed shadow casters from the reprojected color (Fig. 4.4,4.5).

For supervised training, we need data corresponding to different lighting conditions of the exact same views, that is hard to capture with real photos. Instead, we use professionally-modeled, realistic synthetic scenes to generate physically-based renderings with many different viewpoints and lighting conditions. We introduce a flexible compositing methodology to generate a large variety of illuminations on-the-fly at training time. This avoids the combinatorial explosion in the number of images to render. Synthetic scenes also give us ground truth shadow masks.

To train the shadow refinement, it is impossible to capture real data and we cannot directly use the ground truth shadows cast by the synthetic geometry. A model trained with these perfectly accurate shadows would not generalize to real photographs, since it would have never seen the reconstruction errors of the stereo-based proxy. Instead, we generate an approximate 3D proxy for each synthetic training scene using stereo on renderings, from which we obtain the input illumination buffers and (inaccurate) RGB shadow images. The ground truth shadow masks are used as targets to supervise the refinement sub-networks. This approach makes our model robust to 3D reconstruction errors at test time and limits the generalization gap between real and synthetic data.

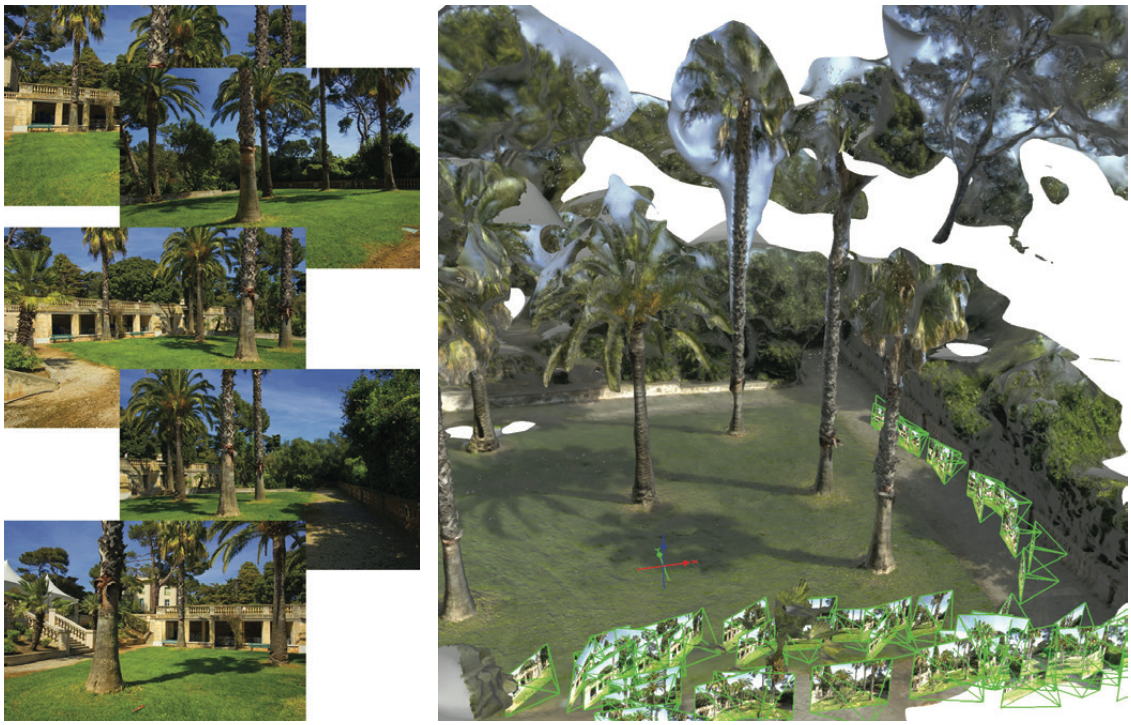
4.3 Geometry-aware relighting network

Our relighting solution is built around a neural network that takes one image from a multi-view dataset, and a set of corresponding image-space buffers as input, and produces a new image, with the lighting altered. We identified three key difficulties to successfully implement this image transformation: modeling the *illumination* changes (color, intensity, etc), and *removing* and *resynthesizing* cast shadows.

To overcome these difficulties, our learning solution exploits a *geometric 3D proxy* which we obtain by first calibrating the input virtual cameras using structure from motion (SfM) [171], then running a Multi-View Stereo algorithm [59, 152]. Fig. 4.3 illustrates this procedure.

Because our CNN operates in the image domain, we encode the geometry and lighting parameters as image-space *illumination buffers* B. These include normal maps, per-pixel specular reflection direction, etc. See Section 4.3.3. In our ablation study, we found these buffers to be instrumental in synthesizing plausible novel illuminations (Section 4.5.5).

Furthermore, the proxy gives us a particularly powerful means to guide the shadow removal and re-synthesis process. We use it to obtain two shadow masks, S_{src} and S_{tgt}, corresponding to the source and target sun directions respectively, by running a shadowcasting algorithm. If the geometry were perfect, these masks would tell the network precisely which pixels to brighten (resp. darken). However, because of errors in the stereo reconstruction, the masks typically contain significant artifacts and misalignments with respect to the actual shadows in the image.



(a) input views (b) calibrated cameras and 3D proxy

Figure 4.3: (a) Our method takes as input a set of photos of an outdoor scene, shot from varying viewpoints (in this example 140). (b) We calibrate the cameras (shown in green) and build a 3D proxy of the scene using MVS. This reconstruction is approximate, as can be seen from the multiple holes (white) and erroneous over-reconstruction (*e.g.*, blobs around palm trees with reconstructed sky). Our model learns to account for this uncertainty and generalizes well at test time.

While coarse masks are better than no shadow mask at all (see Section 4.5.5), we found that the success of the shadow removal procedure strongly depends on the quality of S_{src}. Similarly, the shadow re-synthesis suffers from errors in S_{tgt}. This led us to build an explicit *shadow refinement* step within our pipeline. We guide the refinement step by introducing *RGB shadow images*. These maps use color information from all images in the multi-view dataset to provide hints to the CNN on reconstruction inaccuracies.

Our overall model can thus be divided into three sub-components (Fig. 4.2). Two sub-networks independently refine the shadow masks S_{src} and S_{tgt}, and a third implements the final relighting given the illumination buffers and the refined shadow masks. The three components are trained jointly in an end-to-end, supervised fashion, using a

training set of synthetic scenes. Our dataset contains ground truth source/target images, and approximate/ground truth shadow mask pairs.

4.3.1 Overall architecture

At a high-level our network is the composition of three sub-networks, two for the source (resp. target) shadow refinement tasks and one for relighting (Fig. 4.2). The refinement networks both take the RGB shadow images (Section 4.3.2.1) and the input images and predict refined greyscale shadow masks. These two refined shadow masks, along with the illumination buffers, are sent to the relighting sub-network which infers the target sun condition image and an overcast image. This 3-step approach is supported by recent results (*e.g.*, [188]) showing that decomposing shadow detection and removal in two consecutive subtasks within the same network greatly improves quality. The overall architecture of our network is shown in Fig. 4.2; we use a ResNet [66, 90] for the shadow refinement and the relighting modules [211]. We also experimented with a Unet-like architecture [85], that gave marginally inferior results. Our network outputs two images: the relit target image, and a "cloudy" rendering which we use to produce different degrees of overcast lighting conditions (Section 4.5.6.1).

4.3.2 Shadow refinement with RGB shadow images

Strong shadow cues are central to the shadow removal and re-synthesis process (see Section 4.5.5 for a comparison). The proxy can be used to compute standard (greyscale) *shadow masks* (Fig. 4.4 (c), black pixels are occluded by the geometry). We found however that, because the proxy is only approximate, the shadow masks are usually too coarse, which motivates our shadow refinement pipeline. To reap the most benefits from this refinement, we introduce a novel representation — RGB shadow images — that is robust to inaccurate geometry (Section 4.3.2.2).

4.3.2.1 Independent source and target shadow refinement

Both source and target shadow maps are obtained from the proxy, and therefore need refinement (Fig. 4.4). We process the two maps *independently*, with two sub-networks that perform fundamentally different tasks.

Refining the source masks is an easier problem because the shadows in the input image are in exact correspondence with the shadow mask: the refinement network can use the image as guide.

This does not apply to the target masks. Since we want to change the lighting, the target masks are generally not aligned with the shadows in the input image, making the problem inherently more ambiguous. If instead, we used the *same* shared network for both tasks, the quality of the refined source shadows would degrade. Unlike specialized modules, a shared network cannot expect the masks to be consistently aligned with the image data.

We use synthetic data to create ground truth / proxy pairs for shadow refinement (Section 4.4.3). The source shadow refinement process uses the actual boundary in the input image, giving better overall results compared to the target shadow refinement (Fig. 4.4, (e)).

4.3.2.2 RGB shadow images

We introduce *RGB shadow images* to guide the refinement of both source and target shadow masks: this is a key element to the success of our solution. RGB shadow images S^{rgb} (Fig. 4.4, (d)) are created by reprojecting colors from all the other images in the multiview input.

Their purpose is to help the network recover from over-reconstruction errors, *e.g.*, the beams of the pergola in Fig. 4.4 appear connected as a solid ceiling. Our RGB shadow images will show blue pixels in the (incorrectly) shaded area corresponding to the sky, which easily disambiguates this error (compare (c) and (d) in Fig. 4.4).

We illustrate the computation of RGB shadow images in Fig. 4.5. For each pixel in shadow, we cast a ray in the direction of the sun $\boldsymbol{d}_{\text{sun}}$ from the corresponding 3D scene point $\boldsymbol{x} \in \mathbb{R}^3$ (see Fig. 4.5). The ray intersects the occluding proxy geometry at a point $\boldsymbol{x}_{\text{o}}$, that we reproject into the other input images. We accumulate a weighted average color collected from the other views. The weight for the contribution of a given image i to the color of a pixel in the RGB shadow image is computed as:

$$\frac{1}{||\boldsymbol{x}_{\text{o}} - \boldsymbol{p}_i(\boldsymbol{x}_{\text{o}})||_2^2 \cdot |1 + \boldsymbol{c}_i^{\intercal} \boldsymbol{d}_{\text{sun}}|^2 + \epsilon}, \tag{4.1}$$

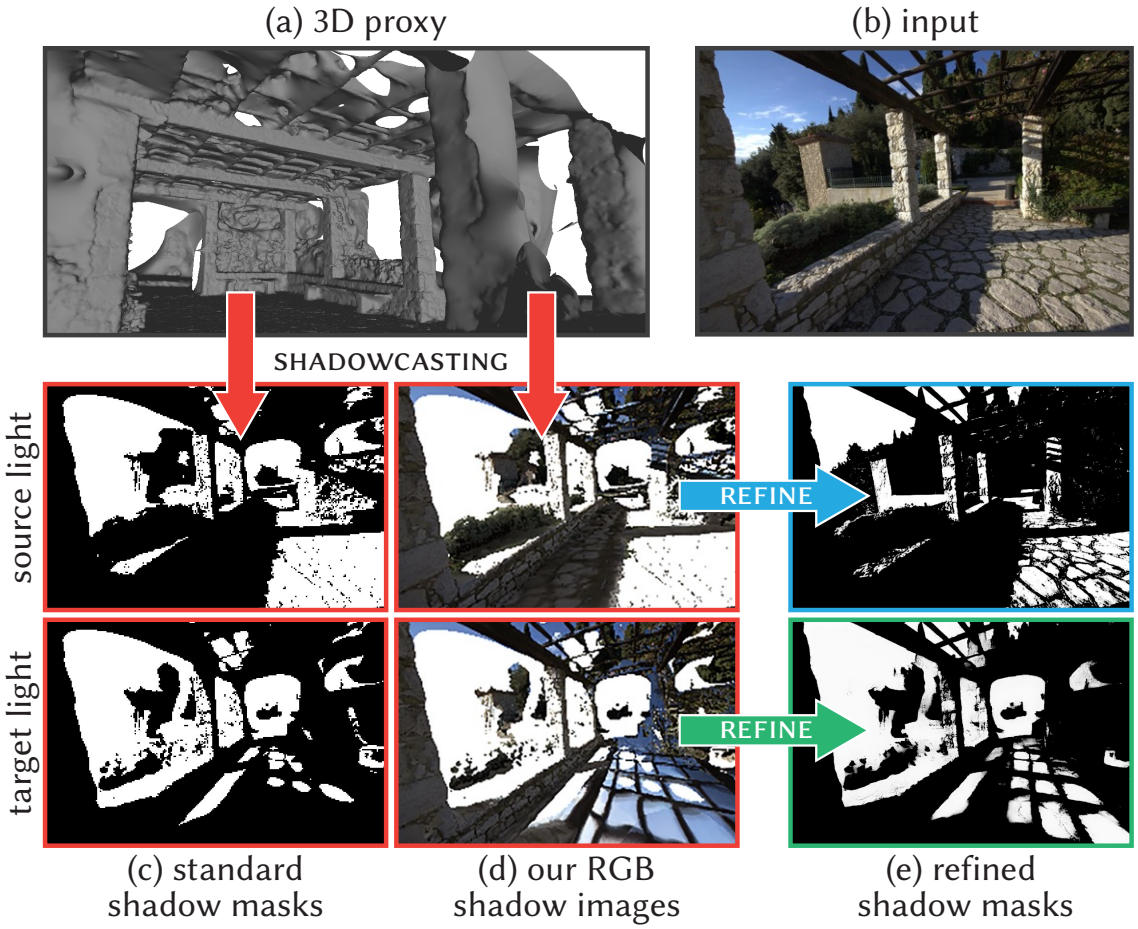


Figure 4.4: We use the 3D proxy (a) to cast shadow masks corresponding to the *source* and *target* lighting conditions. Traditional shadow masks (c) already provide strong cues for a relighting model, but they often suffer from errors in the multi-view reconstruction. Our new RGB shadow images (d) are more expressive and help us recover from the proxy's errors. We process them with two independent shadow-refinement subnetworks to obtain finer shadow masks (e). In turn, these refined masks guide the removal of shadows in the input (b), and the synthesis of detailed cast shadows for the new lighting condition.

where $\boldsymbol{c}_i$ is a unit vector giving the direction from camera i to $\boldsymbol{x}_o$, $\boldsymbol{p}_i(\boldsymbol{x}_o) \in \mathbb{R}^3$ is the first intersection of the camera ray defined by $\boldsymbol{c}_i$ with the proxy (Fig. 4.5) and $\epsilon = 1e{-}5$. The first term reduces contributions of images i when an object occludes $\boldsymbol{x}_o$ from the point of view of camera i.

The second term tries to reduce reprojection error due to depth inaccuracy.

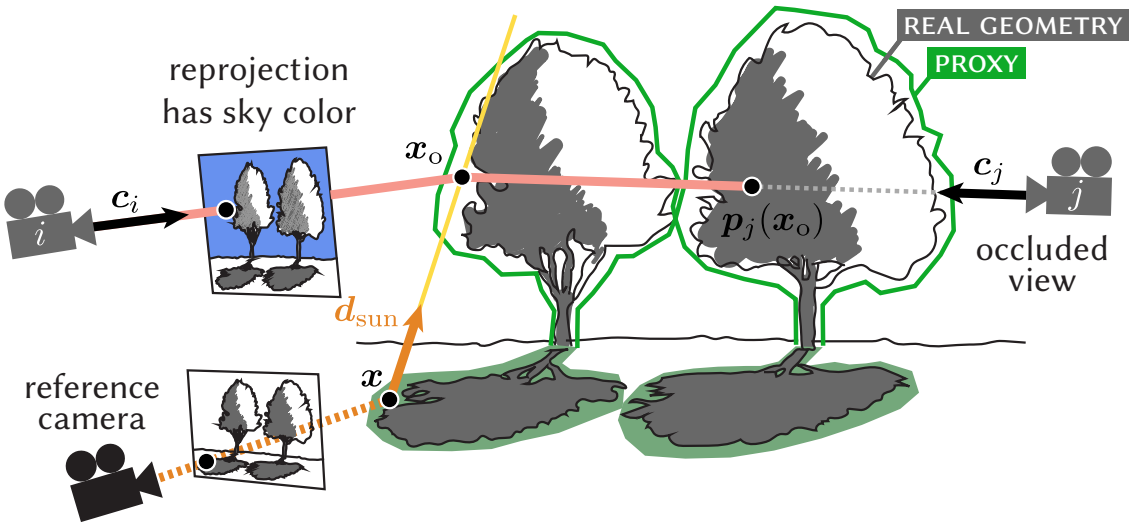


Figure 4.5: Computation of the RGB shadow images. For a visible point $\boldsymbol{x}$ we reproject the shadow caster point $\boldsymbol{x}_o$ on the proxy into the input images. In this example, image i contributes a (blue) sky color, indicating the proxy is inaccurate at $\boldsymbol{x}_o$. The contribution of image j is reduced thanks to our weighting term because $\boldsymbol{x}_o$ is occluded by the rightmost tree from the point of view of camera j.

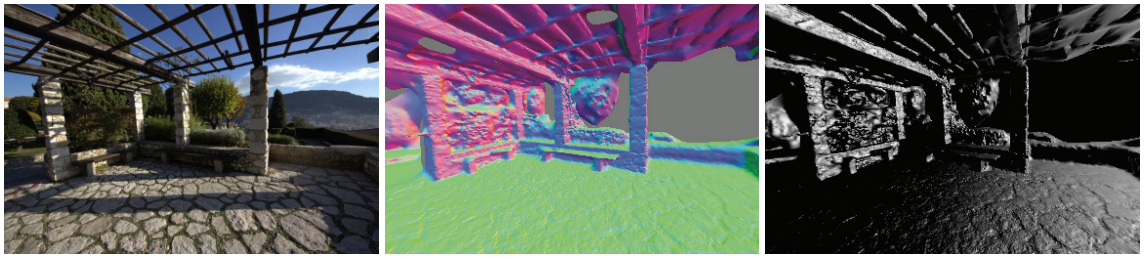
It encodes a preference for views that are closer to the sun direction, in a similar spirit to blending weights for IBR [19].

In addition to the weighted average of reprojected colors, we also maintain two additional pieces of information. First, we store the ratio of the distance from the visible point $\boldsymbol{x}$ to $\boldsymbol{x}_o$ to the distance from the current camera to $\boldsymbol{x}$; this provides a hint to the network on how soft the shadows should be. Second, we store the uncertainty of reconstruction, provided by the MVS algorithm since geometry is more likely to be erroneous whenever the algorithm's confidence is low. Our RGB shadow images can be computed quickly at test time for a captured scene.

4.3.3 Image-space geometric information via illumination buffers

The network takes as input a *source* and *target* lighting condition, that are defined by the respective sun positions. To help the network perform the lighting transformation, the first illumination buffers we provide are sun elevation for both source and target as scalars proportional to the angle between the horizon and the sun direction, as well as the sun directions in camera space as unit vectors. We also help the network determine surface

lighting depending on sun orientation, by using the proxy to compute normal maps in camera space. Finally, we provide a reflection buffer that is the dot product between the direction from the camera to the surface and the mirror reflection of the incoming sun ray at the surface. These help the network synthesize illumination consistently and also affects shadow removal. Our illumination buffers are illustrated below, and their impact on final relighting quality is evaluated in Section 4.5.5.



input image | normal maps | reflection features

4.3.4 Training the model

The three sub-modules of our network are trained jointly in a supervised manner to minimize the sum of three losses:

$$\mathcal{L} = \mathcal{L}_{\text{relight}} + \mathcal{L}_{\text{src}} + \mathcal{L}_{\text{tgt}}. \tag{4.2}$$

These loss functions compare the accuracy of our network's predictions (the final relit image as well as both intermediate refined shadow masks) to synthetic ground truth, which we detail in Section 4.4.

To refine the source shadow mask, a straighforward L_1 loss proved sufficient. Intuitively, this task is less ambiguous than refining the target shadow maps because the input image contains the source shadows:

$$\mathcal{L}_{\text{src}} = \mathbb{E}\Big[\big|r_{\text{src}}(S^{\text{rgb}}_{\text{src}}, I) - S^{\star}_{\text{src}}\big|\Big], \tag{4.3}$$

where r_{src} is the source refinement network, $S^{\star}_{\text{src}}$ is the ground truth shadow mask, I is the input image, and $S^{\text{rgb}}_{\text{src}}$ is the source RGB shadow image. The operator $\mathbb{E}$ denotes expectations taken over the training set.

For the target shadow refinement however, the network has less information to exploit from the input image, so we use a more complex perceptual loss:

$$\mathcal{L}_{\text{tgt}} = \mathbb{E}\Big[w_1 \cdot \mathcal{P}(r_{\text{tgt}}(S^{\text{rgb}}_{\text{tgt}}, I), S^{\star}_{\text{tgt}})\Big], \tag{4.4}$$

with r_{tgt} the target refinement network, $S^{\text{rgb}}_{\text{tgt}}$ the target RGB shadow image, $S^{\star}_{\text{tgt}}$ the ground truth target shadow mask, and w_1 a weight map.

$\mathcal{P}$ is a perceptual loss function. It extracts features from the two images independently using a pretrained VGG19 network and compares them with an L_1 loss. We use the implementation of Chen & Koltun [25].

In practice, pixels shadowed by geometry that is not reconstructed in the proxy are fundamentally ambiguous. They tend to bias the target refinement network towards conservative outputs (see Section 4.4.3). We reduce the contribution of these pixels using a binary mask computed from the ground truth shadow mask. We set $w_1 = \frac{1}{10}$ for the masked pixels, and 1 otherwise (value found empirically to give satisfactory results).

For the relighting network, we also use a weighted perceptual loss. We weight the loss using the difference between the ground truth and proxy shadow image so that we do not penalize parts of the shadow mask where the refinement step failed. Specifically, the weight is given by $w_2 = 1 - 0.9|r_{\text{tgt}}(S^{\text{rgb}}_{\text{tgt}}, I) - S^{\star}_{\text{tgt}}|$.

The overall goal of the relighting network is to produce a relit image I^{R}, which we encourage to match the ground truth target lighting condition $I^{\star}$ using, again, an image-space perceptual criterion:

$$\mathcal{L}_{\text{relight}} = \mathbb{E}\Big[w_2 \cdot \mathcal{P}(I^{\text{R}}, I^{\star})\Big]. \tag{4.5}$$

Note that I^{R} depends on the input image, the illumination buffers and the *refined* source and target shadow masks.

4.3.4.1 Details

The weights of all the convolutional layers are initialized according to He et al.'s recommendation [65] and the biases to 0. We optimize the network parameters using the

Figure 4.6: A sample viewpoint from each of our 10 ground truth training scenes.

ADAM solver [99], we train with a batch size of 4, and a learning rate of 2×10^{-4}. The remaining parameters of the ADAM optimizer are kept to the values recommended by the authors. Our model is implemented in Tensorflow [1]. Unless specified otherwise, the models were trained on a NVIDIA GTX 1080 Ti GPU until the loss stopped improving (typically 3–4 days). The architecture is a 64-channel ResNet for the relighting module and a 16-channel ResNet for the shadow refiners, following [211].

4.4 Synthesizing training data

Capturing a large-scale dataset of real photographs to train our multi-view relighting network would be cumbersome and fraught with practical difficulties. To guarantee sufficient coverage of the lighting scenarios, such a campaign would have to cover many different locations, maintain strictly fixed viewpoints during capture, and require day-long (or even month-long) capture sessions with many cameras. Even if this approach were practically possible, it would lack in diversity, *e.g.*, for the kind of lighting conditions and scene content available. In addition, data for shadow refinement supervision cannot be directly captured.

To bypass these issues, we use synthetic training data and render photo-realistic images using the Mitsuba [87] pathtracer. We gathered a set of 10 synthetic scenes from which we compute the data required for training. This approach allows us to generate arbitrary lighting conditions easily and have full control over the supervision at training time. To maximize diversity while keeping rendering time under control, we factorize the lighting computation by separately rendering the sun and sky contributions and compositing

the two at training time. The use of synthetic data also allows us to render ground truth shadow masks $S^{\star}_{\text{tgt}}, S^{\star}_{\text{src}}$, which is critical to our shadow refinement sub-network (Section 4.3.2).

The key requirement for our training images is that they closely resemble real photographs. That is, the scenes must contain highly detailed models of outdoors scenes, with realistic materials. For this reason, we chose to use professionally built models (either purchased or freely available) and develop a set of data augmentation techniques. Our experiments show that, even though our network is trained entirely on this synthetic dataset, it generalizes well to real images (Fig. 4.17, 4.18).

4.4.1 Synthetic scenes

We gathered 10 different outdoor 3D scenes to generate our training data; a sample viewpoint from each scene is shown in Fig. 4.6. The first 8 scenes were professionally modeled scenes we purchased[1]. We also used the large scene published by NVIDIA[2] and created two separate subscenes (a street and a square). The scenes are in standard industry formats (typically Autodesk 3DSMax), and include hand-crafted materials with complex shading trees which we export as Mitsuba scene description files [87].

We render the scenes using path tracing and Mitsuba's physically-based sun model, with HDR sky environment maps from Stumpfel et al. [173]. We remove the sun from these environment maps by mirroring the envmap. The physically-based model provides correct colors for the sun at different sun elevations, as well as a sky environment map [79]. We apply the average color and intensity shift of the sky for a given sun position during compositing (Section 4.4.2).

4.4.2 Photo-realistic rendering, layer decomposition and compositing-based data augmentation

Path-tracing complex outdoors scenes with a physically-based sun/sky model is expensive: rendering a converged image at 1024×768 takes about 10 minutes on our 400-core cluster. However, we noticed that our method works well with relatively noisy images,

[1] Scenes purchased from http://evermotion.org, taken for collections Archexteriors vol. 17 and 22.
[2] https://developer.nvidia.com/orca/amazon-lumberyard-bistro

so we use 64 samples per pixel for all our renderings, with good results. This corresponds to a recent observation that learning with noisy rendering data can be robust [115]. Generating the same resolution image with these settings takes about 10 seconds on the same cluster. For our dataset which contains about 17,000 rendered images this approach reduces rendering time from about 100 days to only 2 days.

For each training scene we select around 30 different viewpoints to obtain as much content variety as possible. To increase the number of lighting conditions within a fixed rendering budget, we render sky and sun illumination as two separate images that we composite on-the-fly at training time. This allows us to apply random intensity variations before we generate the final image. For each scene and each viewpoint, we render with 49 sun positions, and 5 sky conditions, while varying the cloud coverage. We store these render buffers as floating point linear images. Thus, for each viewpoint we have $5 * 49 = 245$ lighting conditions, before applying any intensity data augmentation. This leads to $245^2 = 60$K pairs of training lighting conditions for each per viewpoint.

On-line compositing. For a given training step, we need to generate a source "input" image, corresponding to the input photo we will use at test time and a target ground truth image, corresponding to the desired image relit with the target lighting configuration.

We start by randomly selecting 2 out of the 49 sun position images to be the source and target conditions and we randomly select a single sky condition image used for both, scaling the sky with the corresponding average color shift computed using the sky model [79]. Sky and sun illumination are highly correlated so this is not strictly physically accurate, but the quality of the results was satisfactory despite this approximation. We also randomly scale the sun intensity separately per channel, and randomly scale all channels of sky intensity.

Data-augmentation. We first randomly scale our images and select a random crop of 256×256 pixels. Real-world images have a variety of exposure and white balance settings. To be robust to this variety in the input, we apply random variations to both source and target images during training.

We next take the source and target linear images with all the random perturbations applied, and perform gamma correction, with small random variation on the gamma

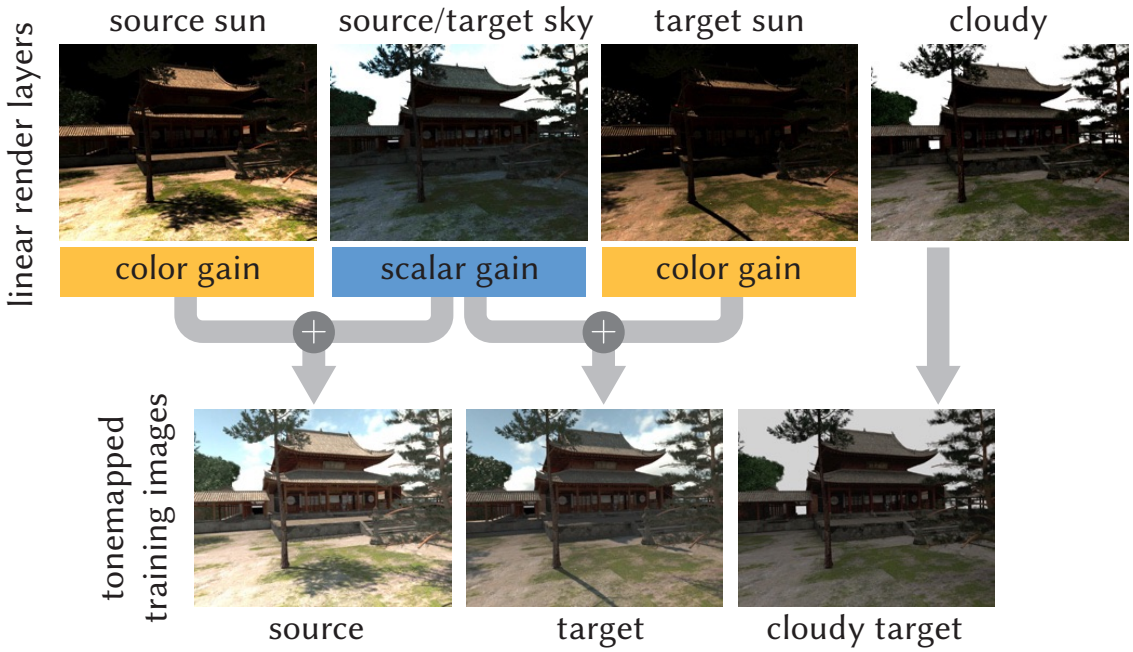


Figure 4.7: **Top**: linear images from left to right, source sun image, sky image, target sun image, cloudy image. **Bottom**: composited source, composited target and cloudy images, after data augmentation and tone-mapping. All linear images overexposed for visibility.

value. We use the same random variables for all transformations, including compositing, for source and target to preserve coherence; these are also applied to the RGB shadow images. We illustrate this process in Fig. 4.7; details of all the processing steps for compositing and augmentation are provided in appendix A. Our data augmentation scheme is critical to the performance of our algorithm, as we show in an ablation test (Section 4.5.5).

In addition to the relit image, we train our network to produce a "cloudy" layer, *i.e.*, an image lit only by a uniform mid-gray sky. We use it to approximate different degrees of overcast conditions.

4.4.3 MVS reconstruction of synthetic ground truth scenes

When relighting a real scene, we only have the approximate proxy representation of the scene to generate shadow images. For shadow refinement to be successful at test time, the network needs to learn the mapping between approximate proxy shadows and

(a) three of the renderings used for MVS reconstruction of a training scene

(b) ground truth geometry (c) reconstructed proxy

Figure 4.8: To bridge the gap between our synthetic training data and real multi-view datasets, we purposely degrade the quality of the training geometry by running a multi-view stereo algorithms on our renderings (a). Compared to the ground truth geometry (b), the proxy (c) is inaccurate and misses many details (*e.g.*, the trees).

ground truth shadows at training time. To achieve this we create a second representation of each synthetic scene by rendering a set of views, subsequently used as if they were photographs of a real scene. These photos are then processed with SfM and MVS to create a proxy of the synthetic scene, with the typical reconstruction artifacts of this process (Fig. 4.8). For more details on this step, please see appendix A.

4.4.4 Training with synthetic data

We train our network using both representations of each synthetic scene. The ground truth geometry and materials are used to render the sun and sky layers, and to create the ground truth greyscale shadow masks. The proxy is used to generate the illumination buffers and RGB shadow images.

RGB shadow images for training. RGB shadow images depend both on the source lighting condition, as we need to sample pixels from the source images, and on the target lighting condition as shadows are cast from the target sun direction. To generate the complete set of RGBD shadow images for training, we would have to render 49×(5+49) images for each viewpoint. Our RGBD shadow images depend on the target sun position for the shape of the shadows, and also on the source images for the color reprojection. These source images are composited sun and sky renderings. Given the number of viewpoints in each scene, the cost of rendering and storing these images would be too high. Instead, we only compute the colors for 5 source sun positions, and thus compute 49×(5+5) images for each viewpoint. During training, we use the closest source sun position rendering stored for the sun layer in the compositing. We can then apply the same image processing transformations on the fly as for the source and target image to the color values in the RGBD shadow images using the same random variables for consistency.

4.5 Implementation, Results and Experiments

We have implemented our method in both interactive and batch processing contexts. To perform relighting, we require a set of calibrated cameras and a proxy. The user must then specify the source sun position by clicking on a shadow caster and the corresponding shadow on the textured mesh (see supplemental video). We present quantitative and qualitative results, comparisons to previous work, ablation studies and applications. Our results and video can be found at http://fungraph.inria.fr/deep-relighting.html.

4.5.1 Qualitative results

We show the output of our relighting method for a variety of scenes, under a large range of lighting conditions in Fig. 4.18. Our method successfully removes and resynthesizes shadows, and achieves convincing changes in illumination levels for different times of day and lighting conditions. We present an extensive set of relighting results for the 8 different scenes in Fig. 4.18 with large sun arc movements in supplemental material. These include 2 drone video captures (first two rows of Fig. 4.18), 2 scenes from Duchene et al. [39] (last two rows of Fig. 4.18) and 4 scenes we captured ourselves.

(a) input (b) ResNet output (c) our output

Figure 4.9: Relighting is a challenging task for standard image-to-image networks. Even when provided with our auxiliary inputs and shadow masks, a ResNet model (b) struggles to remove cast shadows in the input (a), and cannot generate globally-consistent color changes and plausible novel shadows. Our model gives much more realistic results (c).

4.5.2 Comparison to a neural network baseline

Assuming proper training data is available, a natural approach to relighting outdoor scenes would be to train a standard model such as a ResNet [66, 90] to transform an input image given a target sun direction. We trained such a model with the image and source/target sun direction layers as input, using the same data augmentation (sky/sun rendering, exposure and white balance) as our approach. As shown in Fig. 4.9, the images it produces are not satisfactory. The network completely ignores the sun direction input layers and produces an image with reduced intensity and shadows that are only partially removed. It also does not synthesize cast shadows that are consistent with the target sun direction.

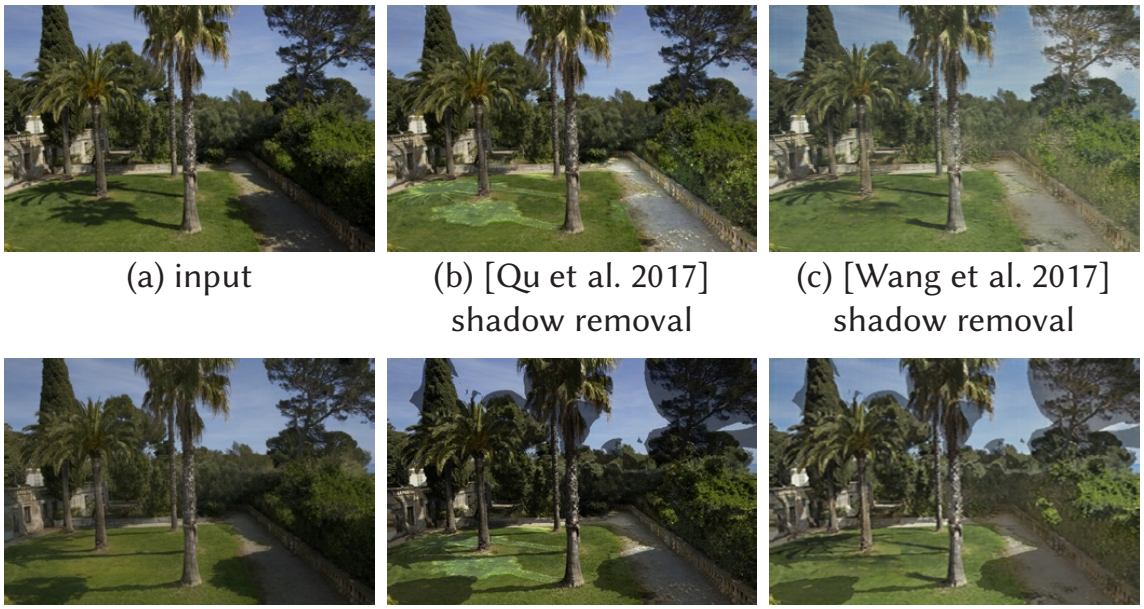
This purely image-based baseline simply does not have enough information to solve the severely ill-posed relighting problem. For instance, even with its large receptive field, the ResNet cannot properly deal with the non-local nature of cast shadows. Furthermore, this baseline has no notion of surface appearance or surface orientation.

4.5.3 Comparison to previous work

As a second comparison we first apply a shadow removal algorithm ([149] or [188]), then cast a new shadow using the proxy geometry. Fig. 4.10 shows the shadow removal generally fails on our real test images. Also, the proxy is often too approximate to use its cast shadows directly, justifying our shadow refinement approach. It is important

to note however that unlike our method, neither of these shadow-removal techniques uses multi-view information, and thus have much less information to work with than our approach.

We also compare to the relighting algorithm of Duchêne et al. [39], where we have used the same 3D reconstruction as in their original method. We see that Duchêne et al. achieve good quality shadow casting close to the original sun direction, but the shadow shape is completely incorrect when moving further away. The method also suffers from residual artifacts due to incorrect shadow classification (examples highlighted by red squares). These artifacts can be better seen in the companion video.



(a) input | (b) [Qu et al. 2017] shadow removal | (c) [Wang et al. 2017] shadow removal

(d) our output | (e) [Qu et al. 2017] + proxy cast shadows | (f) [Wang et al. 2017] + proxy cast shadows

Figure 4.10: We compare our method to a baseline that removes shadows using off-the-shelf algorithms then casts new shadows from our proxy geometry. (b) and (c) are the output of the shadow-removal algorithms from input (a). (e) and (f) show the same images with new shadows generated using the proxy. Our output is significantly cleaner (d).

(a) input (b) [Duchêne et al. 2015] (c) our output

Figure 4.11: Duchêne et al. [39] often leaves shadow residuals (b), bottom. Their method also breaks when the desired relighting is far from the input (b), top. Our method is more robust and can synthesize significant lighting changes (c).

(a) input (b) ground truth (c) our output

Figure 4.12: We evaluate our model on a held-out synthetic scene (a) for which we can generate arbitrary novel lighting conditions (b). Our model can faithfully predict the novel illumination (c) even though it has not been trained on this scene and has only access to the degraded geometry (proxy) and input images.

4.5.4 Comparison to synthetic and real ground truth

We next show comparisons to a synthetic scene held out from the training data in Fig. 4.12. Note that we used a reconstructed proxy and not the perfect, ground truth geometry to obtain these results.

We also show a qualitative ground truth comparison with a real scene, in Fig. 4.17. For this, we photographed the same scene, at different times of day, with the same viewpoint.

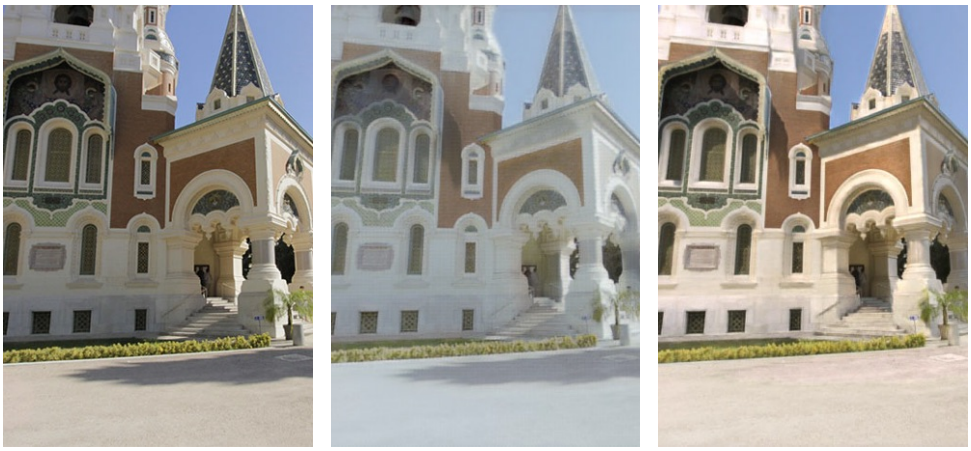
4.5.5 Model ablations

We performed several ablations of our model. For each analysis we trained the different configurations for 100 epochs. We held out one synthetic scene for testing and trained our network on the others.

Table 4.1 summarizes the numerical error for the different ablations

We present interactive side-by-side comparisons of the different ablations as a web page in supplemental materials for three different scenes.

Data augmentation. Our data augmentation procedure randomizes exposure, white balance and gamma correction. It is critical to the success of the network, especially in generating correct illumination levels. In Fig. 4.13, we show an example output of our model trained without data augmentation.



input no data augmentation our output

Figure 4.13

Illumination features. When we remove the illumination features, the network has difficulty finding the correct illumination levels, and generates inconsistent results. These layers help the network alter the image intensity consistently, improving shadow removal. This is visible Fig. 4.14.

input no illumination buffers our output

Figure 4.14

Shadow refinement. If we only remove shadow refinement from our solution, shadow removal is also worse, and shadow re-synthesis exhibits ghosting artifacts as visible in

Fig. 4.15.

input no shadow refinement our output

Figure 4.15

RGB shadow images. If we deactivate our RGB shadow images and use standard gray-scale shadow masks instead, the network cannot overcome over-reconstruction artifacts and the resynthesized shadows mostly follow the masks as we can see if Fig. 4.16.

input no RGB shadow images our output

Figure 4.16

model	L_1 error	L_2 error
our full model	**0.131**	**1.98e−4**
no shadow refinement	0.179	2.43e−4
no RGB shadow images	0.184	2.72e−4
no illumination buffers	0.200	2.51e−4
no image augmentation	0.445	4.54e−4

Table 4.1: We evaluate the error of our relighting numerically on a held-out ground truth synthetic scene. We report the average L_1 and L_2 pixel error. The input illumination buffers, shadow refinement subnetworks and data augmentation procedure all contribute to the final quality of our result.

4.5.6 Applications

Our method can be used in several different contexts. We present four potential applications: interactive relighting, drone-video relighting, relighting for image-based rendering and relighting of reconstructed textured meshes.

(a) input photo (b) same scene, 3h30 later (real photo) (c) our output

Figure 4.17: Our network generalizes to real input images (a) and produces photorealistic outputs (c) that closely match real, novel lighting conditions (b).

4.5.6.1 Interactive Relighting

In our interactive application the user selects the input image to relight, and the network produces the relit image together with the "cloudy" layer. We can simulate varying levels of overcast conditions by inserting a blurring kernel after shadow refinement and before relighting, providing a user-controlled "cloudiness" parameter. We then blend between the resulting relit and cloudy image to produce the output (Fig. 4.19).

We have developed an integrated interactive viewer by calling tensorflow with a CUDA - OpenGL coupling, allowing interactive performance.

Performance numbers: 5-8 frames per-seconds on an Dell Precision 7810 computer with an NVIDIA 1080GTX GPU at 1080p resolution (see video).

Performance breakdown Our pipeline takes less than 10 minutes from the beginning of the multi-view calibration procedure to the final relit result. In particular our neural network runs at interactive rates, which enables a user to alter the lighting dynamically. We report the computational cost for a typical scene with 109 input photos in the table below.

preprocess	camera calibration	1 min
	proxy reconstruction	6 min
	manual sun position input	15 s
runtime	rendering RGB shadow images	40 ms
	rendering other buffers	<10 ms
	network inference	70 ms

4.5.6.2 Drone video relighting

We extract frames from a drone video and perform standard multi-view reconstruction. We can then individually relight the frames of the video using our approach, either at a single different time or dynamically changing the lighting during the video (Fig. 4.1, top row). This is best seen in the supplemental video. Our algorithm treats each frame independently, without explicit temporal regularization, so we sometimes observe flickering in the rendered videos. This is easily corrected using a post-processing temporal smoothing method like that of Lai et al. [106].

4.5.6.3 Relighting for Image-Based Rendering (IBR)

We have integrated our relighting in an interactive IBR system implementing [19], by relighting the blended novel view on-the-fly as if it was an input photo. The ability to relight for IBR overcomes one of the major limitations of these techniques that are

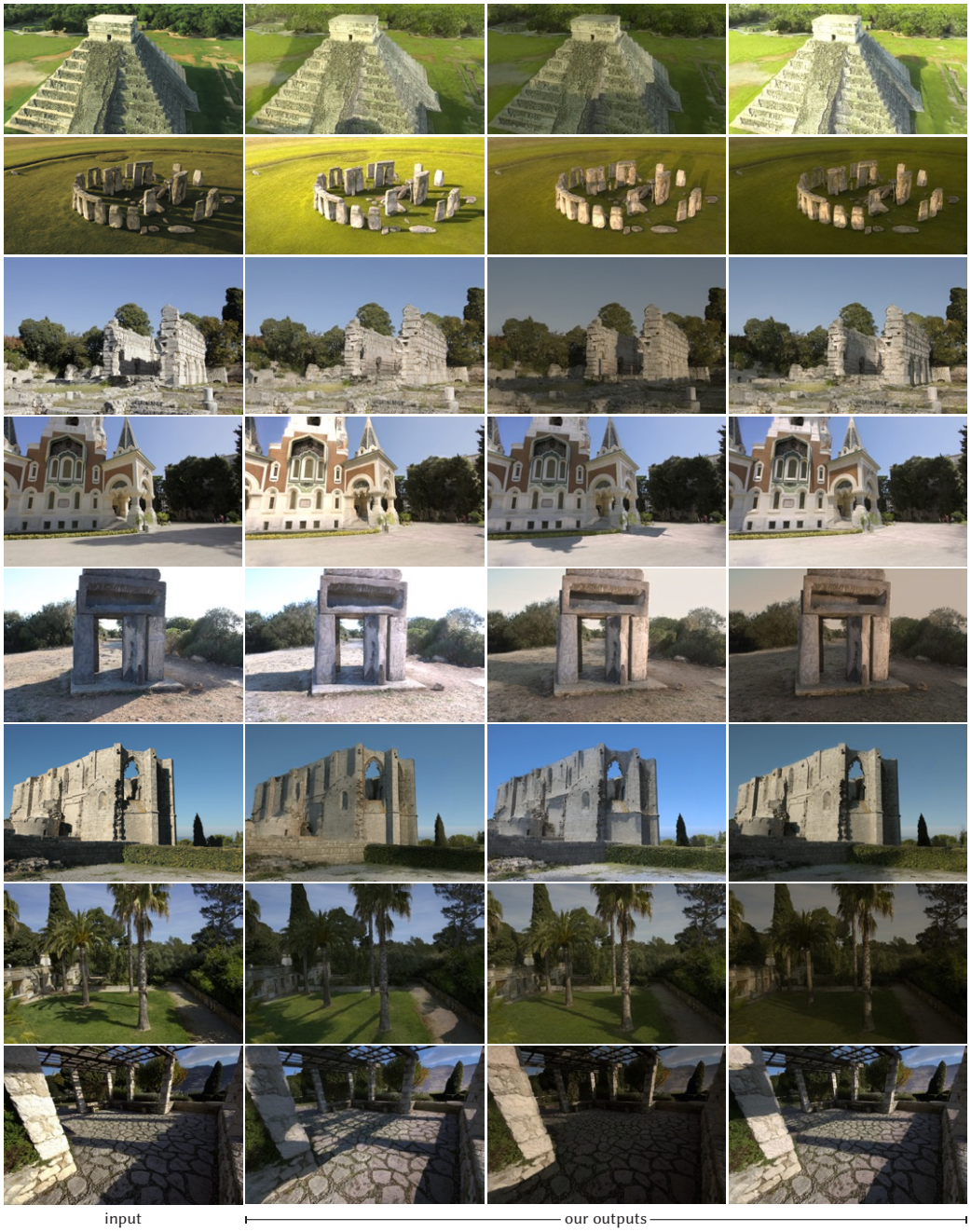


Figure 4.18: Results using our network. The leftmost column is the input, followed by three outputs corresponding to different sun positions. First and second row respectively generated using the Chichen Itza drone video (copyright Drones Yucatán youtu.be/qkveKd3nW9w) and Stonhenge drone video (copyright Namyeska youtu.be/JHeDP7_YBos) both used with permission.

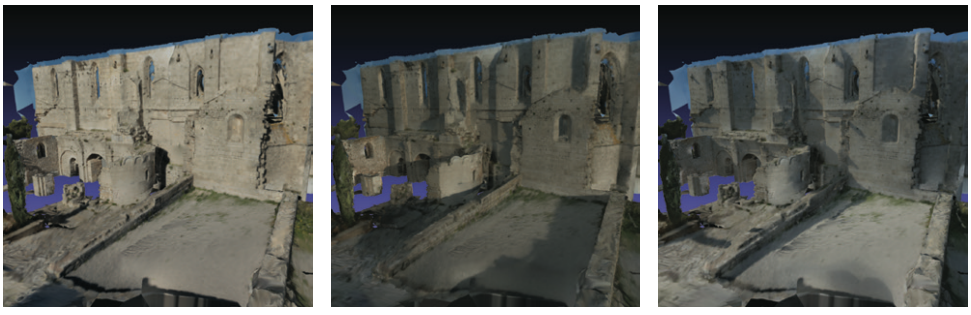
input 50% cloudiness 100% cloudiness

Figure 4.19: Our model exposes a user-controllable "cloudiness" parameter to modulate between *sunny* and *overcast* conditions.

otherwise restricted to the lighting conditions of capture. Please see the video for examples.

4.5.6.4 Relighting for Reconstructed Textured Meshes

We can relight all the images for a given multi-view dataset in a new sun position, and then re-run the final texturing step after geometric reconstruction. In supplemental, we provide three meshes with different versions of the same scene, *i.e.*, two conditions in addition to the original captured lighting (Fig. 4.20).



(a) original mesh (b) relit mesh 1 (c) relit mesh 2

Figure 4.20: Our algorithm can also be used to relight an input textured mesh (a) to different lighting conditions (b), (c).

4.6 Relighting for captured Scene composition

While capturing scenes for such IBR methods is as easy as taking a few tens of photographs, there is no obvious way to change the geometry, and in particular to realistically *combine* content from different captures into a single richer virtual environment. In Chapter 3 we showed how objects can be removed from scenes and Thonat et al. [184] also present a way to move objects after inpainting, but only in a single scene. Compositing objects coming from different scenes poses a bigger challenge since this would require the lighting conditions of the different scenes to be coherent to have a realistic composition. The interaction of shadows between the scenes must be handled and the lighting directions must be "aligned". Compositing scene is thus inherently related to relighting. Using the relighting method we presented, we introduce the first method that allows combination of captured scenes with coherent lighting, suitable for IBR, multi-view texturing but also image manipulation. This extension was developed as part of Baptiste Nicolet's internship which the author of this thesis co-supervised and led to a separate publication [138]. This contributes to the main goal of this thesis by giving additional artistic control to IBR.

Naively cutting pieces from one scene to another would result in three important issues. First, lighting is inconsistent between different scenes both in terms of direction and intensities, since the illumination conditions are not the same in each capture. Second, regions of contact between pieces of one scene often suffer from an unrealistic look of "floating" when directly inserted into another scene. Finally, inconsistencies in camera parameters (*e.g.*, color temperature) may occur when combining captures and negatively affect the final result.

To address lighting inconsistency, we use our multi-view relighting method. However, since it was designed for a single scene, we need to adapt it to the context of scene composition. Specifically, when inserting part of potentially several *source* (or *part*) scenes into a *target* (or *reference*) scene, we need to align the lighting conditions of the source scenes to that of the target while synthesizing new realistic shadows in the composite scene. Correct handling of overlap between shadows in the source and target scenes is also a requirement. We develop a two-pass approach that enables good shadow removal and consistent shadow and lighting in the composite scene, by adapting the network so it can be used without retraining. We also provide solutions to increase the

realism of contact geometry by creating more realistic contact shadows using Ambient Occlusion and allow user control of color temperature inconsistencies.

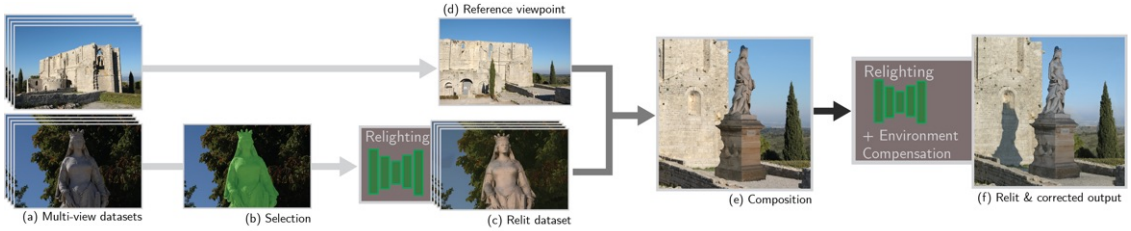


Figure 4.21: Overview of this extension. We use multi-view datasets (a) as inputs. One dataset is considered the "reference" dataset (d), and we use its sun direction to relight the other datasets (c), then we compute a composition of the selected parts (b) into the reference scene (e). Finally, we relight the composite scene as one to synthesize shadows across scenes and we account for the changes in the environment of each scene (f).

4.6.1 Overview

This application can be decomposed into three main components :

1. An interactive application that allows the user to import, select, and move parts of captured scenes to create the desired composition.
2. An adaptation of our relighting solution to enforce consistent lighting and shadows in the composite scene.
3. An environment compensation step to account for the modification of the surroundings of each part.

The overview of the scene composition extension is displayed in Fig. 4.21.

4.6.2 Composition Interface

The first building block of the approach is the composition interface. During this stage, we render a preview of the composition with a slight variation of the unstructured lumigraph algorithm [20], that allows the user to interactively edit their selection and move parts around until the composition is finalized.

We refer to the scene in which the user imports objects as *the reference scene*, and we will refer to the objects imported in the reference scene as *parts*. We refer to all scenes together as the *constituent* scenes, and the final combined scene as the *composite scene*. Finally the scenes used to extract parts are referred to as *part* or *original* scenes.

4.6.3 Consistent Composite Lighting

The composite scene contains the geometry from the different parts and the reference scene. We next proceed with the relighting step, to produce consistent lighting in the composite scene. If applied naively (See section 4.6.5 below), our relighting method cannot handle the multiple constituent parts and their corresponding different lighting and shadow levels. In addition, it cannot handle the different shadow interactions between the geometries coming from separate scenes if used on each constituent scene separately. To avoid the artifacts from such a naive solution, most notably for shadow removal, we proceed in two passes.

1. An *offline* pass relighting the *entire* scene of origin of each *part* to match the lighting conditions of the *reference scene*. We do so by relighting all the input views.

2. A second pass where we relight the composition, allowing us to generate cast shadows between parts. This can be either online in the novel view, or offline on all the input views, allowing interactive IBR for free-viewpoint navigation.

In most of the examples given, we used the lighting conditions of the *reference* scene as target lighting conditions. Compositions are however not restricted to the lighting conditions of a constituent scene, and we can create compositions using any desired target sun direction (*e.g.*, Fig, 4.27).

4.6.4 Environment Compensation

Consistent lighting and shadows in the composite scene are often not enough to achieve a satisfactory level of realism. As each part is extracted from a given environment in its scene of origin, and inserted in a new one, residual visual artifacts may remain even

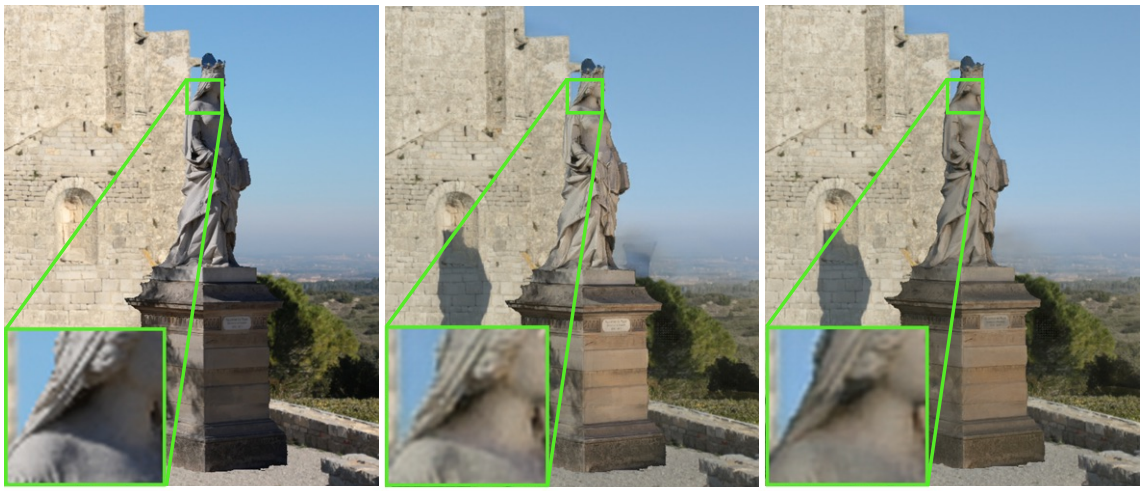
after relighting has been applied. We identified two factors that improve realism of compositions: *ambient occlusion* and *camera parameters*. We estimate the former in both the scene of origin and the reference scene for each part, and apply the corresponding compensation to the result of the second step. We provide the user with a per-scene color temperature slider, since we have no control over the camera parameters with which each scene is captured. The result of this compensation step is the final composition, suitable for IBR.

4.6.5 Naive solution

Direct compositing of different parts into the reference scene creates obvious visual artifacts due to the different lighting conditions in the constituent scenes (*e.g.*, Fig. 4.22(a)). Our first attempt was to apply the relighting network to the composite scene in a single pass. In our case, there are *multiple* source conditions, one for each part and one for the reference scene, while the target condition is common to all constituent scenes. To apply the method to the composite scene, we generate all *source* information (*i.e.*, shadow maps, illumination buffers) on a per-pixel basis, according to the source condition of each original scene; *i.e.*, source shadow map, sun direction, elevation etc. While the network was able to correctly predict the refined source shadow map in spite of the multiple sun orientations, it failed to completely remove shadows in some areas of the composition, as shown in Fig. 4.22.

Indeed, the network was never trained to deal with compositions of multiple scenes. One possible explanation for this failure is that the network cannot handle multiple *levels* of the shadows in the different scenes, since they can be significantly darker from one capture to another. Since the network is trained with single scenes, it may have learned to deal with a global value over the whole scene for shadows to be removed or added.

A direct solution to this issue would be to generate composite training data to re-train the network for multi-scene relighting. However, generating such data is very complex compared to the original, single-scene case. If we wanted to re-train the network, we would have to manually cut different pieces from the various training models, and create a large number of combinations of parts and references where each composition would require manual placement of pieces. This process would be even more complicated when inserting parts from several different scenes into the composite. As a result, we chose to



(a) Naive Composition (b) Direct Relighting (c) Deferred Relighting

Figure 4.22: Example of the failure to remove shadows while directly using the relighting network of [147] in the multi-scene setting. The original shadow in (a) is not completely removed when using the network directly (b). Our approach allows us to remove it more effectively (c).

develop a new approach that can use the original pre-trained CNN by using two separate passes, and some careful preprocessing to generate the correct illumination buffers.

4.6.6 Two-pass Relighting for Composite Scenes

Our two-pass solution consists of first relighting each scene individually, and then relighting the composition in a second pass. Our approach is designed to correctly remove shadows in the multi-scene setting – which cannot be handled directly by the relighting network – and to provide a consistently lit composite scene. We proceed as follows:

- Each scene is relit individually, *i.e.*, we relight all the input images of each scene to match the lighting conditions of the reference scene. This is done once, *offline*, and requires care to only consider the selected parts of each scene. This pass generates consistent lighting for each part, but we are lacking the interactions between parts and the reference scene.
- In the second pass, we relight the current viewpoint of the composition rendered

with the input images modified by the first pass. This adds cast shadows between parts, and creates fully consistent lighting and shadows.

First Pass. The goal of the first pass is to generate lighting and shadows on the selected part itself that are consistent with the reference lighting conditions (*i.e.*, with respect to its orientation in the target scene), and in particular to correctly remove shadows of the source lighting condition of each part. We do this by adapting the relighting network to relight each selected part. This pass is applied on the original part scene, but care must be taken to provide correct layers to the relighting CNN. We need to avoid unselected parts of the original scene from casting shadows onto the selected parts. We modify the shadow casting step to avoid this, see Fig. 4.23. Specifically, we send a shadow ray from each visible intersection point in the sun direction, to determine if the visible point is in shadow. We compute the source shadow image normally, but we compute the target shadow image only with the selected geometry, to avoid shadows cast by the non-selected geometry.

(a) Original Input Image **(b) Relit Image - no shadow filtering** **(c) Relit Image - with shadow filtering**

Figure 4.23: Illustration of our first pass relighting strategy. When relighting the input viewpoints (a) of a scene, we need to make sure that no shadow is cast by *unselected* geometry on the selected part (outlined in white in image a), resulting in hard to remove shadows (outlined in red in image b). We therefore intersect rays only against the selected geometry when computing the *target* shadow map. Since the network cannot hallucinate full shadows, we end up with a "shadow-free" relit image (c).

The shadow refinement part of the relighting network is thus provided with the input

that will produce the desired result. At the end of this pass, we have each input image of each part scene with shadows removed, and self shadows correctly cast from the selected geometry in the reference lighting conditions (Fig. 4.23).

Second Pass. We can now apply the relighting network a second time on the full composite scene to cast shadows between constituent scenes, and finalize the consistent overall lighting.

This pass also requires that we carefully prepare the data sent to the relighting network. Specifically, when computing the *source* shadow images, we ray cast again only considering visible selected geometry of each part. The target shadow image is computed using the full composite scene containing all the geometry.

This pass can either be done online at a given novel view, or as a preprocess on all input views (*i.e.*, reference and part input views), and then used for IBR (see Sec. 4.6.8).

4.6.7 Environment Compensation

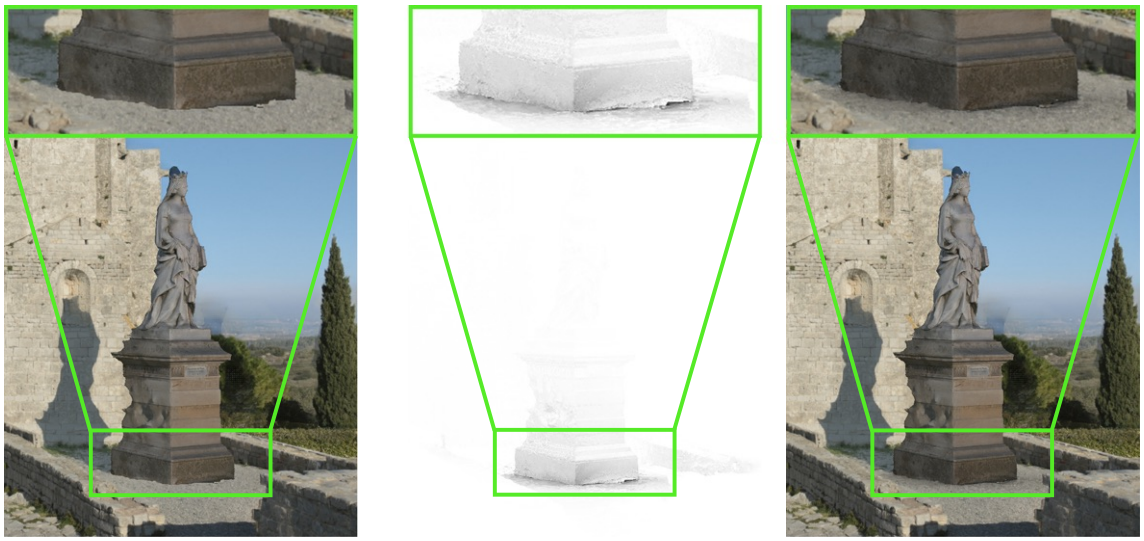
After our two pass relighting, the resulting composite has a greatly improved level of realism, for example Fig. 4.22 (c). However there are two remaining issues.

First, parts inserted into the reference scene often appear to "float" above the ground because we have not captured the mutual shadowing effect between the two scenes in the lighting. We compensate for this problem by computing an Ambient Occlusion (AO) shift based on the geometry of the two scenes, similar in spirit to the differential rendering of Yu et al. [204].

Second, the overall color temperature of the two scenes may be very inconsistent, and may not convey the desired visual effect. We allow the user to control the color balance of the composition to achieve the desired effect.

4.6.8 Implementation & Results

All our results are obtained on scenes reconstructed using standard Structure-from-Motion (SfM) and Multi-View Stereo (MVS) to create a geometric *proxy*. We used the commercial SfM/MVS package RealityCapture [151] to perform reconstructions.



(a) Without AO (b) AO Shift Map (c) With AO

Figure 4.24: Illustration of our AO shift computation. We compute a per-pixel ambient occlusion shift (b) by casting rays in the original scene of the visible part and in the composite scene. We apply the ratio of the computed values (b) to the composite image (a), resulting in image (c).

(a) Before Temperature Compensation (b) After Temperature Compensation

Figure 4.25: Illustration of our color balance compensation.

We enable interactive exploration of the resulting composition by applying our extended method on all viewpoints of each constituent scene. We then use the same per-pixel ULR as for the composition preview, reprojecting modified images. In order to prevent occlusion issues when relighting a viewpoint of a given scene occluded by another part,

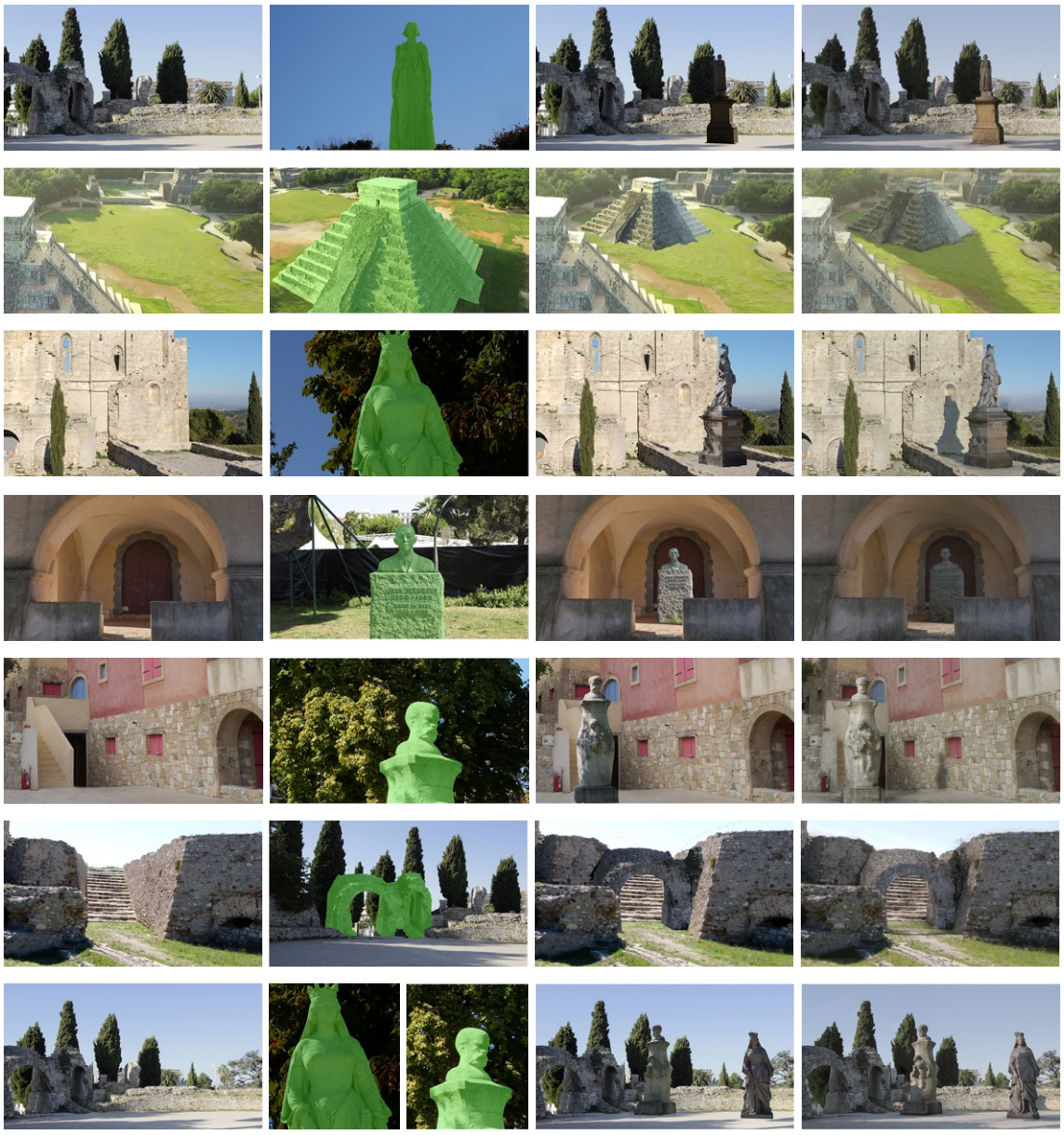


Figure 4.26: Examples of compositions created with our approach. For each row, the leftmost image corresponds to the reference scene for our composition, the next image shows the scene from which we extract a part, highlighted in green, the next image is the naive IBR composition of the scenes, and the rightmost image is the result of the composition using our approach.

Scene	Pass 1	# Images	Pass 2	# Images
2nd row	7m36s	177	33m8s	354
3rd row	3m1s	75	30m56s	247
4th row	3m24s	79	23m35s	194
6th row	4m42s	85	15m26s	126

Table 4.2: Computation time of some of our compositions. For each line, the first column indicates which composition we refer to (row of Fig. 4.26), the second is the duration of the *first pass*, relighting all input viewpoints of the imported scenes. The third column (# Images) is the number of images of each scene, and the fourth is the time of the *second pass*, which allows *interactive* free-viewpoint navigation in the composition after this computation. This step is longer than the first one due to our expensive computation of ambient occlusion via ray casting, and could be accelerated (*e.g.*, using ray-tracing hardware). The last column shows the number of input images (# Images) of the composite scene that are relit.

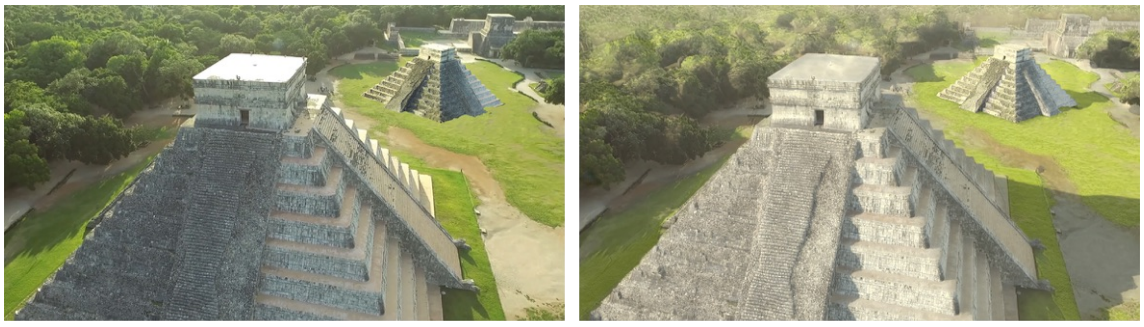
we adjust each camera's clipping planes to be as close as possible to the selection, thus removing most of these issues, and ensuring good quality renderings when the user viewpoint is near a part's input viewpoint.

We show statistics of our scenes and computation times on a Intel Xeon Silver 4110 with 32GB RAM and Quadro P5000 GPU in Tab. 4.2. These computation times can be explained by the need to cast visibility and shadow rays through each pixel of the scene in both passes, as well as AO sample rays in the second pass, and running the result through the network. While we already use accelerating data structures and leverage SIMD instructions on the CPU side, the ray-tracing overhead could be further accelerated using ray-tracing hardware.

The relighting network this extension can be applied in different context such as: multi-view image editing, IBR and textured meshes.

We show 7 examples of compositions in Fig. 4.26, including the case of mixing 3 different scenes, and the case of a different lighting direction from the reference scene (Fig. 4.27). As we can see, our compositions provide a high level of realism, providing a fast way to rapidly create more complex scenes.

Such multi-view editing can be directly used for IBR. We can either apply the second pass relighting for each novel view on the fly (taking approx. one second per frame) or apply



(a) Naive Composition (b) Arbitrary Output

Figure 4.27: We can produce compositions using any provided sun direction (b).

the pass on all the input images for all views of the reference and part scenes. When doing the latter, we can use our per-pixel ULR for interactive free-viewpoint navigation.

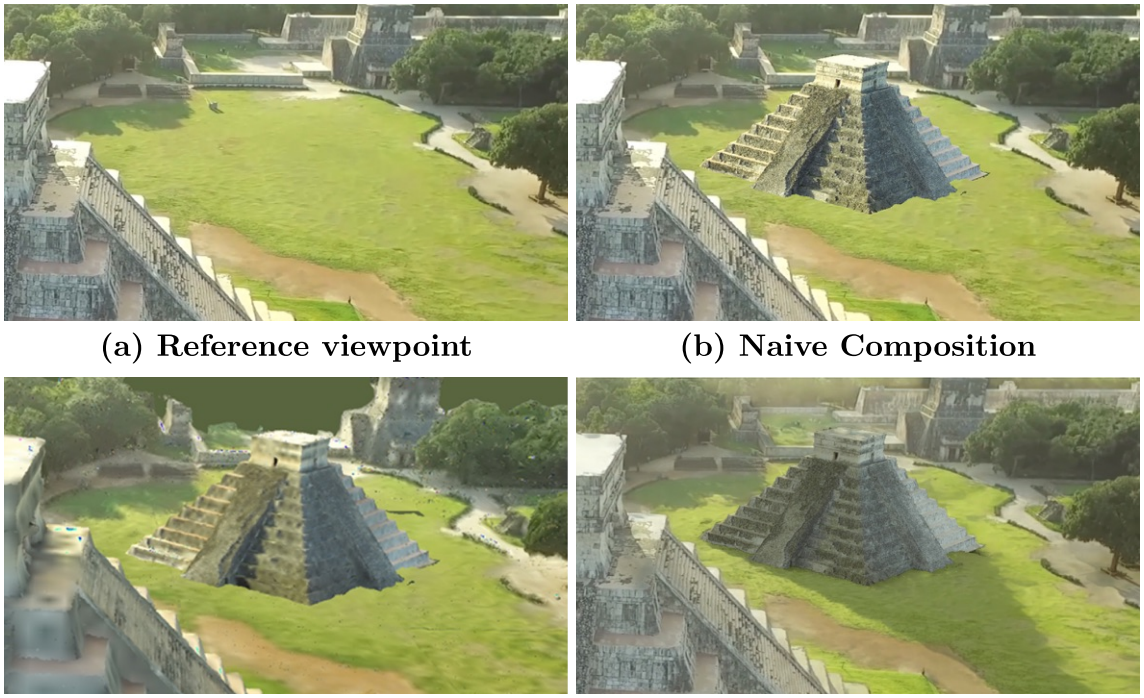
We also show composite meshes for this extension to our method. First, we apply it to all the input images, then re-texture the composite mesh with the relit images. In Fig. 4.28 we show two examples of composite textured meshes with coherent lighting.

Figure 4.28: Two examples of meshes textured using the relit input viewpoints.

4.6.9 Comparisons & Evaluation

We show comparisons with naive compositions in Fig. 4.26. We also show a comparison with the Deep Neural Textures approach [181], which is the only other case of composite

captured scenes, albeit only with pieces of the *same* scene. As we can see Fig. 4.29, Deep Neural Textures do not generate cast shadows for the duplicated pieces.

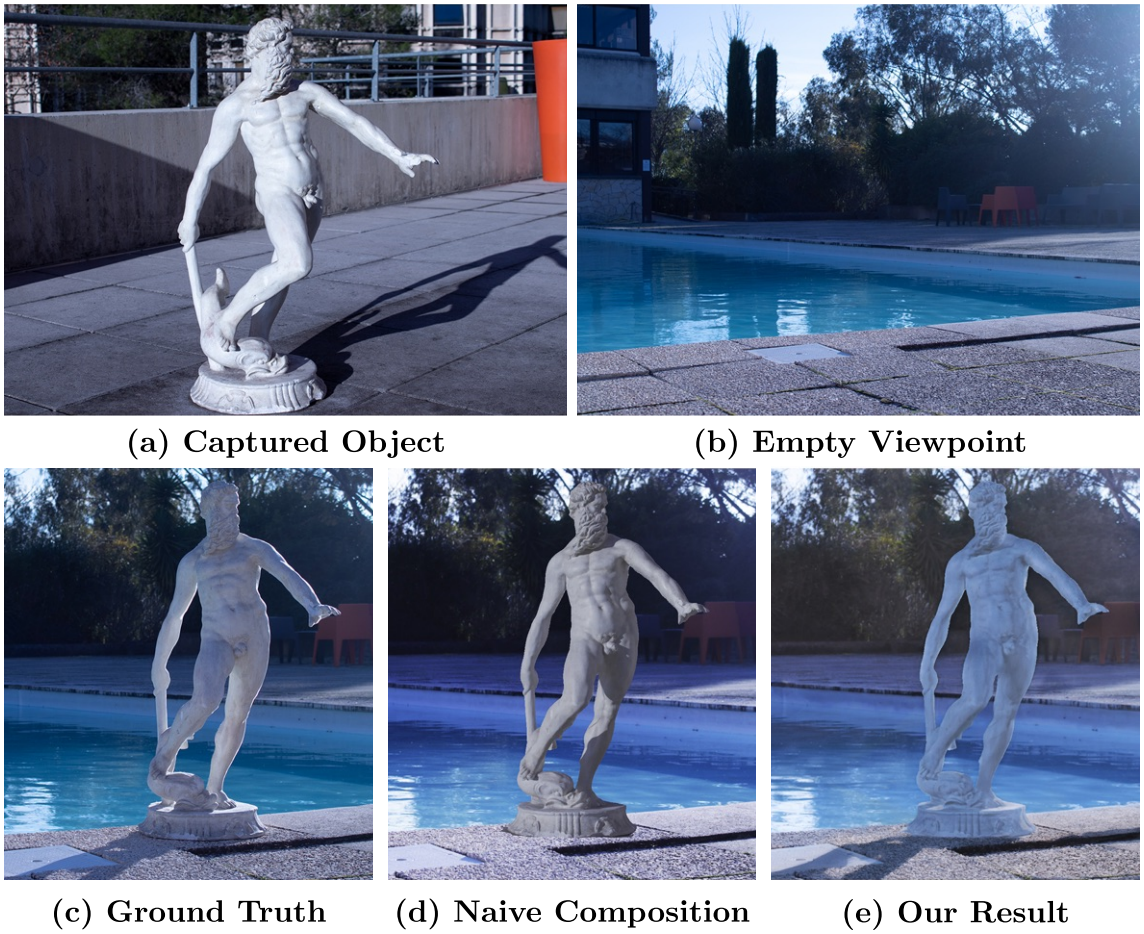


(a) Reference viewpoint **(b) Naive Composition**

(c) Thies et al. 2019 **(d) Our Method**

Figure 4.29: Comparison of scene composition with [181], in the case of object duplication.

Finally we provide a ground truth comparison by capturing a scene twice, once with an additional object and once without. We show our composite compared to the ground truth version in Fig. 4.30. While not perfect, our composition is quite close to the ground truth. Examples of remaining artifacts include small effects such as the highlight on the left arm of the statue, since the network is not designed to handle non-diffuse effects.



(a) Captured Object (b) Empty Viewpoint

(c) Ground Truth (d) Naive Composition (e) Our Result

Figure 4.30: Real-world ground truth comparison of a composition with a picture of a object inserted in a scene. While lacking some global effects, our approach conveys a convincing result compared to the naive solution (d) and the initial conditions of the statue (a).

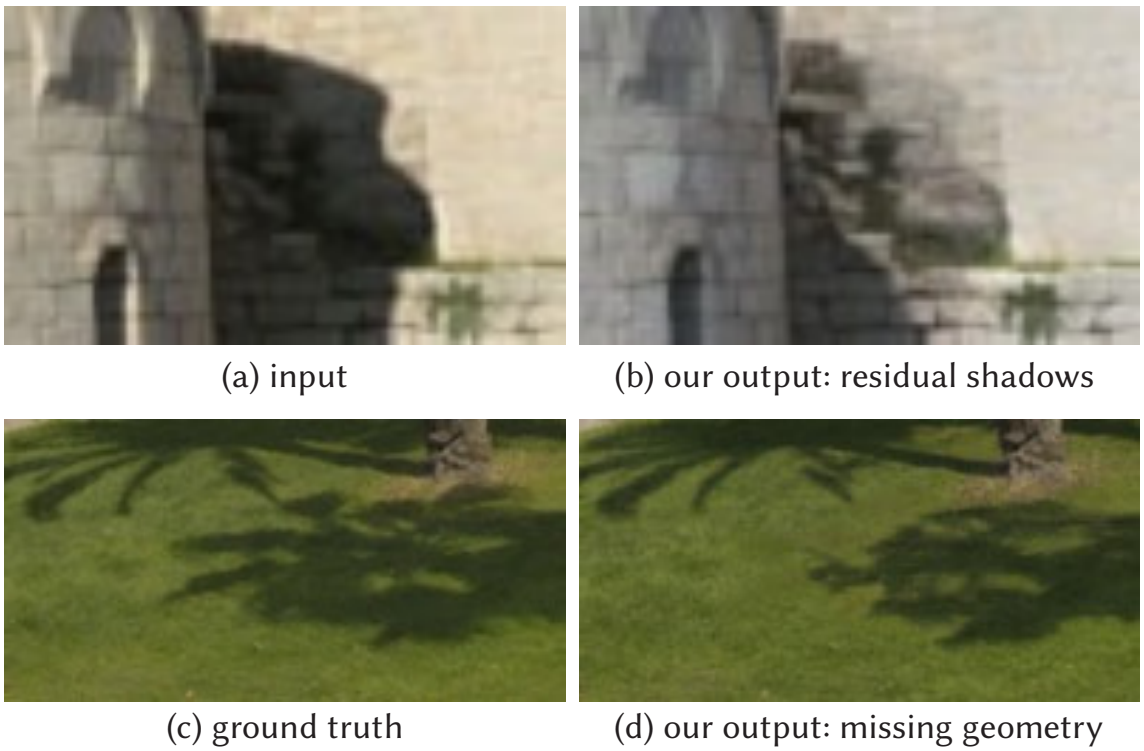
4.7 Limitations and Future Work

Limitations. Our relighting method generally produces plausible results for the scenes we tested, including scenes from previous work, drone captures, internet image datasets and our own captures. It also permits compositing of scenes with different lightings. Occasionally, slight shadow residue is visible in some views of a given dataset (see Fig. 5.10(a)); this typically occurs in overexposed or very dark image regions where shadow information is unreliable. This phenomenon may be even more visible when

applying the network twice in case of compositing. Our network may also occasionally produce small "checkerboard" artifacts, that come from the "deconvolution" upsampling layer. This is a common issue with this type of network.

Our goal is plausibility, and therefore in most cases the network does not hallucinate *additional* shadows when no occluder geometry is available. This can be seen in Fig. 5.10(b) where the top branches of the palm tree are missing from the relit shadow at the input sun position. However the result is plausible since the original shadow is cleanly removed. This limitation is shared with the compositing application where missing occluders do not cast shadows on inserted content. In addition, as the network does not explicitly handle global effects it can have visible impacts on compositions. It can be seen in the real-world ground-truth comparisons (Fig. 4.30), where while we achieve significant improvement (b) over naive compositing (d), our method fails to account for global effects such as the reflection of light over the water's surface, which illuminates the statue from behind.

Future Work. Our method is currently limited to outdoor scenes with a simple lighting model. Being able to treat interior scenes, where materials and indirect illumination are crucial, is one direction to generalize this work; we introduce a method going in this direction in Chapter 6. Another possible research direction is to allow more drastic changes such as day/night. One could also integrate our method with a generative adverserial network to give more faithful shadows especially when dealing with extremely bad geometry. From the image editing perspective, the requirement of a full reconstruction and the number of images required to get one limits the applicability of the method. It would be interesting to reduce the number of images needed and have the method work with other geometry representations such as depth maps, *e.g.*, allowing its direct use with RGBD images.



(a) input (b) our output: residual shadows

(c) ground truth (d) our output: missing geometry

Figure 4.31: Thanks to our RGB shadow images, our algorithm can generally refine inaccurate shadows. However it sometimes confuses texture detail with the input shadows (a), which creates a visible shadow residual (b). When a scene object is not properly reconstructed by MVS (shadow of the palm leaves in (c)), our model cannot hallucinate the missing shadows (d).

4.8 Conclusion

In this chapter, we presented a deep-learning based method, guided by approximate reconstructed geometry, that enables multi-view relighting. Important elements of our relighting solution are the shadow refinement subnetworks, guided by our newly introduced RGB shadow images, as well as illumination buffers. The use of synthetic data allows generation of highly diverse ground truth data, and the creation of a proxy representation in addition to ground truth geometry for each synthetic scene allows supervised training for shadow refinement. The main strengths of our approach and its robustness allowed us to extend the relighting work to a method to simply and quickly create visually compelling compositions of captured multi-view scenes, by adapting the relighting network to our task.

Our results show that by performing relighting of multi-view datasets we greatly increase their utility for traditional applications such as photogrammetry meshing and IBR hence providing a strong editing capability on top of IBR methods. We also demonstrate very powerful novel image and video manipulation applications for drone footage and photos of landmarks, where internet-based multi-view data is available showing that multi-view information can largely improve single image editing capabilities. Lighting is crucial for scene compositing; we also present the first method that can handle several scenes and provide coherent illumination in a resulting composite scene, allowing, through relighting, easy geometry modification of scenes without the need for asset creation.

In this and previous chapters, we presented powerful editing methods for IBR. One of their strengths is also a weakness regarding our main goal which is to have flexible image-based rendering methods. All the methods presented so far are a preprocessing step for input images. These input images are first edited before being used by an IBR algorithm that is agnostic to the fact that images where edited. That means that it increases the number of steps and interaction needed from users which decreases the usability and interest of the pipeline. In Chapter 6 we attempt to reunify IBR and relighting in a single algorithm that allows free-viewpoint relightable rendering. Before presenting this approach, we introduce in Chapter 5 a new machine learning approach for IBR, Deep Blending. More specifically we present our contribution to per-view geometry meshing. While this topic is somehow orthogonal to the main goal of the thesis it greatly contributes to the quality of the final rendering, leading to improvements in state of the art for IBR. Moreover, the experience gained during the Deep Blending project and the better understanding of IBR issues, greatly impacted the design choices of the method presented in Chapter 6.

Chapter 5

Per view meshes for Deep Blending Rendering

5.1 Introduction

As technology and applications evolve, there is increasing demand for *realistic* 3D content display. The traditional graphics pipeline offers significant flexibility but asset creation is usually very time consuming as discussed in Chapter 1. This property makes it hard to apply to the replication of real content especially at the scale of entire rooms or for outdoor scenes. There is a need to make full scenes easy to capture, be used for *free-viewpoint, interactive* navigation. As discussed in Chapters 1 & 2, Image-Based Rendering (IBR) can provide such realistic interactive imagery, but current methods [70, 139, 143], still suffer from many visible artifacts, especially when moving far from the input photos. Novel views are synthesized in IBR by combining warped pixels from input photos; output quality depends on the computation of visibility in the presence of *inaccurate geometry* and on the *blending* method.

Geometry quality. While Multi-View Stereo (MVS) algorithms now provide impressive 3D reconstructions, they cannot achieve the quality required for realistic IBR. A first set of MVS algorithms (*e.g.*, [161]) provides high-quality depth maps containing fine details, while a second set provides better globally consistent geometry, *i.e.*, they estimate a smooth connected surface, even in hard-to-reconstruct (*e.g.*, textureless) regions (*e.g.*, [89, 152]). The meshes provided by either method still result in artifacts if used directly by an IBR algorithm (*e.g.*, Buehler et al. [19]). State-of-the-art IBR algorithms try to overcome this difficulty using *per-view geometric structures*, which may not be globally consistent but achieve good visual quality [24, 70] if blended correctly.

Nevertheless, they still suffer from *geometric inconsistencies*, *outliers*, and *inaccurate occlusion edges*. We develop a new per-view geometry refinement method that overcomes

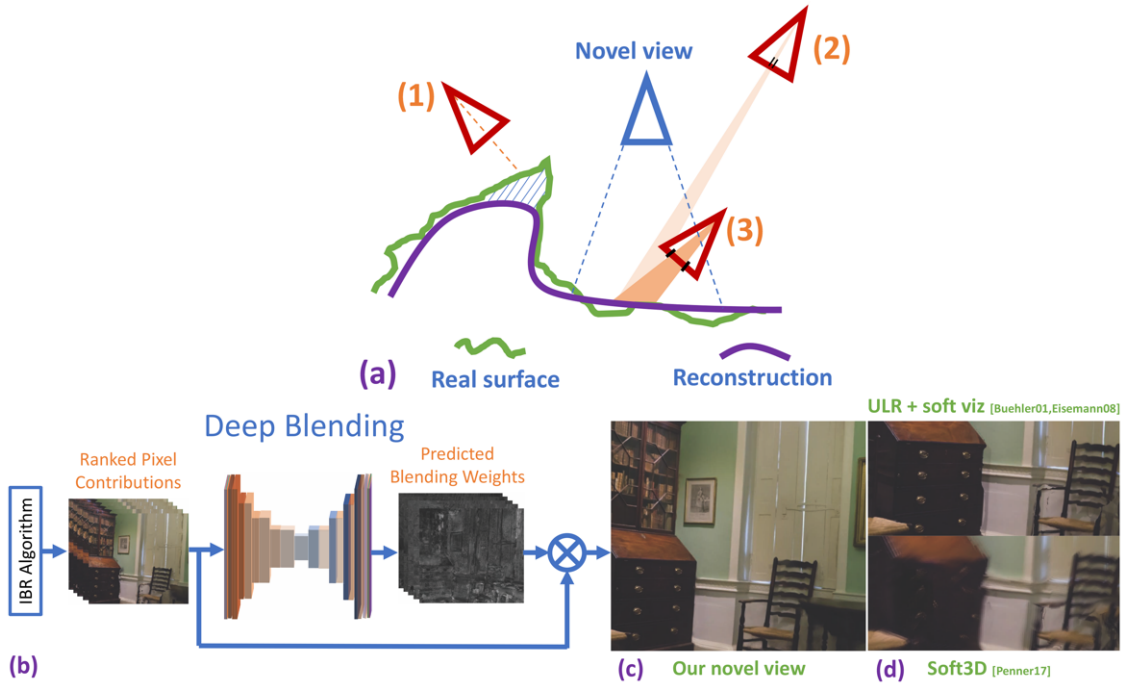


Figure 5.1: Image-based rendering *blends* contributions from different input images to synthesize a novel view. **(a)** This blending operation is complex for a variety of reasons. For example, incorrect visibility (*e.g.*, for the non-reconstructed green surface in input view (1)) may result in wrong projections of an image into the novel view. Blending must also account for, *e.g.*, differences in projected resolution ((2) and (3)). Previous methods have used hand-crafted heuristics to overcome these issues. **(b)** Our method generates a set of ranked contributions (mosaics) from the input images and uses *predicted blending weights* from a CNN to perform *deep blending* and synthesize **(c)** novel views. Our solution significantly reduces visible artifacts compared to **(d)** previous methods.

the shortcomings of each of the two sets of reconstruction methods. We do this by using the information available in the one set of methods as a prior to compensate for information lacking in the other. Our solution, which is separate from our neural network, generates per-view geometry that is of sufficient quality to allow deep blending to succeed.

Blending. With only few exceptions (*e.g.*, [118]), previous solutions use heuristic blending to handle *geometric inaccuracies*, and to correct image seams and ghosting due to view-specific differences in the combined images. Blending needs to correct for artifacts due to incorrect *occlusion edges*, *visible seams* due to *texture stretch/misalignment*,

and *lack of color harmonization*, as well as view-dependent effects from highlights, different exposures, and unsuitable camera selection.

These complex, often contradictory requirements have led prior work to develop case-specific, hand-crafted heuristics that always fail for some configurations.

The main insight is that a data-driven solution is currently a better strategy to effectively satisfy these challenging requirements.

The full pipeline thus introduce a *deep blending* algorithm, leveraging convolutional neural networks (CNNs) that learn *blending weights* that will most reasonably approximate real imagery for novel view synthesis (Fig. 5.1). For this deep blending method to succeed, the underlying geometry used to reproject the original input images in the novel view space should be as faithful as possible. A blending method cannot recover from missing objects or broken reconstruction if none of its input provide a viable candidate.

Contribution. The full deep blending approach mainly introduces an efficient data-driven blending weight computation. In the context of this thesis, while we discuss the full pipeline[1], our contribution is limited to:

- An improved depth-map to mesh conversion algorithm with occlusion edges detection, and a geometry-aware mesh simplification, that together produce high-quality source geometries that greatly improve the input and performance of deep blending for IBR.

Our meshing and simplification solution provides high quality input to the blending network while highly reducing the size of the meshes. The reduced number of triangles of our approach allows for interactive rendering and low memory impact making the full pipeline accessible to consumer GPUs.

For a majority of the scenes tested – both outdoors and indoors – the full deep blending method achieves excellent quality for free-viewpoint navigation. This is thanks to the joint combination of the new blending and to the high-quality inputs enabled by our meshed depth maps. The method is also capable of achieving interactive frame-rates with our highly simplified per-view meshes.

[1]For more details please see the publication, Hedman et al. [72] https://repo-sam.inria.fr/fungraph/deep-blending/

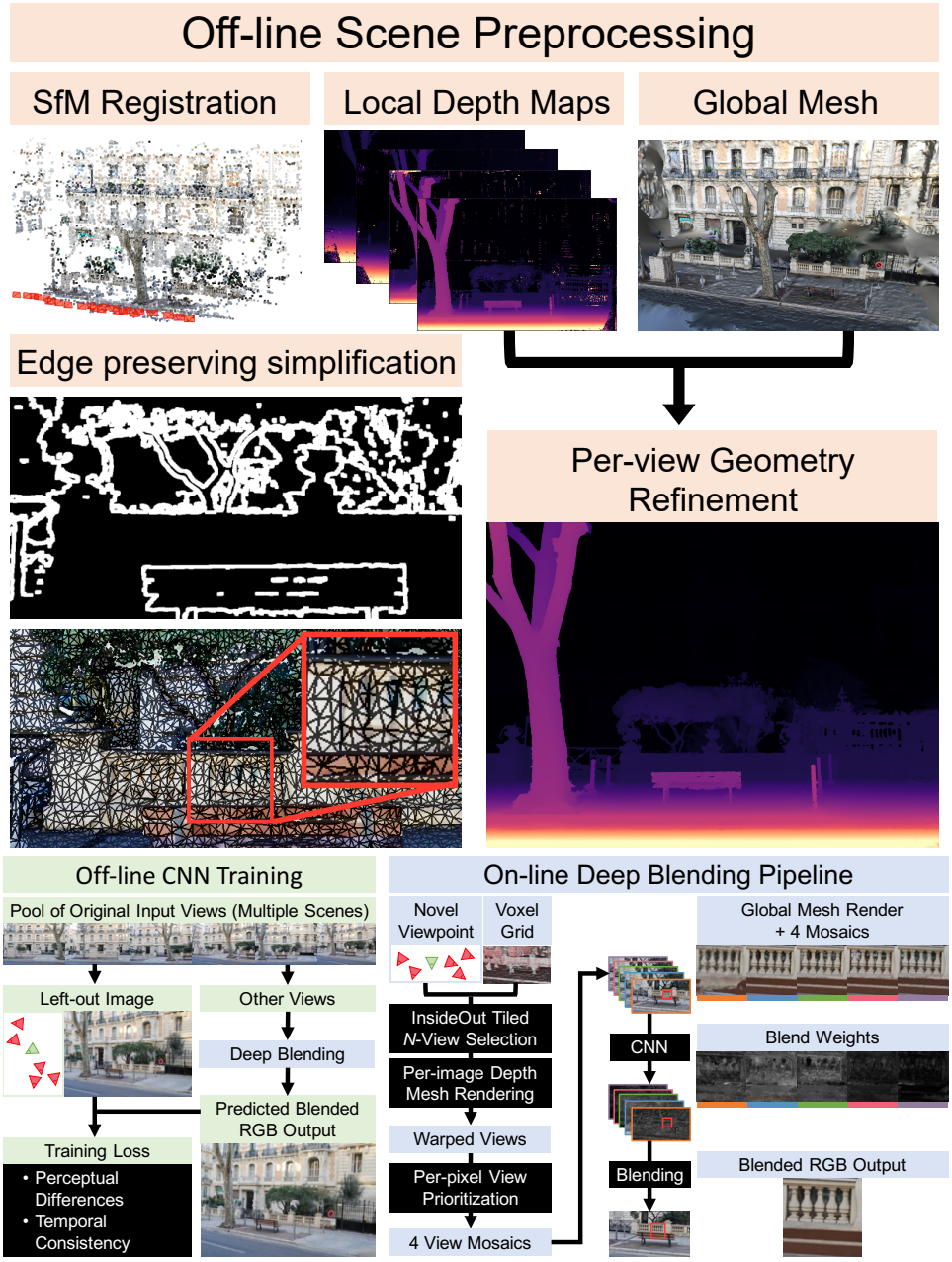


Figure 5.2: Overview of the full pipeline. **Top:** Scene preprocessing entails constructing 1) 6-DoF image positions using SfM, 2) per-image depth maps using MVS, 3) a global mesh, 4) mesh-refined depth maps, and 5) simplified per-view meshes respecting occlusion edges. **Bottom left:** Training uses a perceptual loss to compare our pipeline's output (bottom) for a known viewpoint to the real image; a temporal consistency term penalizes differences after viewpoint perturbation. **Bottom right:** Deep Blending outputs color images for novel scene viewpoints. At each output pixel, InsideOut [70] ranks pixels in the dataset images. Our network takes 4 color *mosaics* of the top samples, plus a global mesh rendering, and outputs per-pixel blend weights. The weighted sum of inputs forms a new color image.

5.2 Overview

The core part of most IBR algorithms is the *reprojection* and *blending* of input images to synthesize a novel view. This blending step is a very complex operation that needs to compensate for inaccurate geometry and for artifacts induced by view- and image-dependent effects. Previous methods use complex heuristics, hand-crafted to work for specific scene configurations. However, they are generally unable to provide realistic free-viewpoint navigation for IBR. Deep learning offers a very powerful tool for coping with variable inputs, while explicitly rewarding high quality blending.

For Deep blending approach to work correctly, we need to provide the network with the best possible geometry to reduce the amount of artifact correction required. When learning blending weights, we restrict the output color to be a positive linear combination of the inputs. This is key in understanding why giving high quality input, close to the expected output, is crucial.

Our input is a set of input photographs of a scene. We first calibrate the cameras using structure from motion (SfM) [160]. Following previous work [70], we use per-view meshes for rendering. The goal of the geometry refinement step is to provide high quality per-view depth map refinement, and generate the compact meshes that respect occlusion edges as much as possible. We achieve this by combining two different MVS methods: COLMAP [161] – based on Patch-Match [8] – that provides fine details in each per-view depth map, and methods based on Delaunay tetrahedralization [89, 152] that provide a smooth mesh estimate in regions where COLMAP fails. The complementary information from each approach is used as a prior for the other one during the per-view depth refinement algorithm. To be able to re-project images into a novel view the depth-maps are meshed. Meshing allows forward re-projection which is very efficient on the GPU. First we detect occlusion edges based on a triangle stretching criteria while avoiding over detecting for surfaces seen at a grazing angle. Then we connect pixels of the depth maps, forming triangles, except at detected occlusions. This gives a clean qualitative mesh, which contains one vertex per pixel. This high number of vertices limits performance, so we introduce a simplification method that preserves the geometric information in image space while allowing efficient reprojection. Our contribution in this thesis is on the meshing and simplification of the refined depth-maps.

Our experiments show that using our simplification method and occlusion detection, our meshes can be simplified at lower scales while maintaining high fidelity to the refined depth maps. This high level of simplification allows interactive rendering for many of our scenes. Compared to previous work, our method significantly reduces errors in meshing and artifacts caused by simplification.

5.3 High-Quality Per-View Meshes for Deep IBR

The input to our method is a set of photos calibrated with SfM [160]. From these, we can use MVS reconstruction to generate per-view depth maps, that can then be converted into per-view meshes for IBR [70]. There are several desirable properties for these meshes: 1) they need to respect occlusion edges, and should be "cut" with no geometry straddling a depth discontinuity, thus avoiding reprojection artifacts in novel views; 2) they should have low triangle complexity to minimize the effect on frame-rate during rendering, and preferably have fewer triangles in the background. When these requirements are satisfied, visibility-related artifacts are reduced during rendering, making the task easier for our deep blending approach, while maintaining interactive frame-rates.

To respect occlusion edges, we introduce a per-view refinement algorithm that fuses and combines information from two different MVS reconstruction methods that have different completeness-vs.-accuracy tradeoffs. To achieve low mesh complexity, we present a view-dependent mesh simplification method that achieves very high compression rates while respecting occlusion edges. Our solution results in significantly better preservation of occlusion edges and much lower mesh complexity overall, compared to previous methods [70].

5.3.1 Per-View Geometry Refinement

To achieve good quality per-view refinement, we combine two complementary MVS methods, one with better detail accuracy, and one with better global completeness. For accurate depth map reconstruction, we use COLMAP [161]. COLMAP resolves small details accurately, but can break down for large textureless regions. For global meshing, we use RealityCapture [152], which is the commercial evolution of the CMPMVS method [89]. This method provides a smooth global mesh estimate, providing informa-

Figure 5.3: **Top:** Globally consistent reconstruction with RealityCapture [152]. **Bottom:** Same scene and viewpoint using COLMAP [160, 161]. We can clearly see that the featureless wall is completely missing from the COLMAP depth maps, but a smooth estimate is provided by RealityCapture. Conversely, the details of the back of the chair are only present in the COLMAP depth maps.

tion in regions that are not reconstructed by other approaches (*e.g.*, textureless content). The strengths and weaknesses of each method are clearly illustrated in Fig. 5.3.

The Patch-Match-based algorithm of COLMAP successfully finds small structures in the scene, but often tends to produce unreliable edges, since it optimizes for photoconsistency using a patch. This is a known problem in stereo algorithms, *i.e.*, the algorithm

Figure 5.4: Merging globally and locally accurate depth maps leads to improved occlusion edge handling and artifact minimization. **Left:** Reference image **Top middle:** Globally complete RealityCapture mesh. **Top right:** Locally accurate COLMAP depth map. **Bottom middle:** Fused COLMAP and RealityCapture depth maps. **Bottom right:** Our refined depth map.

must choose between edge fattening and missing small structures (see Scharstein and Szeliski [158], who point out that approaches focusing on details fail in textureless regions). However, since we already have a globally consistent geometry, we can significantly constrain the search space for our stereo optimization and still obtain reliable results by performing single-pixel optimization.

At a high level, we optimize for per-pixel photoconsistency, but use the geometries estimated from MVS as strong priors to avoid ambiguity. We first fuse the COLMAP depth maps with the RealityCapture mesh wherever COLMAP is uncertain, replacing any unreliable depth by the global geometry. Then, we run a per-pixel photoconsistency optimization to refine the occlusion edges further, similar to InsideOut [70].

The resulting depth maps both preserve small features and provide reliable information in textureless regions. In Fig. 5.4, we show how our approach combines the advantages of both sources of 3D. As this step is presented in another Ph.D thesis [69], we refer the readers to the full paper[2] [72] for more details.

[2]url: https://repo-sam.inria.fr/fungraph/deep-blending/

5.3.2 Occlusion Edges and Meshing Simplification

The refinement step above produces a dense depth map which we use to create per-view meshes for IBR. We now need to simplify the per-view meshes as much as possible while preserving occlusion edges, to minimize the effect on rendering performance. We do this by first identifying occlusion edges, then "cutting" the mesh to preserve them, followed by a mesh simplification step.

5.3.2.1 Meshes from Depth maps

First, we need to reliably detect triangles at occlusion edges. Noisy detection often creates isolated components which degrades simplification. Previous work used a simple 3D distance threshold [70] for this purpose. We found that the min/max aspect ratio measure from [141] more reliably detects degenerate triangles at occlusion edges. To avoid over-cutting background surfaces and surfaces at grazing angles, we also account for depth and the slope of the underlying surface when detecting occlusion edges.

We connect each pixel to its neighbors with a quadrilateral. This results in two different possible pairs of triangles. We keep the pair which minimizes ρ, the maximum aspect ratio of the two triangles. Quads with a large ρ value are likely to be at occlusion edges, since this value corresponds to a large difference in depth. A naive solution would thus be to discard quads with ρ above a threshold. For surfaces seen from grazing angles, this removes too many triangles as the depth varies greatly. We would also discard too many small foreground quads for rendering, since our measure is scale-invariant and we want to keep closely seen objects as clean as possible. To overcome these issues, we modulate ρ by two terms. For grazing angle triangles, we estimate a smooth normal for each pixel and modulate ρ by the dot product between the normal and direction to the viewpoint. We clamp this modulation to avoid a more than 10 times reduction. For foreground pixels, we also modulate ρ by a second term, which is the disparity at the pixel normalized by the median disparity over the image. This modulation is clamped between 50% and 200%. To detect pixel-precise occlusion edges we need a local measure that is scale invariant, sensitive to strong gradients in depth maps and that measures geometry quality. The aspect ratio of a triangle, defined by $q(t) = \frac{\|t\|_\infty}{h(t)}$ where $\|t\|_\infty$ represents the length of the longest side and $h(t)$ the height of the triangle with respect

to this side, is such a measure. For each pixel we consider the quad formed by neighbors. This quad can be meshed by two configurations of two triangles. We define ρ as being the minimizer over the configurations of the maximum of $q(t)$ between the two triangles.

This measure guarantees good occlusion edge detection and good geometry in most cases as pixels with a large ρ correspond to stretched triangle configuration with inhomogeneous depth variation. Nonetheless for surfaces seen at a grazing angle this measure alone, as the ones used in InsideOut [70] or Casual3D [71], leads to over-detection. To overcome this shortcoming we modulate ρ by:

$$d_P = max(0.1, |cos(\vec{v} \cdot \vec{n})|)$$

where v is the view direction. We also modulate ρ by a second term

$$d_D = min(2, max(0.5, D/D_{\text{mean}}))$$

where D is disparity. This term allows to cut more in the background as we have less certainty about the estimated depth and our method is more robust to the lack of per-view geometry than to false geometry. Finally, occlusion edges pixels are defined as those with $\gamma = \rho * d_P * d_D$ larger than the threshold $\tau = 25$ using a hysteresis threshold to avoid instabilities for edges with γ close to τ. Examples of occlusion edge cuts using our method are shown in Fig. 5.5.

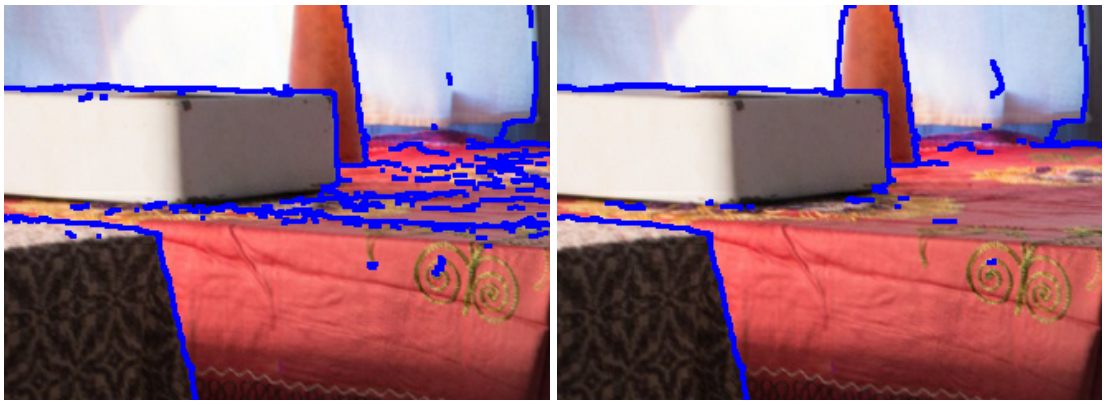


Figure 5.5: **Left:** Occlusion edges, marked in blue, for a depth map using InsideOut [70]. **Right:** Our solution provides much cleaner edges on surfaces at grazing angles.

5.3.2.2 Mesh Simplification

Once occlusion edges have been cut, we create a full-resolution mesh that we simplify based on the edge-collapse method of Garland and Heckbert [56]. There is a rich literature on view-dependent simplification for polygonal models, including, *e.g.*, screen-size error thresholds and silhouette preservation [127], or sophisticated methods to avoid folding triangles [43]. The solution we present here is simpler, since we target the specific case of per-view meshes for multi-view photo datasets.

We construct our per-view meshes to have uniformly sized triangles in image-space. Compared to earlier approaches [70], which strive for uniformly sized triangles in 3D, this better preserves objects close to the camera (with smaller triangles), while also reducing the number of triangles in the background. We achieve this by dividing the standard quadric simplification cost by the squared distance between the edge and the camera.

One frequent problem with quadric error metrics are "folding triangles": we avoid this by preventing normal deviation caused by collapse over 60°. Finally, we multiply the weight of occlusion edges by the targeted simplification ratio to encourage their preservation. As we can see in Fig. 5.6, we can achieve extremely high simplification factors (up to 256x) with little loss in quality.

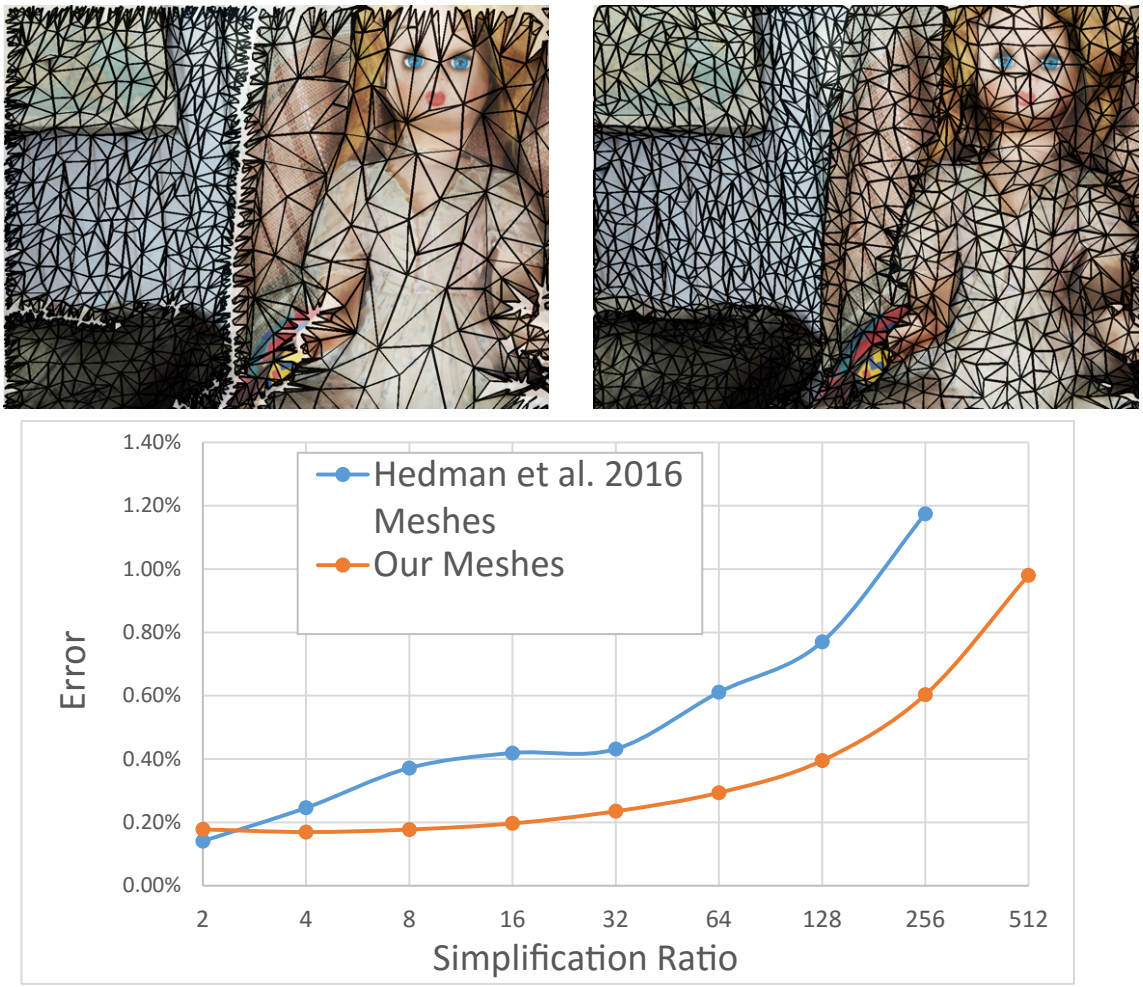


Figure 5.6: **Top Left:** Simplification using InsideOut meshing [70]. **Top Right:** Our solution, which provides higher uniformity in image space and preserves occlusion edges with good quality. Meshes were simplified by a factor of 256x. **Bottom:** Mean relative error in 3D at different simplification ratios for a set of 20 meshes from different scenes. More than 80% of meshes could not be simplified at 512x using InsideOut.

5.4 Rendering algorithm

Input and Network. To allow a per-frame interactive rendering loop that includes a CNN evaluation, we choose a U-net [155] architecture, and generate a fixed set of inputs to the CNN. For rendering, we build on InsideOut [70], which at each output pixel selects a variable number of input photos to blend into a final image (Fig. 5.2). In our rendering loop, we rank these per-pixel selections to generate a fixed number of *mosaics*

that are blended into the novel view. Each pixel of the first mosaic contains the color value of the best selected pixel, the second mosaic contains the second best, and so on. Pixels are ranked according to an IBR cost [70] and a new visibility term.

Training. Generating training data for IBR is difficult since it is hard to obtain ground truth. We overcome this problem by using training data for our supervised learning of the CNN weights in a non-traditional manner: the same photo serves, at times, as one of the *inputs* to the mosaic-building step, or it is held-out so it can serve as the ground truth *output* that the network tries to reconstruct from mosaics of other input photos. We generate a large dataset of input images through round-robin use of this hold-out strategy, and through data augmentation. The test results we show are distinct samples (*i.e.*, novel viewpoints) never seen in training. Finally, to achieve good visual quality, to overcome alignment issues and to reduce flickering, we use a perceptually-motivated loss [90] and introduce a temporal coherence term. Our experiments show that the network training can be run on a general set of images from training scenes, and that performance is stable even when no images for a given testing scene were seen during training.

We refer the readers to the original paper for details about the rendering pipeline.

5.5 Implementation, Results and Experiments

We implemented the full system in C++ and OpenGL, combined with the C++ interface of TensorFlow [1] for runtime blend evaluation. The rendering pipeline is based on the code of InsideOut [70]. The meshing and simplification were implemented in C++ and parallelized using OpenMP independently from the rest of the pipeline.

To maximize the pipeline's potential for interactive rendering times, we implemented custom TensorFlow operations that directly copy the input and output OpenGL textures to and from the network in on-device GPU transfers, making use of the OpenGL/CUDA interop interface available in the CUDA version 9.0 library. A custom CUDA kernel was implemented to increase the speed of 2D spatial upsampling; all other network components used TensorFlow's native operations with cuDNN 7.1 [27].

5.5.1 Comparisons

In Figs. 5.7 we show results from 8 scenes, comparing our method with other end-to-end IBR systems:

RealityCapture: The textured mesh from RealityCapture [152].

Selective IBR: The superpixel IBR approach by Ortiz-Cayon et al. [139] using the RealityCapture mesh as input geometry.

ULR: Unstructured Lumigraph Rendering [19] with soft visibility [42] and the RealityCapture mesh as the geometry proxy.

Soft3D: The novel view synthesis algorithm by Penner and Zhang [143], using their custom soft 3D reconstructions.

InsideOut: The indoor IBR system by Hedman et al. [70] using their custom depth-sensor based 3D reconstructions.

5.5.2 Evaluation of Deep Blending vs. Geometric Refinement

We evaluate the relative effect of each of our steps, namely deep blending vs. our improved geometric refinement and meshes. We implemented an IBR algorithm similar to Hedman et al. [70], with two variables: InsideOut per-view refinement vs. our refinement, and InsideOut-style blending using our depth-modulated cost (which we call *heuristic blending* from now on) vs. Deep Blending, for a total of four configurations. For a fair comparison with our refinement, InsideOut refinement is performed using only the RealityCapture mesh.

In Fig. 5.8, we compare heuristic blending vs. deep blending using our refinement, isolating the effect of blending only, and observe there are several visible artifacts that are corrected by the neural network. In Fig. 5.9, we show the effect of using our refinement vs. InsideOut refinement, but using Deep Blending for both, isolating the effect of refinement only. We also show InsideOut refinement with heuristic blending that is clearly worse.

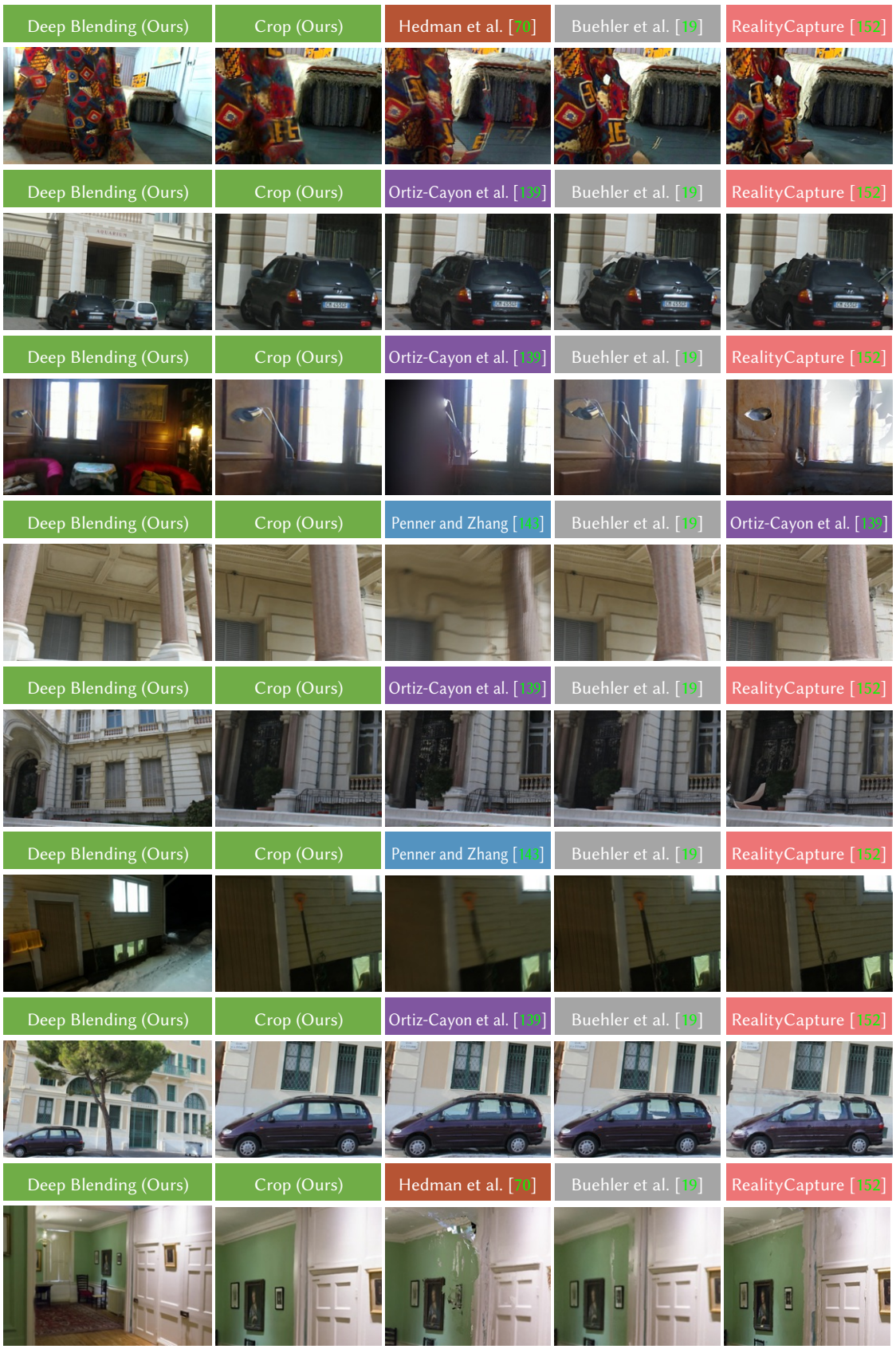


Figure 5.7: Results from 8 scenes. Left: Full novel view from our solution, followed by a crop using our method. The remaining three columns show previous methods. In most cases, errors due to inaccurate geometry result in visual artifacts such as ghosting, incorrect edge reconstruction as well as blur.

Figure 5.8: Our deep blending network vs heuristic blending, both using our refined per-view meshes. **Left:** Our full method, showing the difference in quality due to Deep Blending. **Right:** Heuristic blend cost, see original papers for details.

5.5.3 Performance

We evaluate the runtime performance of our interactive rendering pipeline at a resolution of 1280×720 px on a system with a 3.47 GHz Intel Xeon processor and an NVIDIA GTX 1080Ti GPU. (Note that the videos in our supplemental are rendered offline at a resolution of 1920×1080 px.) Table 5.1 shows average per-frame execution times of our interactive pipeline over several scenes. Runtimes are specific to the scene and are not

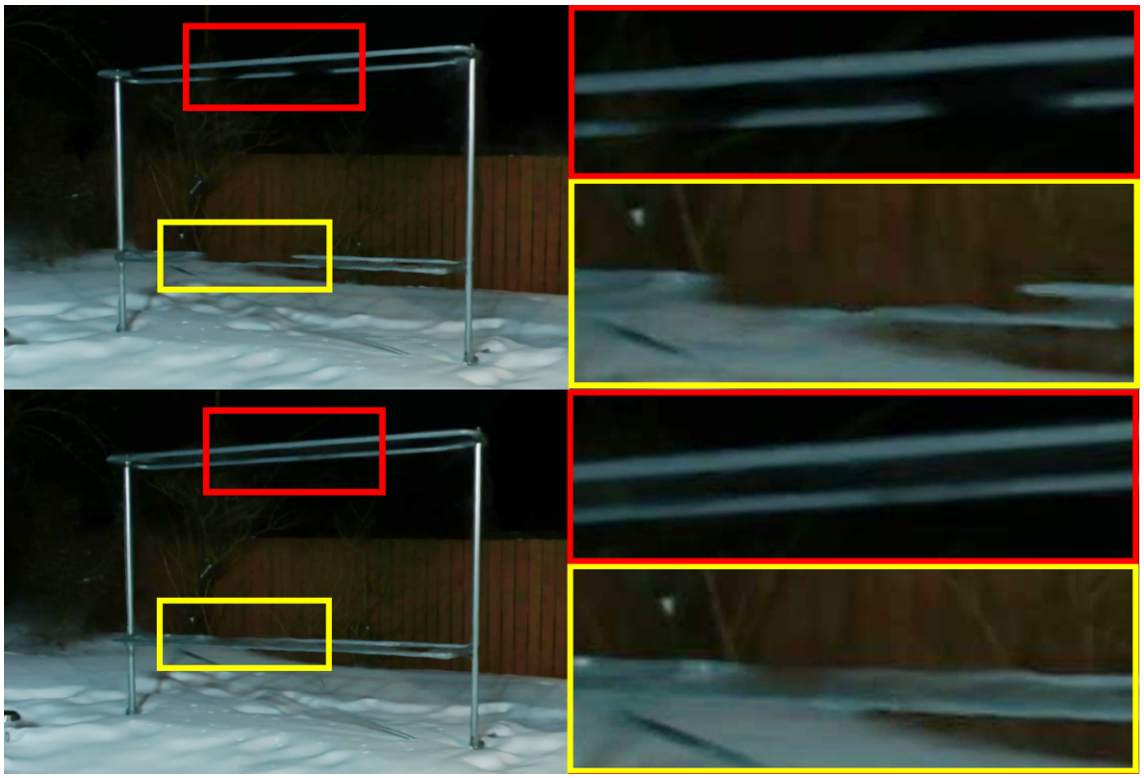


Figure 5.9: Different geometry refinement approaches with our deep blending network. **Top:** Using the geometry refinement and meshes from Hedman et al. [70]. **Bottom:** Our full solution, which better recovers thin structures.

driven only by the deep blending network component. For smaller scenes, especially indoor scenes, our implementation achieves greater than 30fps performance, which falls off gradually as scene complexity increases. For large outdoor scenes, we observe that the primary bottleneck is in the voxel-wise camera selection, which in our current implementation is tied to a fixed-size spatial subdivision. An adaptive or hierarchical approach is an interesting future direction for this issue.

Preprocessing times for our pipeline varied from scene to scene but generally finished overnight. We treat network training separately, which takes two days, but does not necessarily have to be re-run for each new scene. Example processing times for the Creepy Attic scene are shown in Table 5.2.

Table 5.1: Average runtimes (ms/frame) over 100 frames for our IBR system.

Scene	Non-network Runtime	Total Runtime	Scene	Non-network Runtime	Total Runtime
Museum-1	6.2	26.2	Hugo-1	14.1	35.6
Creepy Attic	7.1	26.8	Night Snow	15.3	33.0
Dr Johnson	12.4	33.6	Boats	19.7	47.6

Table 5.2: Preprocessing times for Creepy Attic (249 images at 1228×816) using a 4-core Intel Core i7 processor and an NVIDIA Titan X GPU.

Component	Runtime	Component	Runtime
COLMAP SfM	1h	Depth map Refinement	0.75h
COLMAP MVS	5.25h	Meshing	1.5h
RealityCapture MVS	0.5h	Network Training	37h

5.6 Limitations and Future Work

Limitations. Full pipeline: Our method gracefully degrades when there is very little 3D geometry from the initial 3D reconstruction; we show an example in Fig. 5.10. Our method will also tend to flicker when the differences in exposure are very large and inconsistent. The network occasionally creates blur, typically in regions with missing geometry or where difficult decisions need to be made (highlights, resolution mismatches etc.). Nonetheless, in the vast majority of the 19 scenes, the full pipeline outperforms all previous solutions, although in some cases one artifact is traded for another (e.g., blur instead of seams).

Meshing and simplification: The simplification ratio is fixed in our method. In images with a lot of occlusion edges, the number of connected components may increase drastically leading to poor performance, if simplifying a lot, as geometry is spent on edges preservation. Another issue with our approach is that if some meshed parts are never good candidates for reprojection they are still rendered and use geometry budget when simplifying. We discuss possible avenues to address these limitations in future work, below.

Figure 5.10: **Left:** From the rendering of the mesh, we see that part of the geometry of the car is completely missing. **Right:** Despite significant visual improvement, our method cannot hallucinate the missing geometry.

Future work. In future work, we would like to address the problems outlined above. To reduce blur, one possible avenue would be the use of an adversarial loss [84]; it is unclear how well such an approach would deal with temporal coherence. To reduce flickering for very large exposure inconsistencies would probably require an effective pre-processing step for color harmonization, while respecting differences due to view-dependent materials.

Achieving real-time performance, especially in the context of stereo viewing (*i.e.*, at least 90 fps), requires performance improvements. This can partly be addressed with upgraded hardware. A solution to the meshing problem that would also benefit performance would be to preprocess the depthmaps to remove parts for which many better candidates exist in other images. This would allow removing geometry that is never used to feed the network. This would require careful treatment of correspondences between the images.

5.7 Conclusion

In this chapter we have presented our meshing and simplification method used for Deep Blending. We demonstrate large improvement compared to the state of the art both in quality and fidelity to input depthmaps while allowing higher simplification ratios.

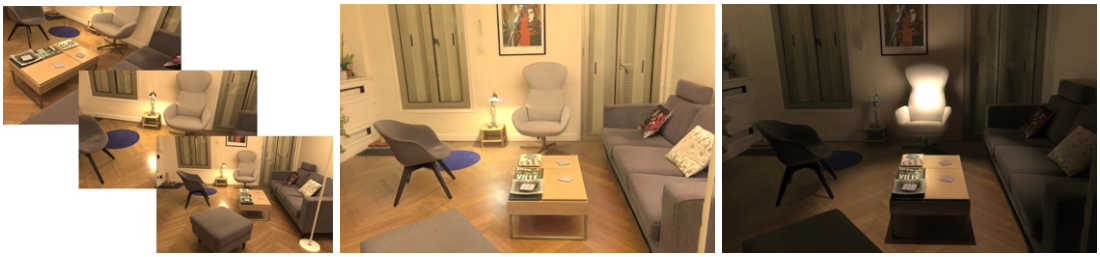
Overall this allows better quality input to the network while improving reprojection performance. These two factors enhance the usability of the full deep blending pipeline as they strongly impact interactivity and rendering quality. The results in this chapte also shows that giving high quality heuristics inputs to a network positively impacts its performance, allowing it to generalize even with small datasets. In the next chapter we present a relightable neural renderer that heavily relies on this idea. We use IBR heuristics as inputs to a network both to get high quality reprojection but also to enable material analysis.

Chapter 6

Relightable Neural Rendering of Multi-view Indoor Scenes

6.1 Introduction

In the previous chapters we introduced and discussed methods to edit and render multi-view data. First, in Chapter 3 we treated the problem of object removal, then we introduce a method to relight and compose outdoor scenes in Chapter 4, and we finally discussed a deep learning based IBR algorithm focusing on the geometric representation in Chapter 5. Each of these methods contributes to better, more flexible IBR on their own. But they also treat each problem separately. To allow real ease of use and editability of IBR methods, ideally, all these contributions should be unified in a single algorithm.



(a) three of n input views (b) novel view synthesized by our algorithm (c) the same novel view with modified illumination

Figure 6.1: Our neural algorithm takes as input a multi-view capture of a real scene under a single lighting condition (a). It re-renders the scene from novel viewpoints, accounting for view-dependent glossy effects, e.g., on the floor (b) and on the table (c), allowing free-viewpoint navigation. Our network internally builds an implicit representation of the scene materials, based on complex input feature maps we compute with a novel hybrid image- and physically-based rendering algorithm. This enables a user to insert new lights in the scene and/or turn off the original illumination; the network produces a new rendering of the scene under the modified lighting (c). Note how the light – that was on in the input photos – has been turned off with our method.

This chapter is a first step toward that goal. While we do not treat all editings at once we introduce a method that is capable of both relighting and novel view-synthesis in indoor scenes. Existing view-synthesis methods [72, 131] do not allow the user to *change the illumination* of a scene, while relighting techniques often require an involved capture process to accurately measure the geometry (e.g., laser scanning), materials and illumination (e.g., via light probes [34, 179]). These measurements are typically used in an expensive inverse rendering model to re-render a new image [204].

We introduce a *neural rendering* algorithm in which a convolutional network learns to create an implicit representation of the materials and lighting from a multi-view capture of an indoor scene under a fixed *source* lighting condition. The user then specifies a *target* lighting condition and, after a short pre-computation, the network can interactively synthesize new viewpoints under the novel lighting conditions. Our algorithm combines the best of two worlds: *inverse rendering* — that decomposes scenes into explicit models re-rendered with physically-based methods, and *image-based rendering* — that renders new views by blending multiple input images. Compared to image-based techniques, our algorithm can handle more complex glossy materials, and it can change the scene's illumination, via a lightweight physical simulation. Yet, unlike inverse rendering, it is faster and requires no explicit material model.

6.2 Overview

We work with wide-baseline multi-view captures (typically 100–300 photos) from which we compute a 3D mesh by Multi-View Stereo [50, 89, 152]. This means we start with richer information about the materials and lights that make up the scene than single-image methods (*e.g.*, [200]). We summarize this rich information as a set of *feature buffers* that are reprojected in the novel view, and passed to our convolutional network. We make the simplifying assumption that the scene lighting can be decomposed into the sum of two components: view-independent diffuse and view-dependent glossy. Our input buffers encode the salient parts of the illumination under this decomposition, for both the source and target conditions.

More precisely, our features consist of three major elements that together help the network learn an implicit representation of lighting and materials, and synthesize the final image.

First, to treat the diffuse component of relighting, we use elements of physically-based rendering to estimate the *source* diffuse irradiance buffers via a lightweight Monte Carlo integration. From this diffuse irradiance, we compute an approximate albedo stored in the mesh. This albedo mesh, while being a very rough estimate of the scene, can be used to compute approximate *target* diffuse irradiance buffers via path-tracing given user-specified synthetic light sources. The network learns to change the diffuse illumination by comparing and combining reprojections of the source and the target buffers. In particular this includes cast shadows, global illumination, color bleeding, overall illumination levels, etc.
Second, we use a fast single-ray mirror reflection computation to generate source and target *mirror images*. Source mirror images, along their corresponding source input images, facilitate learning the relationship between diffuse and glossy illumination in the input. These provide some material understanding, and can be precomputed. The network then uses the target mirror image, generated at render time, to reproduce glossy effects on the fly. Third, inspired by image-based rendering, we introduce a new scheme to reproject the input images and their corresponding feature buffers; this naturally enables the network to process any novel viewpoint, giving free-viewpoint navigation capabilities. Our reprojections of input images provide good estimates of the novel view under the source lighting condition and also capture the view-dependent image variations due to glossy materials.

We train our network entirely on synthetic data. To ensure the trained network transfers well from synthetic to real scenes, we use the exact 3D geometry to compute our ground truth, but we also reconstruct each training scene with MVS to provide the network with the approximate geometry available at test time.

Ours is the first solution that can relight complete indoors scenes with glossy materials and indirect lighting effects. We demonstrate our method on several captured scenes, showing sessions where the user adds lights, or turns-off the original scene lights.

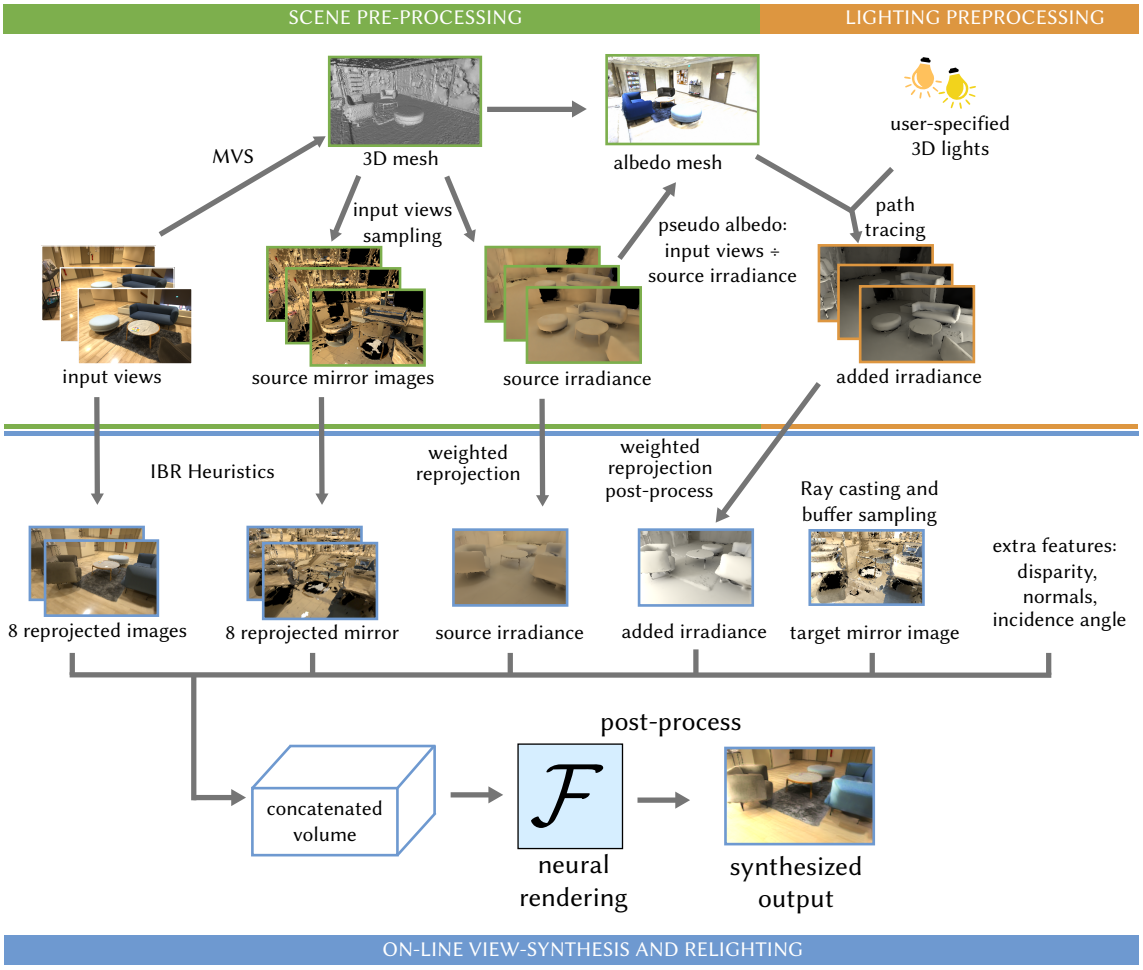


Figure 6.2: Overview of our pipeline. We first perform scene pre-processing of our multi-view dataset and MVS geometry to precompute source diffuse irradiance, source mirror images and an albedo mesh, stored with the scene. For a new lighting condition target added irradiance is precomputed. Online view synthesis is achieved by reprojecting the information into a novel view using our neural network to synthesize a relit image.

6.3 Multi-view neural relighting

We start from a set of calibrated undistorted 14bit photographs and an approximate 3D mesh reconstructed with MVS, ensuring that lights are visible in the photos. Our goal is to allow a user to alter the scene's illumination by adding new light sources in the virtual 3D space and/or turning off all the captured lights.

As the user navigates freely in the scene, our algorithm synthesizes realistic novel views

and recreates convincing glossy reflections for both the original and synthetically added light sources. This relighting process requires an estimate of the *source* (*i.e.*, at the time of capture) and of the *target* illumination. Instead of explicitly modeling the lights and materials — leading to a difficult inverse problem and requiring costly physically-based rendering — we train a convolutional neural network to learn an implicit representation suitable for relighting and view synthesis. We derive a simplified illumination model (Section 6.3.1) that guides our design of easy-to-compute input features for the network. We compute these features with a novel hybrid rendering algorithm that blends elements of physically- and image-based rendering (Section 6.4). Figure 6.2 gives an overview of our pipeline.

6.3.1 Simplified illumination model

The rendering equation for a static, non-emissive surface is given by:

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o) = \int f(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i) \boldsymbol{\omega}_i \cdot \mathbf{n} d\boldsymbol{\omega}_i, \tag{6.1}$$

where L_o and L_i are the outgoing and incoming radiance at a 3D point $\mathbf{x}$ with normal $\mathbf{n}$ and $\boldsymbol{\omega}_o$, $\boldsymbol{\omega}_i$ are the outgoing and incoming directions respectively. We make the simplifying assumption that the bidirectional reflectance distribution function f is the sum of two components: a diffuse albedo term $a(\mathbf{x})$ which is independent from the incident and outgoing angles, and a view-dependent glossy component $s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ which we assume is non-zero only in a small angular neighborhood around the mirror direction at $\mathbf{x}$:

$$f(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx a(\mathbf{x}) + s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o), \tag{6.2}$$

This is a reasonable assumption for isotropic materials. Integrating over incident angles, diffuse irradiance is given by:

$$E(\mathbf{x}) = \int L_i(\mathbf{x}, \boldsymbol{\omega}_i) \boldsymbol{\omega}_i \cdot \mathbf{n} d\boldsymbol{\omega}_i, \tag{6.3}$$

and the near-mirror glossy illumination term is:

$$S(\mathbf{x}, \boldsymbol{\omega}_o) = \int s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i) \boldsymbol{\omega}_i \cdot \mathbf{n} d\boldsymbol{\omega}_i. \tag{6.4}$$

Combining with Equation (6.1) gives us a linear decomposition of the outgoing radiance:

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o) \approx a(\mathbf{x}) E(\mathbf{x}) + S(\mathbf{x}, \boldsymbol{\omega}_o). \tag{6.5}$$

Our relighting strategy is to design an efficient rendering algorithm to estimate approximations of the diffuse irradiance E and near-mirror view-dependent illumination S from which a neural network can infer an implicit representation of the lighting and materials, and use it to re-render the scene with complex glossy effects and illumination changes.

Note the intrinsic link between glossy behavior in relighting and view synthesis: moving the light or moving the viewpoint for the same angle leads to the same radiance. Our representation implicitly exploits this link for both relighting and view synthesis with glossy surfaces.

6.3.2 Illumination representation as 2D feature maps

The global nature of light transport makes it extremely difficult for a convolutional network to decouple the intrinsic material properties from the scene illumination. To simplify the learning problem, the network takes as input a representation of the scene that approximates E and S from Equation (6.5). We construct this representation around three major components.

First, we estimate the source diffuse irradiance E by stochastic integration from the input images. This allows us to compute an approximate source diffuse albedo mesh, and in turn, to compute the diffuse target irradiance modifications from the user-modified illumination using path-tracing. The very rough albedo mesh is only used for the target irradiance computation and not given as input to the network.

Second, we compute *mirror images* that estimate the energy in the mirror direction at each surface, for both source and target illuminations. Under our near-mirror assumption for S, the mirror images help our network measure the glossy component of scene materials as well as synthesize new reflections.

Third, we reproject the multi-view RGB images in the novel view to synthesize, so we can exploit the wealth of real-world light transport cues they provide. Our reprojection scheme uses blending heuristics to get good texture information but also tries to maximize color entropy, leading to more diverse observations of view-dependent effects. In Section 6.4, we describe our rendering algorithm to compute this representation.

All source information is computed once at capture time and stored with the scene. For any novel lighting condition, a short preprocessing time (a few minutes) is required to

compute the target diffuse irradiance modification (addition). The final rendering of the relit novel view is performed by a neural network that takes all the above elements as input to synthesize a novel view.

6.4 Generating the network inputs

Our network consumes different image buffers that all need to be represented in the novel view image space. A first natural set of buffers to use is the set of input views, $I_{1..n}$. As they lie in input view space for each input camera, they need to be reprojected in the novel view space; this reprojection is done through an operator noted Π_I. It produces eight image buffers through blending and pooling in novel view space. For a given scene and user-specified target illumination we also pre-compute static information, specifically the source mirror images M_i^{src} to help with material understanding, that are also reprojected using Π_I. To guide relighting, we also compute a diffuse source irradiance map E_i^{src} that is an estimate of the irradiance in the input view i, the user defined added diffuse irradiance E_i^{add}, and finally the diffuse irradiance to remove E_i^{rem} used to specify whether or not to keep the original lights in the relit result. These are all precomputed for each input view $i \in 1..n$ once for each scene and cached. The various irradiance maps are then reprojected in novel view space with another operator Π_B that produces one image for each kind of irradiance map. Finally, we also compute one target mirror image M^{tgt} and a few additional buffers $addB$ directly in the novel view space. The final relit novel view $\mathcal{R}$ can thus be expressed as:

$$\mathcal{R} = \mathcal{F}\Big(\Pi_I(I_{1..n}), \Pi_I(M^{src}_{1..n}), \Pi_B(E^{src}_{1..n}), \Pi_B(E^{add}_{1..n}), \Pi_B(E^{rem}_{1..n}), M^{tgt}, addB\Big)$$

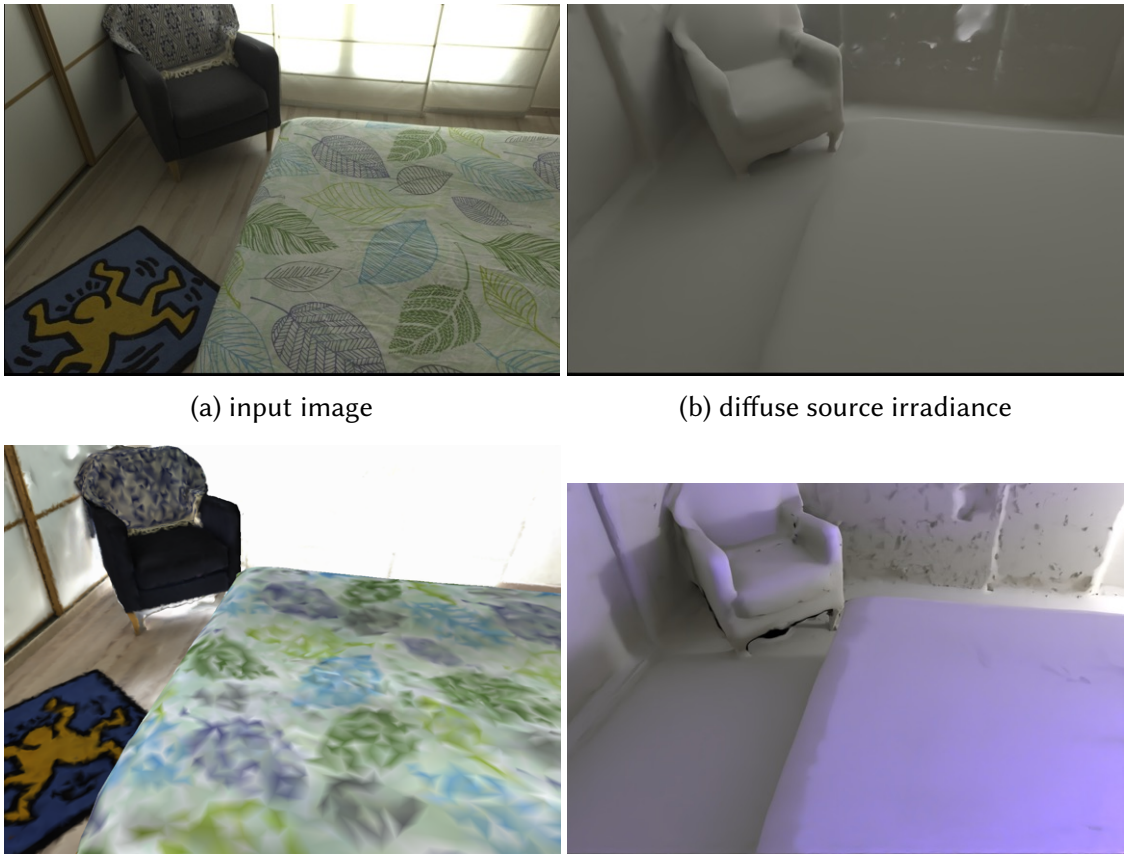
Where $\mathcal{F}$ is our neural network.

Algorithm 1 summarizes our rendering pipeline.

6.4.1 Diffuse source irradiance - $E^{src}_{1..n}$

We compute a source irradiance map E_i^{src} for each input view by stochastic integration. For each pixel in each viewpoint, we cast a ray in the 3D scene. If the ray intersects the geometry, we randomly sample 128 directions in the hemisphere above the intersection

and integrate the pixel colors from the other input images. Specifically, we sample the image for which the camera–intersection vector forms the smallest angle with the secondary ray (this is the same test as the target case, see Fig. 6.4). If the secondary ray does not reproject to any view, we ignore it during integration. Our irradiance maps can be noisy, so we denoise them using Optix [23] to avoid magnifying the noise when reprojecting. An example source irradiance map is shown in Fig. 6.3b.



(a) input image (b) diffuse source irradiance

(c) albedo mesh (d) added diffuse irradiance

Figure 6.3: We estimate the diffuse irradiance (b) corresponding to the input illumination by stochastic integration, sampling the input views (a). This allows us to estimate an approximate albedo by dividing the input image data by the estimated irradiance, which we store at the vertices of the 3D mesh (c) . When adding a new light in the scene, we can compute the added diffuse irradiance by path-tracing using the albedo mesh (d). Here the light was added on the lower right of the view. Even though in this example the scene illumination is mostly neutral, our irradiance maps contain full RGB information.

6.4.2 Approximate diffuse albedo mesh

We compute the target added irradiance using physically-based rendering. This requires an estimate of the scene albedo. Using our estimate of the diffuse source irradiance E_i^{src}, we compute a per-view approximate albedo map as $a_i = I_i / E_i^{src}$. We want to use this information for path-tracing the target added irradiance; we thus need a mesh colored with albedo. To do this, we reproject the albedo maps onto the mesh and collect the average of all the reprojections that pass the depth test, weighted by the inverse distance to the corresponding camera. We store the average albedo at mesh vertices (see Fig. 6.3c for a visualization). This estimate of the albedo is rough but it allows to compute good estimates of global irradiance as the influence of this albedo is only for second bounces and more.

6.4.3 Path-traced added irradiance - $E_{1..n}^{add}$

The user provides light sources to be added as 3D area lights with constant emittance. For each view we compute the new irradiance image E_i^{add} by bi-directional path tracing using the albedo mesh in Mitsuba [88]. An example of target added irradiance map is shown in Fig. 6.3d.

When the user wants to add several lights, we can either compute a single lighting setup with all the lights, or for each light l to add, compute a full set of added irradiance maps ($E_{1..n\ l}^{add}$). The sets are then linearly composed at render time by the user using sliders to control colors and intensity, leading to the composed added irradiance maps and set which we respectively refer as E_i^{add} and $E_{1..n}^{add}$ for simplicity.

6.4.4 Removed irradiance - $E_{1..n}^{rem}$

The user can optionally request to turn off all the original scene lights, allowing more drastic changes in lighting. To specify it to the network we have a third irradiance map, E_i^{rem}, for each input view. If the source lighting is to be kept, E_i^{rem} is equal to zero, otherwise it is equal to E_i^{src}. This map is interpreted as the irradiance to be removed from the input.

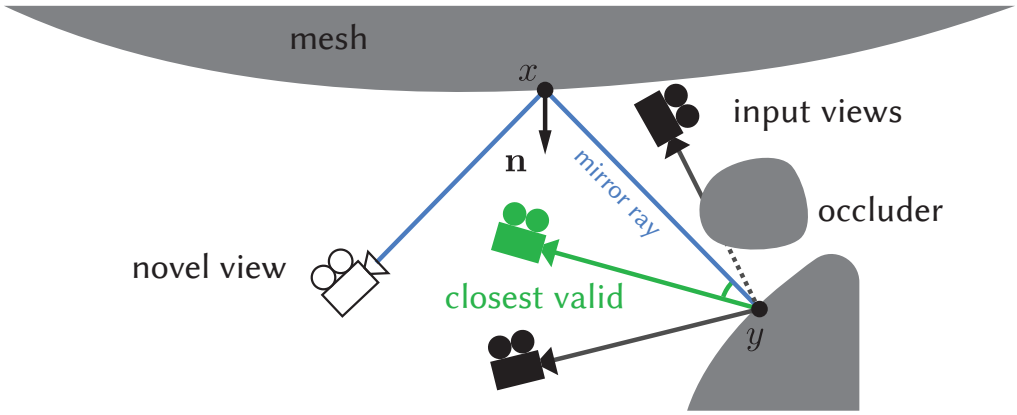


Figure 6.4: To compute the mirror value at a reprojected point x in the novel view, we sample pixels from the input view that best align with the mirror direction, making sure to ignore cameras that do not see the intersection y between the mirror ray and the geometry using a depth test.

6.4.5 Source mirror images - $M^{src}_{1..n}$

Relighting requires our network to learn to decouple view-dependent effects from the diffuse illumination. However, we cannot expect a convolutional network to be able to sample distant parts of the scene. We make the simplifying assumption that glossy effects are concentrated around the mirror direction (Section 6.3.1) and compute first-bounce mirror images M^{src}_i for each input view $i \in 1..n$. More precisely, we cast a single ray per pixel into the scene, for each input camera i. We re-project the reflected ray — mesh intersection (if any) into all the other views. Amongst them, we retain the camera whose viewing direction is closest to the direction defined by the reflection, say j, making sure to ignore viewpoints from which the reflected point is not visible using a depth test. See Figure 6.4, for an illustration. Sampling I_j at the reprojected intersection gives us the color value for M^{src}_i. With these mirror reprojections, we observe the scene from two material models: the original materials I_i and a pure mirror variant M^{src}_i. When reprojecting several version of these two maps (using Π_I) and comparing them, the network will be able to learn to assign the correct energy to the glossy BRDF component (Fig. 6.5).

(a) input view (b) source mirror image

Figure 6.5: For in each input view (a), we compute a mirror image (b) by tracing a ray in the mirror direction of the first mesh intersection and sampling the image whose camera—intersection vector is closest to the mirror ray in angle (see also Figure 6.4). These mirror images help the network analyze the materials in the scene, and identify complex glossy effects like the salient highlight on the floor.

6.4.6 Interactive reprojection and rendering - Π_I and Π_B

When rendering a novel view, we reproject the per-view feature maps as well as the raw input images into the novel view before passing them to the neural network. We also compute a target mirror image M^{tgt} interactively which the network uses to synthesize new glossy reflections. As reprojecting all the views into novel-view space would lead to a huge volume with many empty pixels, we need not keep track of all the input views content. Instead, we introduce two reprojection algorithms. The first one (Π_I) builds a small constant set of composites that are assembled from pixels sampled in the n inputs. The second one (Π_B) is used for the diffuse irradiance maps and produces only one image.

This design of the reprojection algorithms (Π_I,Π_B) satisfies the following constraints: 1) maximize the reprojected image resolution by selecting pixels from viewpoints that are close to the novel view, 2) maximize coverage to limit the number of pixels with no data, 3) maximize temporal coherence, *i.e.*, avoid popping of input views and temporal discontinuities when changing viewpoint. And for Π_I only: 4) maximize viewpoint diversity so we can observe the non-diffuse scene materials from multiple angles (*i.e.*, we try to avoid views that are clumped together), 5) yet keep the number of network inputs small to limit running time and memory consumption.

Our reprojected views are obtained by blending or pooling pixels from the n input views.

6.4.6.1 Multiple reprojections for texture quality and material observation - Π_I

Our reprojection scheme for the input views and corresponding mirror images outputs eight different images, $\pi_k\ k \in \{1..8\}$. Four of these, $\pi_k\ k \in \{1..4\}$, are produced to ensure view synthesis quality and are computed by blending input views. The last four, $\pi_k\ k \in \{5..8\}$, provide observation of the material appearance variation. Π_I is thus defined as follow:

$$\Pi_I(I_{1..n}) = \{\pi_k(I_{1..n}) | k \in \{1..8\}\}$$

Heuristic blending for view synthesis. For a given novel view v^* we compute four reprojections $\pi_k(I_{1..n})\ k \in 1..4$ as a blending of reprojected input views:

$$\pi_k(I_{1..n})(p) = \frac{1}{W_k} \sum_{i=1}^{n} w_k(v^*, v_i, 3D(p), n(p)) * I_i(R(p, v_i))$$

Where p is a given pixel in novel view, $R(p, v_i)$ is the reprojection of p in I_i, $w_k(v^*, v_i, 3D(p), n(p))$ is the weight for the sample, depending on the novel view v^*, the sampled view v_i, the 3D position corresponding to the pixel $3D(p)$ and the normal at this point $n(p)$. W_k is the sum of the weights for normalization.
We define the four weighting schemes as follow:

$$w_1(v^*, v_i, 3D(p), n(p)) = \frac{1}{\|pos(v^*) - pos(v_i)\|_2} \tag{6.6}$$

$$w_2(v^*, v_i, 3D(p), n(p)) = ((pos(v^*) - 3D(p)) \cdot (pos(v_i) - 3D(p)))^2 \tag{6.7}$$

$$w_3(v^*, v_i, 3D(p), n(p)) = \frac{1}{\|pos(v_i) - 3D(p)\|_2^2} \tag{6.8}$$

$$w_4(v^*, v_i, 3D(p), n(p)) = e^{\frac{(pos(v_i) - 3D(p)) \cdot n(p)}{\|(pos(v_i) - 3D(p))\| * 0.3}^2} - 1.0 \tag{6.9}$$

w_1 favors cameras that are close to the novel view. w_2 favors cameras that see the point from the same direction as the novel view. w_3 is not view dependent and favors images that see the point from a short distance, maximizing texture information. Finally w_4 favors observation of the surface from an orthogonal viewing direction, which reduces errors due to erroneous depth. These schemes naturally provide different observations of the material behavior: for instance w_4 reduces Fresnel effects, and is temporarily smooth. Only w_1 and w_2 are view dependent and they smoothly blend all the inputs to avoid popping.

Pooled luminance for material understanding. The last 4 reprojections aim to fulfill our goal of covering the variation in outgoing radiance due to glossy materials. For each texel in the novel view, we sample the luminance from all the input images Ii, $i \in 1, \ldots, n$ and sort them. We define these reprojections as:

$$\pi_5(I_{1..n})(p) = Ij1(R(p, v_j1)) \text{ with } j_1 = \underset{i \in \{1..n\}}{\text{argmax}}\ lum(I_i(R(p, v_i))) \tag{6.10}$$

$$\pi_6(I_{1..n})(p) = I_j2(R(p, v_j2)) \text{ with } j_2 = \underset{i \in \{1..n\} \setminus j_1}{\text{argmax}}\ lum(I_i(R(p, v_i))) \tag{6.11}$$

$$\pi_7(I_{1..n})(p) = I_j3(R(p, v_j3)) \text{ with } j_3 = \underset{i \in \{1..n\}}{\text{argmedian}}\ lum(I_i(R(p, v_i))) \tag{6.12}$$

$$\pi_8(I_{1..n})(p) = I_j4(R(p, v_j4)) \text{ with } j_4 = \underset{i \in \{1..n\}}{\text{argmin}}\ lum(I_i(R(p, v_i))) \tag{6.13}$$

Where lum is the luminance of the sampled pixel. π_8 samples the image with the minimum observed value of luminance that is often very close to the diffuse component of the surface. On the contrary π_5 and π_6 pool the two highest observed value based on luminance, providing observations of the highest specular highlight, if any. The variation between π_8 and π_5 already gives insight on whether a material is glossy or not. Finally, π_7 samples the image with the median value of luminance; this has the nice property to discard outliers. When geometry is badly reconstructed or even missing, IBR methods still sample the pixel corresponding to the missing geometry leading to "floaters". The median pooling of π_7 allows to discard floaters on both sides of the luminance spectrum.

For the source mirror images, we use the same reprojection scheme but weights and pooling are based on the input view which provides correspondences.

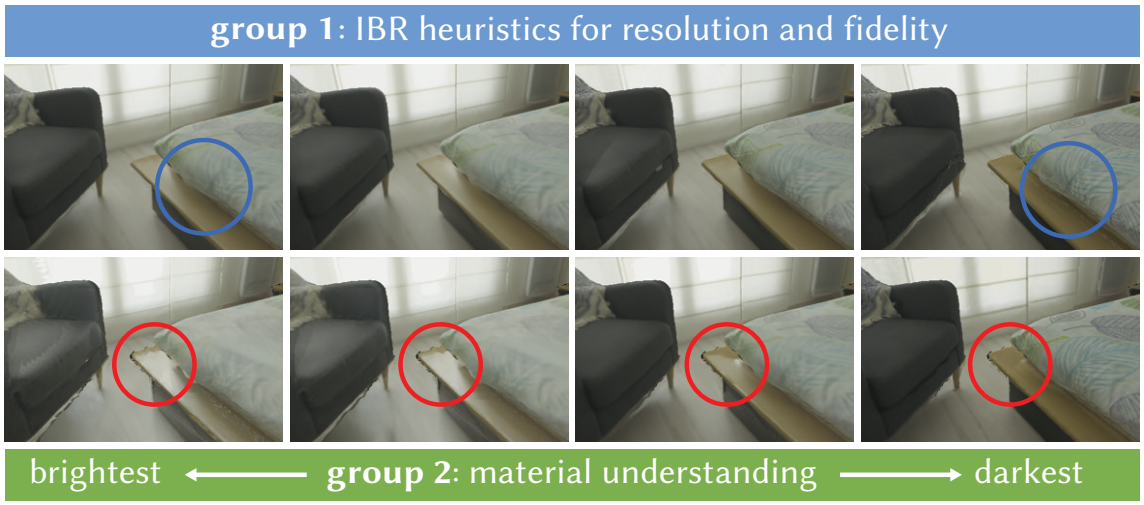


Figure 6.6: We assemble a total of 8 reprojections from the input images, divided in two groups. The first group (top) tries to maximize reprojected resolution, image fidelity and coverage. For instance favoring cameras close to the novel view reduces reprojection errors but favoring cameras close to the observed point provides higher texture quality (blue circles, top). The second group (bottom) gather pixels that span the largest possible range of luminance values to help material understanding. In particular, these reprojections capture the brightness variations due to specular highlights (red circles, bottom).

6.4.6.2 Reprojection of the irradiance maps - Π_B

For the source and added diffuse illumination, which are view-independent, we simply average the reprojections of all the E_i^{src} and E_i^{add} that pass the depth test, weighted by the inverse distance to the corresponding camera:

$$\Pi_B(E_{1..n}) = \frac{1}{W_k} \sum_{i=1}^{n} w_1(v^*, v_i, 3D(p), n(p)) * E_i(R(p, v_i))$$

6.4.6.3 Target mirror image - M^{tgt}

We also dynamically compute a mirror image M^{tgt} under the new lighting. Unlike the source mirror images M_i^{src}, used for material understanding, this target image helps the network synthesize new glossy reflections that correctly account for the target illumination. To compute it, we trace a mirror ray per pixel and sample the image data

from the viewpoint closest to the mirror direction (like in Section 6.4.5), as well as the corresponding diffuse irrandiances E_i^{src} and E_i^{add}.

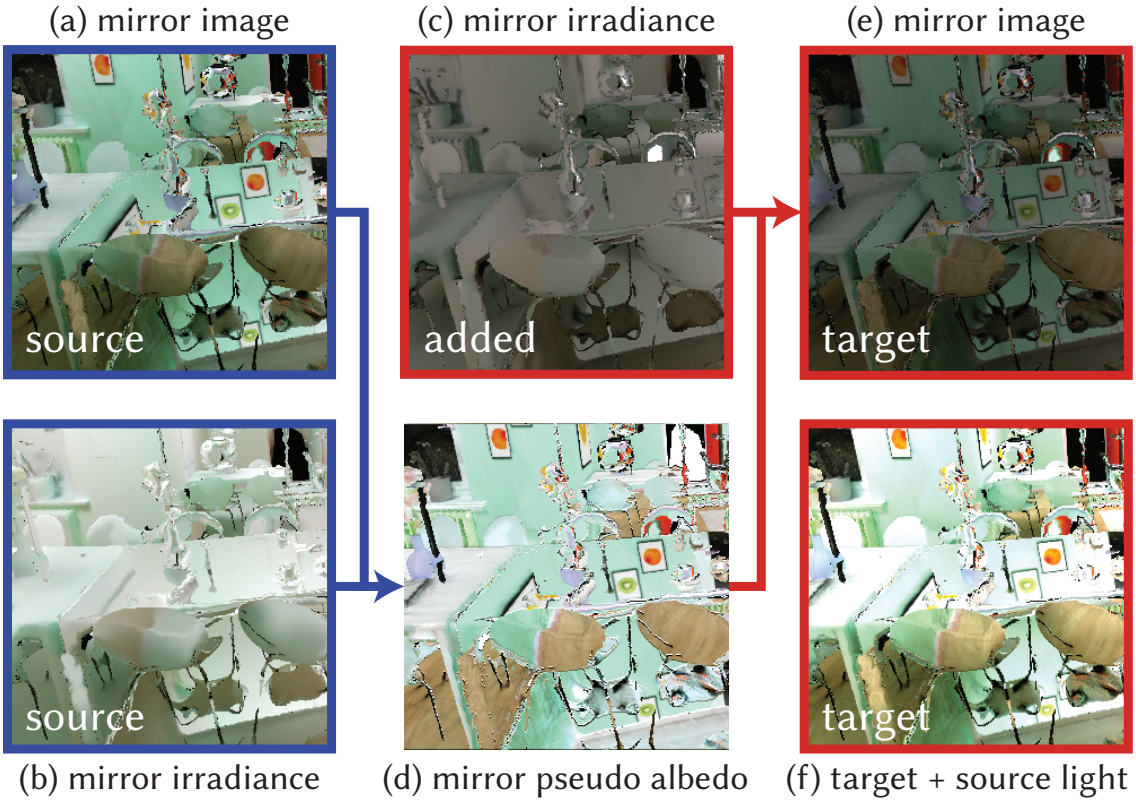


Figure 6.7: To compute target mirror images that properly account for the added light sources, we first divide the mirror input (a) by a mirror image of the source irradiance (b). This gives us a pseudo-albedo (d) that we can multiply with a mirror image of the target irradiance (c) to obtain the final target mirror image (e) corresponding to the newly added illumination. We can also decide to preserve the original lighting in addition to the synthetic target illumination, in which case the target mirror image would account for both source and target illumination (f).

We obtain a pseudo mirror albedo by dividing the RGB value sampled from image I_i with the corresponding value of E_i^{src}, which we multiply with the added target irradiance to obtain the target mirror value. If we keep the original lights on, we also add the RGB value of I_i to this value (see Fig. 6.7). This ray-tracing step runs in real time using Optix [140].

6.4.7 Additional features

We provide a few additional inputs to the network to help it reason about surfaces, discontinuities, and reflections. For each pixel in the novel view we compute the normalized inverse depth (disparity), and the surface normal at the corresponding mesh point. We also provide the incident angle, *i.e.*, the angle between the normal and the view vector (from input camera to visible scene point). This angle map helps the network deal with Fresnel effects.

Finally we provide the ratio of distances between the observed and the reflected points and the observed points and the camera center. This guides the network in smoothing the reflection for rough surfaces. These features are illustrated in Fig 6.8.

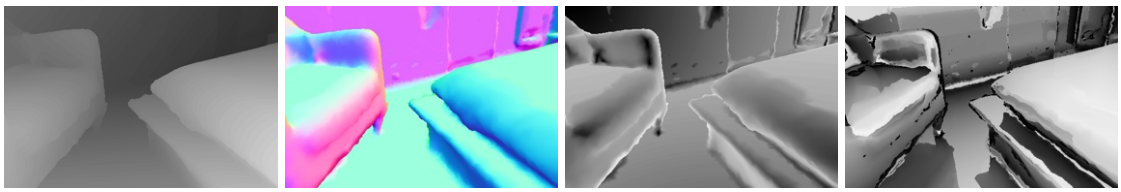


Figure 6.8: From left to right: The disparity map to reason about geometry, depth and occlusions. The normals. The angle map to guide Fresnel effects. And the reflection ratio to guide smoothing due to roughness.

6.4.8 Implementation Details

We first simplify the mesh reconstructed with MVS and apply Laplacian smoothing on the normals. We also indentify planes by running RANSAC on the mesh vertices. We snap to plane normals for inlier points with a normal close to that of the plane; this improves overall robustness of our approach.

Algorithm 1 Pre-processing and online rendering pipelines

inputs
n images captured from multiple viewpoints $I_1, \ldots, I_n$
procedure PREPROCESSSCENE ▷ off-line pre-computation
Compute 3D mesh from multi-view images
for i=$1, \ldots, n$ **do**
Compute source irradiance maps E_i^{src} (§ 6.4.1)
Color mesh with approximate albedo Ii/E_i^{src} (§ 6.4.2)
Compute source mirror images M_i^{src} (§ 6.4.5)
end for
Compute albedo-colored 3D mesh using the approximate albedos
end procedure
procedure PREPROCESSTARGETLIGHTING ▷ fast light estimate
for i=$1, \ldots, n$ **do**
Compute target added irradiance maps E_i^{add} (§ 6.4.3)
end for
end procedure
procedure RENDER ▷ interactive novel view rendering
Compute the reprojections of images and source mirror buffers $\Pi_I(I1..n)$ and $\Pi_I(M_{i..n}^{src})$ (§ 6.4.6.1)
Compute the reprojections of source and target irradiances $\Pi_B(E_i^{src})$ and $\Pi_B(E_i^{add})$ (§ 6.4.6.2)
Compute target mirror image M^{tgt} (§ 6.4.6.3)
Compute additional features (§ 6.4.7)
Forward pass with relighting network (§ 6.5.1)
end procedure

6.5 Network and Training

From the input features described in Section 6.4, we render a novel view with novel illumination using a convolutional network, which we train on synthetic data (§ 6.5.2).

6.5.1 Network architecture

Our network uses a multi-scale architecture shown in Figure 6.9. A 4-block ResNet [66] processes the input. Its output is then decomposed into a 5-level feature pyramid, using convolution layers with kernel size 4 and stride 2 as a downsampling operator. Each pyramid level is processed by an independent 4-block ResNet. The outputs of the 5

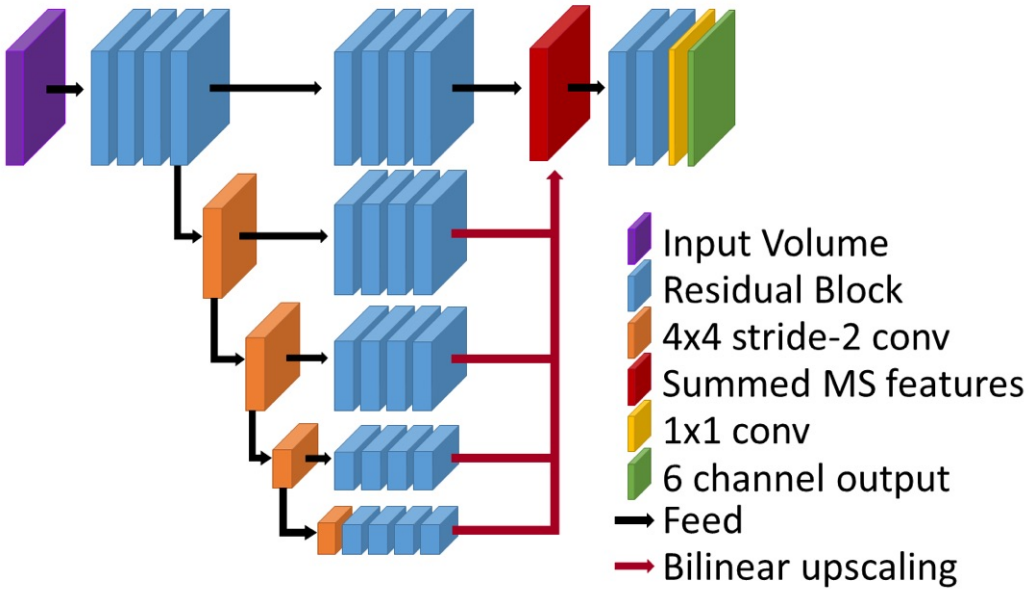


Figure 6.9: Our network uses a multi-scale architecture to process the input features maps. It produces two output images, for the diffuse and specular image component respectively.

pyramid levels are bilinearly upsampled back to the original resolution and summed. A final 2-block ResNet, operating at full resolution, processes the summed features to produce two outputs: one containing the diffuse image component, and one containing the view-dependent specular effects.

We use fixup ResNets [207], with 6 residual blocks and 66 filters per convolutional layer, with kernel size 3, zero padding and ReLU activations. The input feature maps make up 66 channels as well. We found the "fixup" initialization to significantly improve convergence speed and output quality. The diffuse and specular components are supervised separately, then summed to obtain the final relit image.

Input/output tonemapping. To ensure the distribution of the values is not two skewed toward darker pixels — which we found is detrimental to network convergence — we apply a tonemapping operator to all the radiance inputs to the network (reprojected source images and reprojected mirror images). We use the tonemapping operator

proposed by Kalantari and Ramamoorthi [93] and defined as:

$$x \mapsto \frac{\log(1+\mu x)}{\log(1+\mu)},\ \mu = 64. \tag{6.14}$$

The diffuse and specular outputs of the relighting sub-network are also in the tonemapped space. To get the final output composite we just invert the tone-mapping operator for each component and them together giving a linear-space composite image.

Loss functions. We supervise the overall network to minimize the sum of three losses: one for data fidelity, one for multi-view consistency of the diffuse component and an adversarial loss that improves the overall texture sharpness and plausibility of the output. The data fidelity loss is:

$$\mathcal{L}_{vdep} + \mathcal{L}_{diff} + \mathcal{L}_{im}. \tag{6.15}$$

Each individual loss is computed in the tonemapped domain defined by Equation (6.14), and is itself the sum of a standard L_1 loss and a so-called perceptual loss based on the L_1 distance between VGG activations [90]. We use the VGG features after the first three maxpooling operations weighted by 1.0, 0.5 and 0.1 respectively to give more emphasis to low-level features. We found this helps recover sharper outputs than using the L_1 loss alone. It also speeds up the network convergence. The VGG loss has weight 0.1, relative to the plain L_1 loss.

The multi-view loss is computed using ground truth optical flow to reproject pairs of closeby views on each other. It is the $L1$ difference between the reprojections. This also enforces that the diffuse component does not contain residual view dependent effects and is temporally stable.

Finally we use a adversarial loss on the 6 channel output, the discriminator is a global GAN that implements the Relativistic Gan loss introduced by Jolicoeur-Martineau [91] and the same discriminator architecture [150].

6.5.2 Synthetic training data

Training data for relighting and view synthesis of complete indoors scenes needs to have sufficient variety of content (geometry, materials, lighting) and allow easy generation of different lighting conditions. Following the same scheme as in Chapter 4, we use

synthetic data, with the similar constraint that we need our training to transfer to the real-world data we will capture.

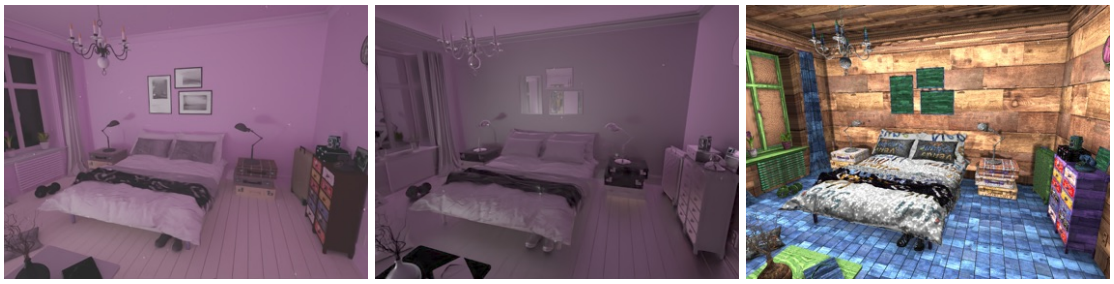
We have developed a comprehensive system to allow generation of synthetic training data from artist-modelled scenes, either purchased or freely available [14] [1]. For the former, our system exports V-Ray or 3DS Max materials to a format suitable for Mitsuba [88], by evaluating the shade trees in 3DS Max and exporting spatially-varying BRDF maps, while the latter are already in Mitsuba format. Our interactive interface then allows us to easily place cameras and lights to prepare each scene for training. For training, we used a total of around 2000 viewpoints, each with 6-7 different lighting conditions with lights placed manually. These are stored separately for diffuse and glossy in HDR format, and are then augmented during training in arbitrary combinations of exposures.

Using the Heckbert notation [68] we separate the types of paths into "diffuse", *i.e.*, $L(S|D)^*DE$ and "glossy", *i.e.*, $L(S|D)^*SE$, where L are the lights, S and D are glossy and diffuse materials respectively and E is the eye or camera. These two components are stored separately, and correspond to the integrals $a(\mathbf{x})E(\mathbf{x})$ and $S(\mathbf{x}, \boldsymbol{\omega}_o)$ respectively (see Sec. 6.3.1). The corresponding images are rendered with bi-directional path-tracing at 256 samples per pixel, which is efficient for the type of indoor scenes we treat using our modified version of Mitsuba. Images are stored in linear 32-bit floating point format. The two rendered and denoised components are shown in Fig. 6.10.

For each hand-modeled training scene with exact geometry, we also compute rendered images that we use to reconstruct an MVS proxy used for all rendering during training, similar to Chapter 4. This encourages the network to learn to correct the rendering errors due to imprecise geometry, since it compares its rerendering against the images rendered with exact geometry. However, generating such data for indoor scenes is much more challenging than for outdoor scenes for example, since indoor environments have many large textureless surfaces such as walls, ceiling etc, and many objects with highly reflective materials that make MVS algorithms fail.

To overcome the problems caused by lack of texture our system automatically generates alternate representations of the scenes by adding texture on large surfaces with uniform appearance. These scene variations are only used during MVS reconstruction and their

[1] We use the scenes from evermotion.org Archinteriors 30: 001-010, Archinteriors 01:, 004 and 005 and the scenes Country Kitchen and the White Room from [14]. We modified the scenes to add more color on the walls and furniture.



(a) synthetic rendering (diffuse)

(b) glossy component (brigthened for display)

(c) automatic scene modification for MVS reconstruction

Figure 6.10: We decompose our synthetic target renderings into diffuse (a) and glossy components (b). To better simulate real data, the network inputs used at training time are computed using an MVS reconstruction of the scene instead of the ground truth geometry. We facilitate the reconstruction using an automated tool that changes the lighting and adds markings and textures (c).

modified appereance is not used in any way by our rendering pipeline. We use both "natural textures" and markings that help MVS find features. We show the modified version of a training scene in Fig. 6.10.

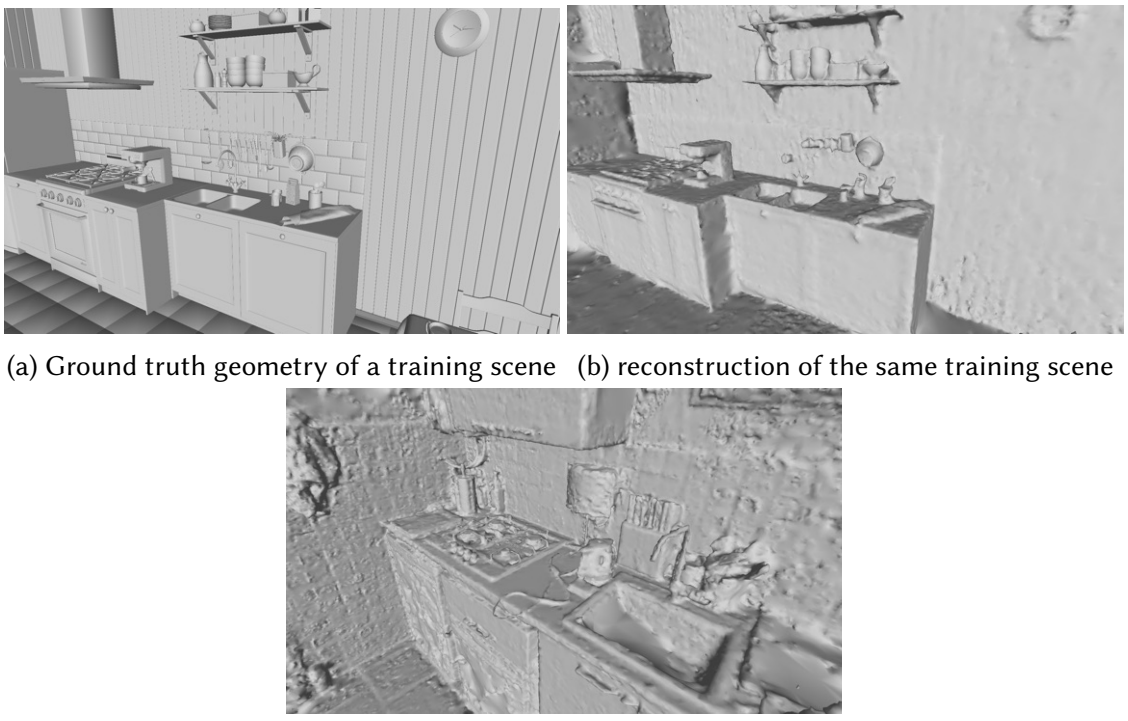
To overcome the problem of materials, we render the scene as completely diffuse with simple diffuse lighting that allows the MVS algorithms to work well. The resulting reconstructions (see Fig. 6.12) have similar artifacts to those of captured real scenes, allowing the network to learn the desired domain transfer.

To limit the storage usage, we save the images as lossless 16-bit PNG files. This still provides enough bit-depth to preserve the data's dynamic range (especially for the glossy illumination). For better bit allocation, we tonemap the images before quantizing to 16-bits with the map $x \mapsto (x/(x+1))^{\gamma}$, where $\gamma = 0.4$ is the gamma correction [153]. All our lights have emissive intensity normalized to 1.0.

6.5.3 Training details

We train our network and discriminator using the RMSProp algorithm with default parameters. We use a learning rate of 10^{-4} with a batch size of 1 for 600k iterations, which takes approximately 96 hours on a RTX 6000 GPU.

Figure 6.11: A selection of the synthetic scenes we use for training.



(a) Ground truth geometry of a training scene (b) reconstruction of the same training scene

(c) reconstruction of a real scene

Figure 6.12: To minimize the impact of the domain gap between real data and our synthetic training scenes, we compute the network inputs on a degraded mesh at train time, obtained by multi-view stereo. The training reconstructions (b) exhibit artifacts that are qualitatively very similar to what we expect to see in real scenes (c). This helps the network generalize well to real scenes.

Figure 6.13: A selection of view-synthesis and relighting outputs produced by our neural algorithm. For each row, we show one input view (a), a newly synthesized view (b) and two relightings, also at a novel viewpoint (c) and (d). The scenes are respectively: Bedroom2, Sofa, Bedroom1, LivingRoom and Hall.

6.6 Implementation, Results and Experiments

We captured multi-view datasets for qualitative analysis on real-world scenes (§ 6.6.1). To the best of our knowledge, no previous method can jointly relight scenes and enable free-viewpoint rendering for indoor scenes with glossy materials. We thus compare

to view-synthesis (§ 6.6.3) and relighting (§ 6.6.2) techniques separately. We run our quantitative comparisons and model ablations (§ 6.6.4) on a held-out synthetic scene, shown in Figure 6.18. Figure 4.1 and 6.13 shows our model's output for relighting and view-synthesis on five real scenes. Our results are best appreciated in the supplemental video. As can be seen in the figures and videos, our method allows plausible relighting of scenes with complex, glossy materials, while allowing free-viewpoint navigation. Our method provides faithful flow of highlights when moving the camera, both for the original (real) light sources, and those added for relighting.

6.6.1 Real-world scene capture

We created a dataset of real test scenes, using a Canon 50D DSLR and a 18-50mm lens, photographing in 14-bit RAW format from 250–350 viewpoints per scene. Specifically, we captured the following scenes, shown in Fig. 6.1, 6.14-6.13: Bedroom1 (352 photos), Bedroom2 (306), LivingRoom (321), Sofa (282), Kitchen (274), Hall (291).

Our irradiance estimation requires that the real light sources be directly visible in at least some of the photos. The linearity and higher dynamic range of the RAW format helps us estimate relative light powers. Despite the higher bit-depth, directly visible light sources can still be clipped. In this case, the user needs to provide a few clicks to allow estimation of the clipped source irradiance; we describe this procedure in Appendix B.1. We manually place additional lights in the reconstructed 3D scene when relighting.

6.6.2 Relighting

To our knowledge, there is no previous method that can relight multi-view datasets of entire scenes. Instead in Figure 6.14 we compare to a baseline that uses the 3D information available to our method, and the state of the art inverse rendering technique for indoor scenes from Li et al. [121]. Their method does not directly relight images; it produces an albedo from a single image. Our baseline assumes a Lambertian material model: it combines their albedo with our target irradiance (computed from the MVS mesh) to get the relit image. Note that we provide the benefit of irradiance computed using 3D information to allow a comparison that is as fair as possible. Compared to this baseline, our relighting solution preserves texture in the scene, and benefits from the

overall quality of our neural rendering.

Input view w/ original lighting | Our relighting result | Baseline relighting result

Figure 6.14: Relighting comparison to a baseline using the method of Li et al. [121]. Left: Input view with original lighting. Middle: Our relighting result. Right: Baseline result. For the baseline computation, we first extract albedo using the approach of Li et al. [121] and directly multiply it with our target added irradiance to form a relit image. As we can see in the zoomed region, contrary to the baseline, our method correctly modifies the lighting under the stool, removes the shadows initially present and does not exhibit geometric artifacts while preserving high texture quality.

We also make a comparison to Wu and Saito [200], which is indicative of modern single-image methods. We asked the authors to insert lights in the scene to produce an effect

similar to our lighting modifications, Figure 6.15. Their approach only uses a single image and has a very approximate notion of 3D geometry compared to the much richer input we use. Compared to ours, their method cannot cast shadows for complex objects such as the armchair. It creates a static highlight for the novel light on the hardwood floor, without the view-dependent effects we can synthesize.

6.6.3 View-synthesis

We compare our model to four free-viewpoint rendering methods: a baseline Unstructured Lumigraph Rendering (ULR) implementation with per-pixel blending weights [19], Deep-Blending [72] already partially presented in Chapter 5, as well as the very recent Neural Radiance Fields — NeRF [132] and Free View Synthesis (FVS) [154]. We used the original Deep-Blending network and code and the authors' implementation of NeRF. The FVS comparisons were run by the authors using the model they trained on the Tanks and Temples dataset [100], which has very different scene content compared to our scenes.

Figures 6.16,6.17 show we can synthesize realistic highlights that move coherently when changing viewpoints while previous methods cannot (see also supplemental videos). Since ULR, Deep Blending and FVS blend multiple frames and have no or limited material understanding, they tend to create double ghost highlights that appear to "jump" from one frame to the next (see video), or even completely suppress glossy reflections. ULR and our method use the same MVS mesh; this shows our neural rendering is largely robust to errors in the 3D reconstruction.

Deep Blending [72] uses our more complex *per-view meshes* described in Chapter 5 that significantly improve the geometry, especially for small objects. As a result, Deep Blending's rendering of small objects is sometimes more detailed than ours (e.g., the legs of the chair in Figure 6.17, first row). Still, our renderings are much sharper and realistic overall. Our method would benefit from per-view meshes or any other improvements to the MVS geometry, but this is largely orthogonal to our view-synthesis and relighting contributions.

As noted by Riegler and Koltun [154], NeRF struggles with the unstructured capture required to reconstruct large scenes such as ours. For fair comparison, we made a best-effort attempt to improve NeRF's output by manually selecting a subset of images that produces the best visual and numerical quality for NeRF. We found much fewer,

Figure 6.15: **Single-image relighting comparison.** We provide an indicative comparison with the *single image* method Wu and Saito [200] that has much less information that our multi-view input. We see that this method cannot reproduce shadows of complex objects such as the armchair, nor the view-dependent glossy effects on the hardwood floor.

more tightly packed images, provide better visual quality *locally*, but they result in low coverage, and therefore worsen the *overall* image quality. NeRF can synthesize convincing moving highlights (see supplemental), but the overall image quality is low.

FVS [154] gives good static results, but as with all previous view-synthesis techniques, does not render moving highlights correctly. They have more visual artifacts and less temporal coherence (see supplemental videos). This is possibly accentuated by the difference in content compared to their training dataset.

Figure 6.16: **View-synthesis comparison.** In the LivingRoom and Sofa scenes, we see that plausible glossy highlights are generated by our method (a) (see also supplemental video for moving highlights). In contrast, ULR (b) and Deep Blending (c) do not faithfully reproduce glossy highlights.

Table 6.1 summarizes our numerical evaluation on a held-out synthetic scene; Figure 6.18 and supplemental show the renderings. Appendix B.2 analyzes this error as a function of the distance to input views.

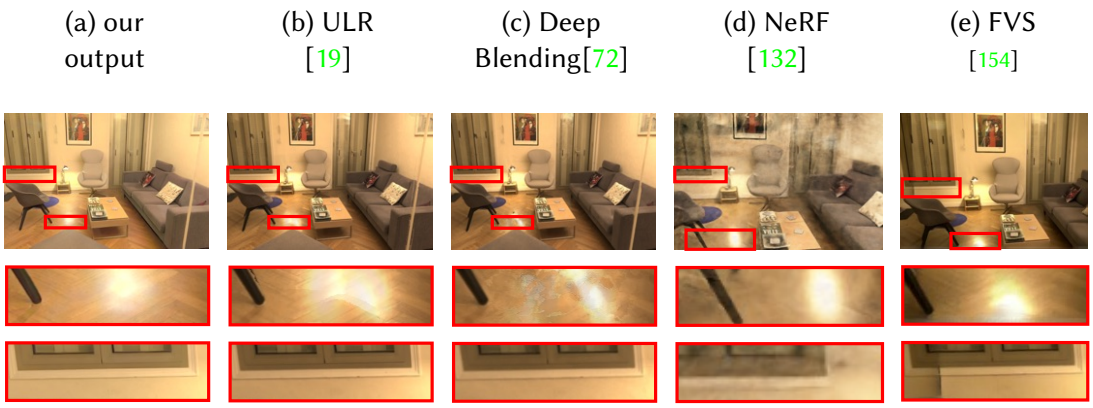


Figure 6.17: **View-synthesis comparison.** Our model (a) synthesizes plausible glossy highlights (see supplemental video to better appreciate the effect for moving highlights) whereas ULR (b) and Deep Blending (c) exhibit artifacts like double highlights, cross-fading and missing highlights. NeRF (d) can synthesize plausible highlights, but generally yields much lower quality on large-baseline indoor scenes. FVS (e) suffers from temporal coherence issues and exhibits strong artifacts.

Table 6.1: **View-synthesis evaluation.** Our method consistently outperforms previous work on a held-out synthetic scene.

method	PSNR ↑	SSIM ↑
Ours	**29.25**	**0.916**
Hedman et al. [72]	27.07	0.911
Buehler et al. [19]	24.50	0.896
Riegler and Koltun [154]	21.66	0.853

6.6.4 Model ablations

We conduct an ablation study to isolate and highlight the importance of our input features and design choices. Every model variant in this section is trained from scratch using the same training procedure. Numerical results are summarized in Table 6.2.

6.6.4.1 Single reprojection

Our reprojection scheme (§ 6.4.6) produces 8 composites that are key to learning a good implicit representation of scene materials. If we replace it with a single reprojection using w_1, the network is much less effective at separating specular and diffuse effects

Table 6.2: **Ablations.** Quantitative comparisons on a held-out synthetic scene shows that each element in our design contributes significantly to the final result quality.

method	PSNR ↑	SSIM ↑
ours	**26.12**	**0.909**
no GAN	20.97	0.876
no GAN nor VGG	25.06	0.895
no target mirror	25.10	0.897
no source mirror	25.06	0.891
single reprojection	23.52	0.874
no extra features	23.85	0.889
no multiscale in network	24.49	0.882
no split diffuse/Vdep	24.95	0.904
GT geometry	24.55	0.881
fewer training scenes	25.33	0.897

and copies incorrect glossy highlights baked in the reprojected input views, significantly reducing realism (Fig. 6.19).

6.6.4.2 No target mirror image

The target mirror image (§ 6.4.6.3) guides the synthesis of new glossy reflections. Without it, predicting accurate specular highlights, for both input and added light sources, is extremely difficult. This ambiguity leads to blurring (Figure 6.20).

6.6.4.3 No separate supervision on diffuse/specular (*no split diffuse/Vdep* in table)

Supervising the diffuse and specular components separately leads to cleaner glossy reflections. If we online supervised the summed output, the network tends to focus on diffuse regions, which leads to unrealistic, muted reflections (Fig. 6.21).

6.6.4.4 No Multiscale module in the network

The Multi-scale module in our network is essential to obtain realistic specularities on rough surfaces. It yields a wider receptive field and can overcome the limitations of our

mirror-like surface hypothesis, as shown in Fig. 6.22.

The remaining ablations in Table 6.2 show: the effect of removing the discriminator loss (*no GAN*), discriminator and VGG (*no GAN no VGG*), the source mirror layer and the use of extra features. We also show the effect of only using ground truth (GT) geometry, i.e., no MVS reconstructions, and training using 8 instead of 16 scenes used for the full method.

6.6.5 Performance

We use RealityCapture [152] for the MVS reconstruction, which takes about 10 minutes per scene. Precomputing the source irradiance takes around 3 sec./image on average on an Intel Xeon E52650 @ 2.2Ghz, while the target added irradiance takes about 2 sec./image using Mitsuba and 16 samples per pixel on the same machine. Both images are denoised using Optix. So, for a typical scene (300 input frames), it takes about 10 minutes to precompute the added irradiance for a new lighting condition. Given this precomputation, our renderer is interactive and runs at 5–8 frames per second on a NVIDIA GeForce 2080RTX graphics card, rendering at 512×384 resolution. Using texture compression, we can render up to 1024×768 images, maintaining the framerate above 2fps.

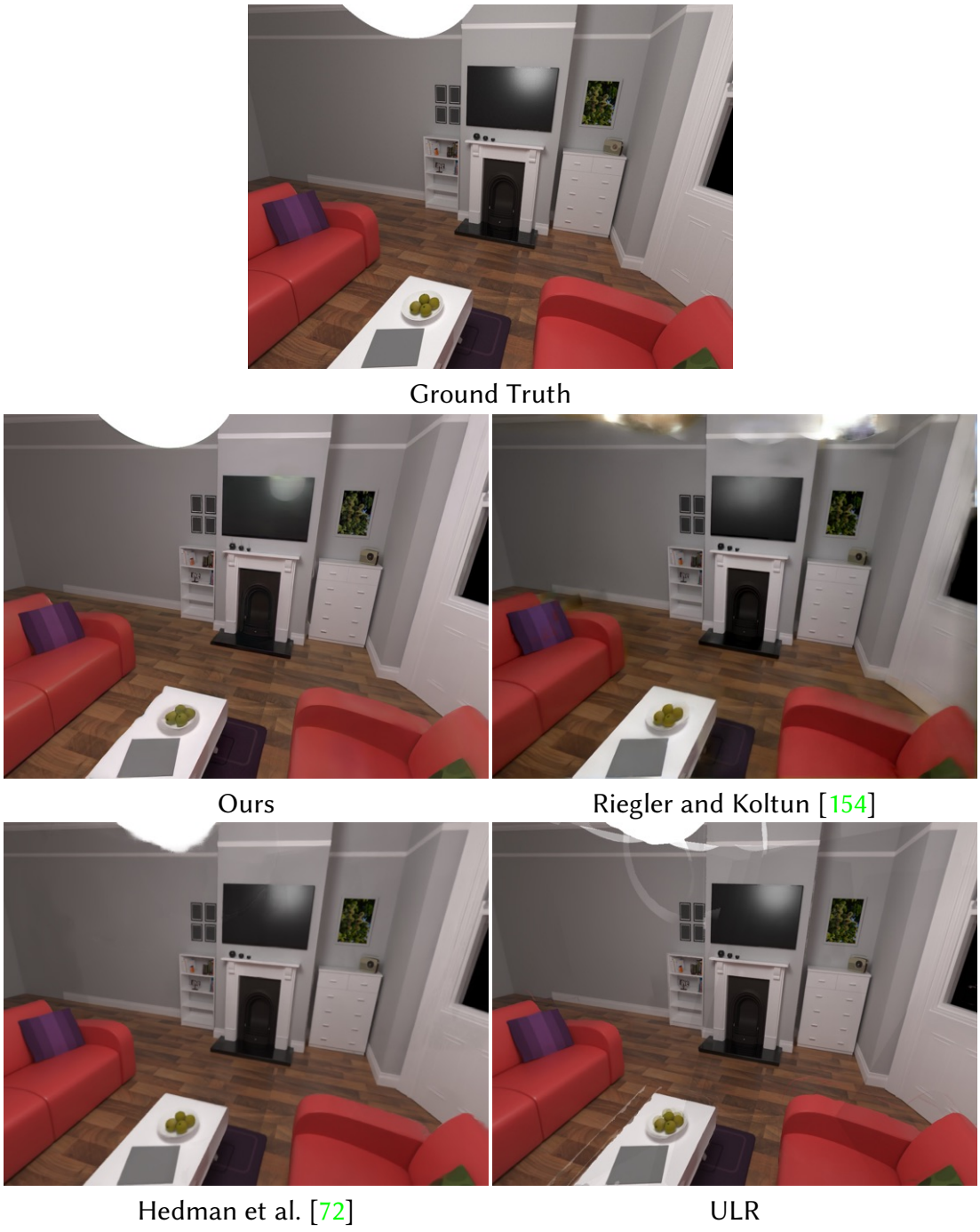
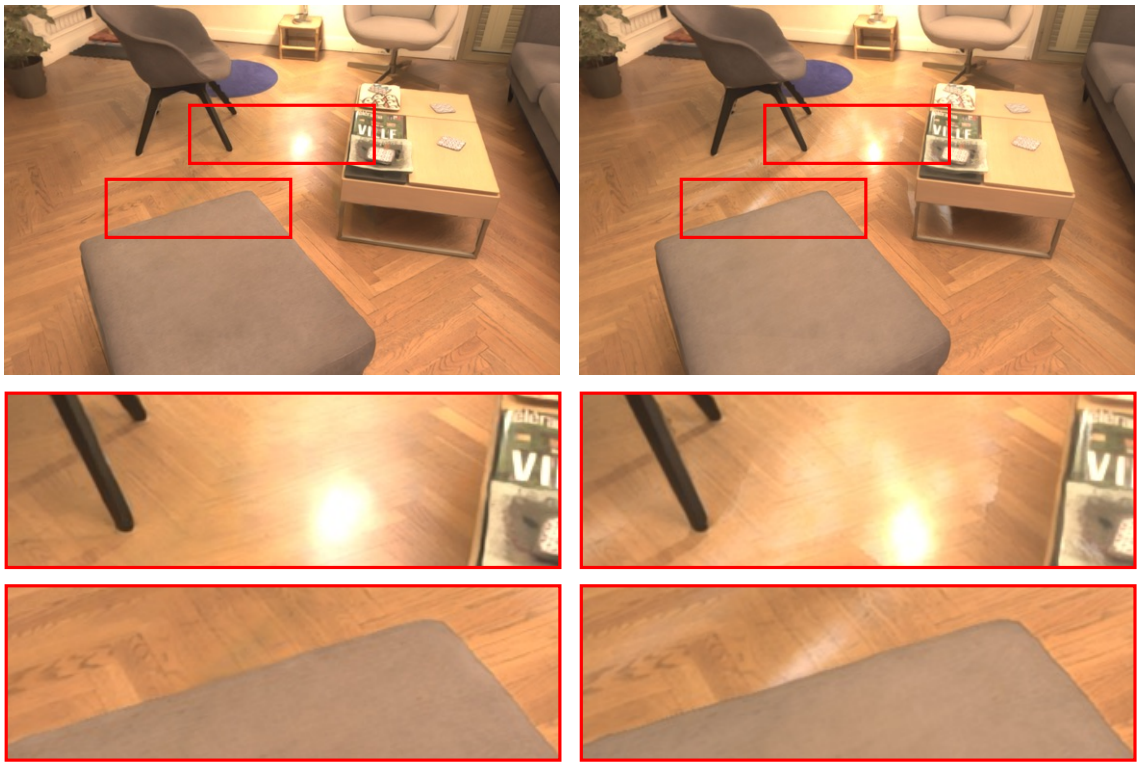
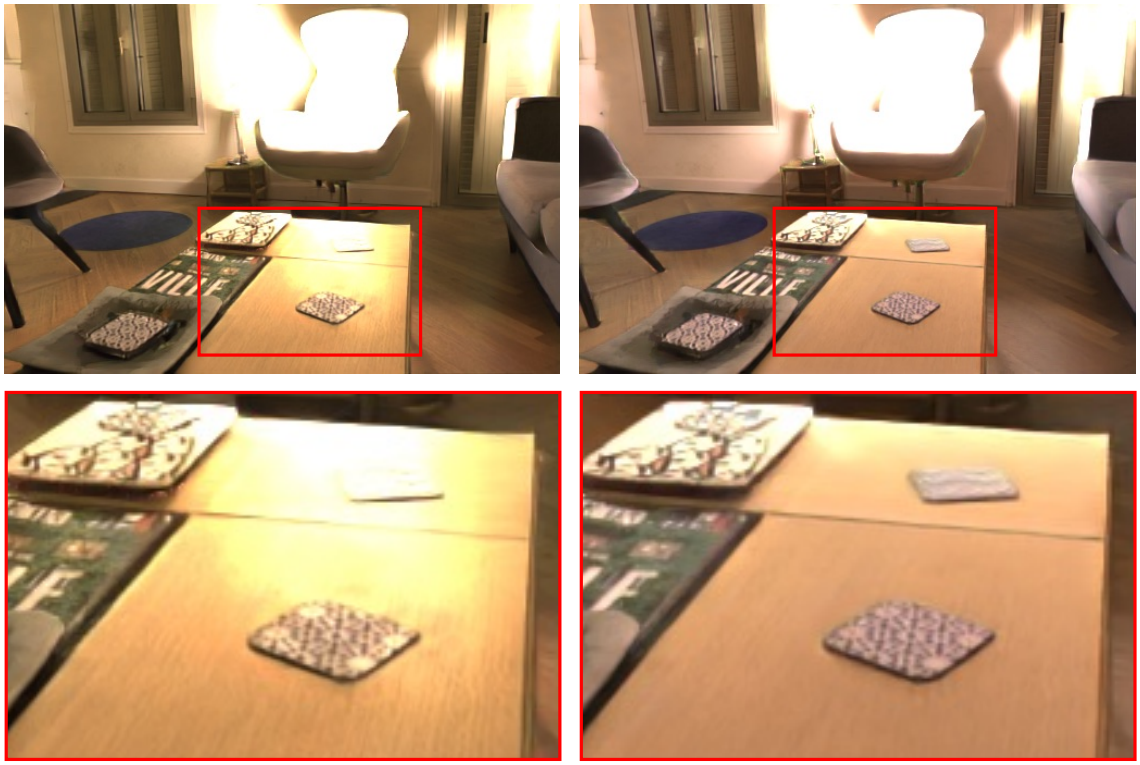


Figure 6.18: **Held-out synthetic scene used for quantitative evaluation.** We show here the synthetic scene used for numerical evaluation. In rows 2 and 3, we show the synthesized result for each method; For all methods but ours, the highlight on the TV does not move correctly (see also supplemental).



(a) our full solution (8 reprojections) (b) single reprojection

Figure 6.19: **Single reprojection ablation.** With only one reprojection heuristic, using the closest views, the model cannot analyze scene materials properly (b) and outputs wrong colors corresponding to glossy highlights observed in the input views. In our full pipeline (a), the reflection is correct and there is no residual error from the reprojection.



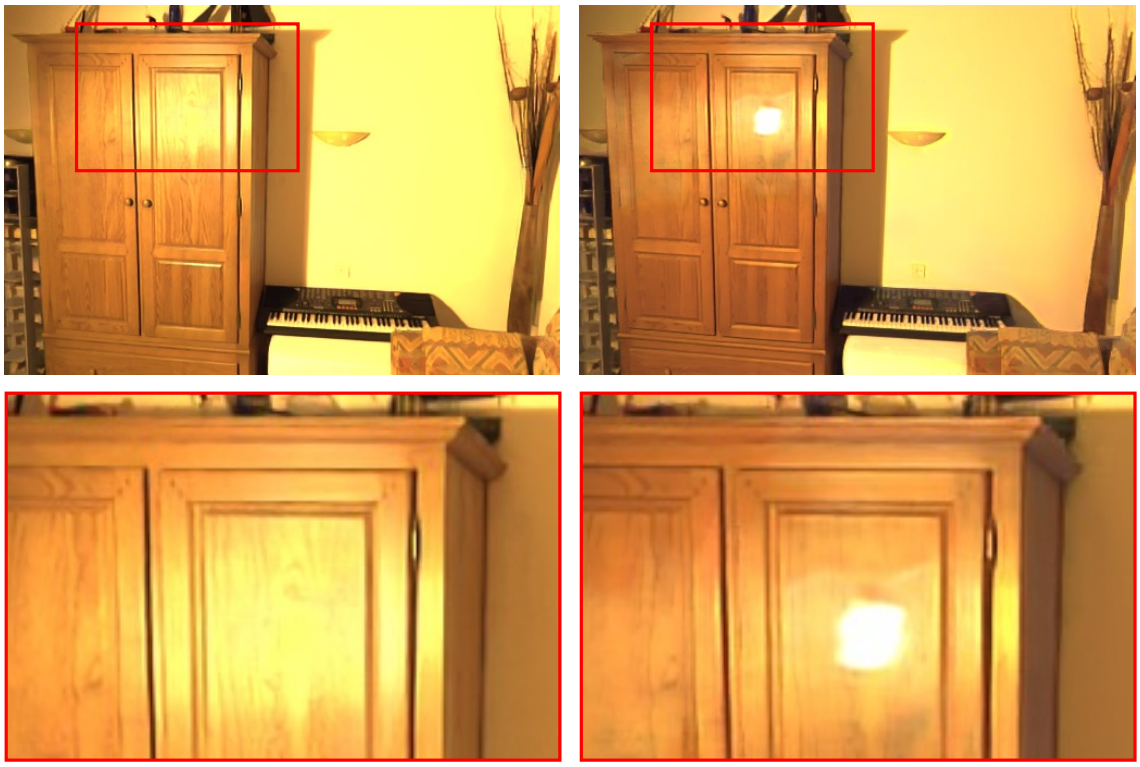
(a) our full solution (b) without target mirror

Figure 6.20: **Target mirror ablation.** In our full pipeline (a), the target mirror image guides the reconstruction of realistic glossy reflections for existing and new light sources. Removing the target mirror (b) leads to lackluster images with no specular highlights.

(a) our output, with diffuse/specular separation

(b) without diffuse/specular separation

Figure 6.21: **Ablation diffuse/specular supervision.** When separating the diffuse and specular output components, the loss can be better balanced, using scalar weights, to ensure correct highlights are synthesized (a). Without the split, the highlights are lost (b).



(a) our full solution (b) without the multi-scale module

Figure 6.22: **Multi-scale module ablation.** Our multi-scale module provides a wide receptive field that is particularly important to analyze complex materials with a large spatial patterns (a). Without it, the network struggles with rough surfaces for instance, and generates pure-mirror reflections (b) instead of spreading them out properly.

6.7 Limitations and Future Work

Limitations. Like most neural rendering approaches, our method suffers from a few residual visual artifacts. Most of them are due to incorrect geometry where the MVS reconstruction failed. This is visible in some sequences in the video especially when a novel light is tangent to a surface. Both the reprojection and Monte Carlo interaction can cause some temporal flickering, but this is highly reduced by the use of the multi-view loss. Currently we can only turn off *all* the lights in a scene instead of individual light sources. Finally, our simplified illumination model is not valid for anisotropic materials; in this case we expect our model to fall back to a standard blending behavior.

Future Work. In future work it would be interesting to test and train on more scenes for higher generalization. We did not train nor test on outdoor scenes, the main issue of this kind of data being the very high dynamic range that might break the stability of irradiance estimations. It would be interesting to evaluate the impact of such instability. Another interesting direction is to try and generalize our approach to enable more edition capabilities directly such as object insertion or material modification. One of the main issue for such tasks is obtaining training data. Training a model to do several tasks at the same time would require a combination of geometry, materials and lighting variations of scenes which makes the required amount of data grows in a combinatorial manner.

6.8 Conclusion

In this Chapter, we presented the first method that permits relighting and free-viewpoint navigation in real indoor scenes with glossy materials. Our algorithm combines the strengths of inverse, physically-based and image-based rendering methods. It uses multi-view inputs, a 3D mesh reconstructed by MVS and efficient approximations of physically-based rendering to compute input feature buffers from which a convolutional network learns an implicit representation of materials and lighting in the scene. The source buffers include source diffuse illumination, mirror images from the input views, and approximate albedo. These enable us to compute the target illumination buffer by path-tracing. The neural network uses this information to relight and re-render the scene. It runs at interactive frame-rates, thus enabling free-viewpoint navigation. We train

using synthetic scenes, that in addition to their exact geometry are also reconstructed with MVS, allowing the CNN to transfer to real-world images. Our results show realistic relighting of complete indoor scenes, including plausible apparent motion of glossy reflections. This opens up an extremely promising avenue for future research, paving the way for more general solutions that could apply to all scenes (including outdoors), overcoming remaining limitations, *e.g.*, related to erroneous 3D reconstruction. This method shows that IBR methods and multi-view editing methods can be merged when using good representations, and opens the way for more general solutions that integrate the flexibility of traditional computer graphics and the ease of asset capture provided by IBR approaches.

Chapter 7

Conclusion

7.1 Lessons Learned and Contributions

In this thesis, we presented several methods that all share the same high-level goal: improving the quality and flexibility of image-based rendering and relighting methods. The different projects also share methodological elements.

First, given multi-view data they try and extract as much information as possible from the input to solve the specific problem they try to tackle. An example is the use of available texture information in several views for inpainting (Chapter 3). Another example are RGB shadow images that sample input colors to give more meaningful content to a network (Chapter 4); the same idea was extended to compute input irradiance and mirror images later on (Chapter 6). A final illustration of this idea is the merging of globally consistent geometry and noisy depth maps with input image consistency to form accurate per-view meshes (Chapter 5).

Second, all methods are designed to take into account the noise and errors in geometry estimation. This was done explicitly when designing per-view meshes for a novel image-based rendering method. In the case of very inaccurate geometry, we introduce a workaround through object removal. For instance, on semi-transparent highly specular objects such as cars, removing the object through multi-view inpainting avoids artifacts caused by geometry. Another way to achieve this goal is through the use of meaningful input to a powerful neural network: this is the case of the RGB shadow images that help to overcome errors due to "over-reconstruction". Finally we take this uncertainty in the input data into consideration with the use of a dual representation for synthetic scenes during training for relighting.

We can also identify some common findings between some of the projects. First, the three projects using deep learning showed that by providing good heuristics and guidance to networks, one can achieve good generalization without needing a lot of data. Whether

it is with real data supervision in the deep blending project or with synthetic data in both relighting projects, the total number of training scenes is less than twenty and the networks still transfer correctly to real, unseen, test data. This is a crucial point for the kind of tasks we treat as acquiring data to train models is by itself a very time-consuming task. Next, most of the difficulties we overcome with our methods are due to errors in estimation that are done before our method can be used, for instance, bad geometry reconstruction. This means that if these estimation methods improve, the results of the proposed solutions should improve too.

Contribution to lighting editing. Image-based rendering techniques are good at displaying new viewpoints of a given scene. But the content is rendered with the lighting condition under which it was captured. We showed that using a geometry proxy and image-space features along with a neural network it is possible to realistically relight captured scenes.
First, we worked with outdoor scenes and simplified lighting conditions. We used synthetic data to train a neural network to change the sun position and the global tone of images. The results outperformed the state of the art for multi-view relighting. We also showed that our methods could be used with a single photo if a proxy geometry can be recovered by other means. This allows users to do single-view relighting of landmarks by using internet photo for reconstruction.
Next, we used our experience with neural relighting to design a method that can alter the lighting in indoor scenes. We can simulate second bounce effects, add multiple light sources, and even turn-off the input lighting. We used irradiance maps, a more general representation of lighting, to achieve these results. Our irradiance maps are based on a physically-based rendering of the proxy geometry using approximate albedo. As they do not use the albedo for the final integration of the radiance but only for global illumination they are quite resilient to errors in the approximation.
Both these methods allow capturing a real scene under a certain lighting condition and rendering it under novel ones, providing much greater flexibility to users compared to previous solutions.

Contibution to geometry editing. We addressed two main geometric types of edition in this thesis. First, we introduced an object removal method for multi-view data. It

can be used to remove content from a scene in a realistic multi-view coherent manner, for pure scene manipulation, but also to remove objects that could impair the visual quality of image-based methods. This method was designed for man-made environments where the piecewise planar model used can be applied, but is the first to enforce multi-view coherency and to respect perspective. Second, we showed that our pre-trained outdoor relighting network can be adapted for captured scenes composition. One of the main issues when compositing two captured scenes is the fact that the lighting may be quite different, for instance, the light orientation may vary leading to unrealistic compositing. Using the network to coherently align lighting, shadows, and interactions between the scenes led to the first method that can compose captured content automatically.

Contribution to rendering. While IBR methods give decent results on many scenes they still struggle with some geometry artifacts. We introduced a method to improve per-view mesh quality for IBR. We demonstrated that careful treatment of refined depthmaps can have a very positive impact on the rendering quality of novel viewpoints (Chapter 5). Finally, we contributed to view-dependent effect rendering in the context of multi-view data. Through the use of a mirror image and by providing observations of material behavior, we train a network to compute novel relit viewpoints with synthesized view-dependent effects, such as specular highlights or fresnel effects (Chapter 6).

7.2 Potential research directions

In this thesis, we presented methods to augment the flexibility of image-based methods. While we advance on several fronts a lot of effort has to be put in research in order to obtain artifact-free results, to reduce the number of images required or to augment performance. These general directions are all open to further research; some solutions some solutions may arise from hardware and engineering improvements while others require rethinking data representation and require more involved research. For instance, performance will certainly benefit from better hardware, advances in machine learning, and in neural network architectures. On the other hand to reduce the number of images either MVS methods have to drastically improve or one would have to change the representation they use for geometry, possibly with a more implicit approach, creating new challenges for instance with light transport simulation.

Material editing. One of the three main elements described at the beginning of Chapter 2 are materials. In this thesis, we do not address the complex problem of material editing. While early solutions were provided by work on intrinsic images, being able to change a texture and correctly impact the scene or make materials more specular is still an important issue as it impacts light transport with complex non-local effects.

Towards a more general framework. While Chapter 6 opens the way for a more general (re-)rendering framework by proposing both free-viewpoint navigation and relighting, the rest of the thesis treats problems separately. Finding a more general framework that can integrate free-viewpoint navigation with all kinds of editing at the same time is both a complex challenge and a very interesting research direction.

7.3 Thesis impact

The different projects presented in this thesis have led to publications and presentations in international journals and conferences, and also got some media exposure. The content of Chapter 3 has been presented at I3D 2018 as an oral presentation. The relighting part of Chapter 4 was presented at SIGGRAPH 2019, it was also presented at CVPR 2020 during a tutorial on neural rendering. While the author was interning at Adobe this project was also presented at AdobeMax under code name #LightRight. The scene composition extension presented in Chapter 4 will be presented at I3D 2020. Chapter 5 was part of a publication presented at SIGGRAPH Asia 2019. The work described in Chapter 6 is currently under review. All of the projects were implemented in the same code base, soon to be open-sourced, that also contains work from many other researchers. This will allow better dissemination and reproducibility of the complex pipelines presented.

7.4 Closing Remarks

This thesis introduces several methods that all aim at improving real asset rendering. We argue that this work is a first, but significant, step towards broader usage and dissemination of IBR methods. Multi-view editing methods not only allow for some degree of artistic control, which is crucial for a wider usage of IBR, but also relax some

constraints of the capture process.

We also show that novel learning based algorithms are not necessarily meant to replace traditional computer graphics. On the contrary we argue by the design of our pipelines that understanding both weaknesses and strengths of these two domains is a crucial step in building efficient methods. More specifically, in our setup, we leverage the efficiency and stability of traditional computer graphics for long range interactions while we use deep learning for localized image processing tasks, using each domain for what it does best.

Appendix A

Chapter 4 Appendices

A.1 Compositing and data augmentation details

Online compositing. We randomly scale the linear sun images by 2^u with u in $[-0.1, 0.1]$, separately for each color channel, allowing a small shift in sun color, and randomly scale the sky image by 2^u with u in $[-1.0, 1.0]$ (giving a uniform sampling between 0.5 and 2, centered at 1), with the same value for each channel. The latter is a simple approximation for sky turbidity since higher turbidity results in higher sky emission. We then sum sky and sun images.

Data-augmentation. To be robust to different exposures in the input, we apply a random exposure operation by multiplying the image by 2^e where e is in $[-2, 2]$. Similary, to handle white balance differences, we multiply each channel by 2^w with w randomly selected in $[-0.1, 0.1]$, separately for each channel.

We now have two linear images: one for source and one for target lighting condition. We perform the standard gamma correction operation to convert to sRGB with gamma randomly selected in $[2.0, 2.8]$ We clip to 1 for the source image, since all input images will be in the $[0, 1]$ range, but we clip the output to 2, to avoid zeroing the gradients and thus adversly affecting training for values slightly above 1.

A.2 Implementation details

- We do not apply transformations such as tanh or sigmoid at the end of the network allowing to produce images with a higher range than [0,1].
- We export our scenes using a 3DSMax to Mitsuba format exporter that we have developed in the 3DSMax scripting language. This gives flexibility in rendering

thanks to mitsuba but we loose a bit of quality of the materials as our exporter only partially handles “baked” materials. The specificities of the V-ray materials in the scenes used poses some issues of compatibility with Mitsuba.

- Generating path traced image for reconstruction would be prohibitively expensive as we require high resolution and little noise. Instead we generate lower resolution ambient occlusion images, upscale and multiply them with full resolution albedo images, which give a coarse approximation to global illumination (see Fig. A.1). This approximation is sufficient for reconstruction in most cases. For some scenes, the repetitive nature of the synthetic geometry and lack of texture detail, requires additional processing to allow SfM and MVS to succeed, e.g., by adding calibration targets in very uniform textures.

Figure A.1: Top: From left to right, the rendered ambient occlusion, the albedo , and the multiplication of both. The right most image is used along other point of views to reconstruct the scene and get MVS-like data. Bottom: The obtained reconstruction.

- Some ground truth scenes we used had very repetitive and/or flat textures which led to poor quality or failure in reconstruction when using Sfm + MVS. To allow for a reconstruction with the same level of quality as with real pictures we had to

modify some textures in two of our 10 scenes by adding targets on them. These modified textures were only used for reconstruction.

- Reconstructed scenes are not aligned with the original ground truth ones. To align them we used pairs of 3D points of the original ground truth scene and the proxy using the respective original and calibrated cameras. We then used iteratively reweighted least squares on those pairs to compute a transformation from proxy space to original (3DSMax) space.

Appendix B

Chapter 6 Appendices

B.1 Light-levels estimation for overexposed real scenes

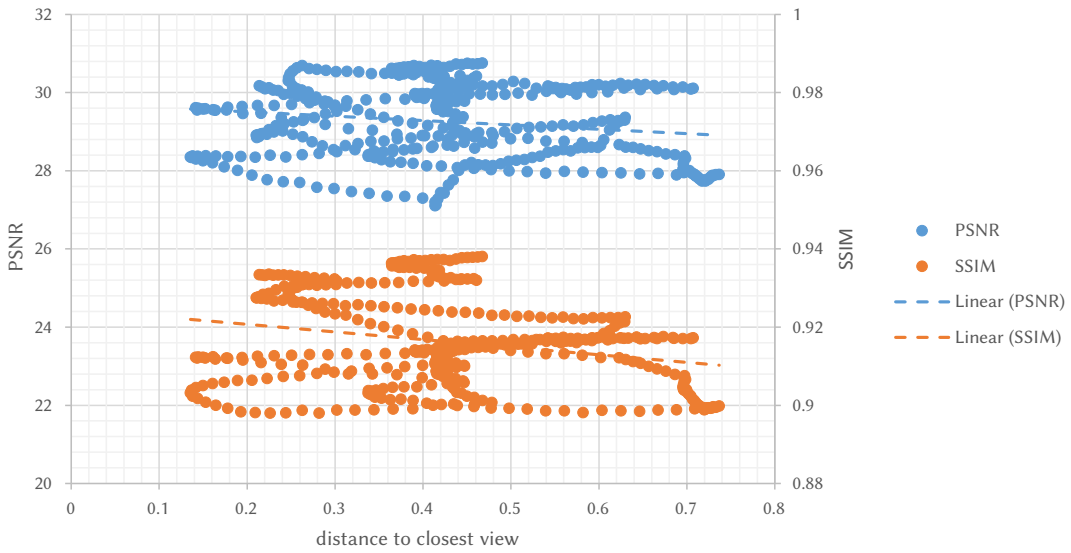
In real scenes, the absolute, global light level can be ambiguous because directly visible light sources may saturate the sensor and clip, even in high bit-depth captures. In such cases, we estimate the global energy level using a simple algorithm. We first detect the overexposed regions in 3D space, by filling a sparse voxel grid (2cm effective resolution). When at least 50% of the visible pixels (from the input views) that reproject to the voxel are clipped, we mark the voxel as overexposed. We then cluster the voxels; each clipped voxel initializes a cluster and we iteratively merge clusters whenever their bounding spheres intersect. The 3D region defined by a cluster roughly corresponds to a real-world light source. We compute a source irradiance map for each such light, assuming constant emittance over the entire region, and a separate irradiance map for the non-clipped pixels. We model the total irradiance as a linear combination of the non-clipped irradiance map and per-light irradiance maps, with unknown non-negative weights. We ask the user to click on a set of points with the same albedo in the scene. The selected points should all have the same albedo, approximated as image value divided by irradiance. This gives us equality constraints to assemble a linear system. We solve for the scalar weights using least-squares optimization; they correspond to each light's intensity. When a single light source is clipped, the user only needs to select two points, one lit by the overexposed light and one in its shadow to avoid an ill-conditioned linear system.

B.2 Dataset statistics for the real scenes

In our quantitative analysis on the synthetic scene *livingRoom2*, we found that the distance between the synthesized viewpoint and input cameras did not significantly correlate with error, as illustrated below. This shows our model generalizes well to distant viewpoints.

scene	mean	std
Living Room	0.187	0.07
Bedroom 1	0.174	0.11
Bedroom 2	0.173	0.11
Hall	0.240	0.14
Sofa	0.192	0.10
Kitchen	0.157	0.08

Table B.1: Mean and standard deviation (std) of distance to input camers for novel views in the test paths.



We also show the mean and standard deviation of distance to the input cameras in Table B.1.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/.

[2] Miika Aittala, Tim Weyrich, Jaakko Lehtinen, et al. Two-shot svbrdf capture for stationary materials. *ACM Trans. Graph.*, 34(4):110–1, 2015.

[3] Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernandez Esteban, Sameer Agarwal, and Steven M. Seitz. Jump: Virtual reality video. *ACM Transactions on Graphics (TOG)*, 35(6), 2016.

[4] Murat Arikan, Reinhold Preiner, Claus Scheiblauer, Stefan Jeschke, and Michael Wimmer. Large-scale point-cloud visualization through localized textured surface reconstruction. *IEEE Trans. Vis. Comput. Graphics*, 20(9):1280–1292, 2014.

[5] Murat Arikan, Reinhold Preiner, and Michael Wimmer. Multi-depth-map raytracing for efficient large-scene reconstruction. *IEEE Trans. Vis. Comput. Graphics*, 22 (2):1127–1137, 2016.

[6] Seung-Hwan Baek, Inchang Choi, and Min H. Kim. Multiview image completion with space structure propagation. In *Proc. of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016. IEEE.

[7] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3), August 2009.

[8] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3):24, 2009.

[9] Jonathan T. Barron and Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. *CVPR*, 2012.

[10] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015.

[11] D Berjon, F Moran, N Garcia, et al. Seamless, static multi-texturing of 3d meshes. In *Computer Graphics Forum*, volume 34, pages 228–238. Wiley Online Library, 2015.

[12] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proc. SIGGRAPH*, pages 417–424, 2000.

[13] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics*, 36(4), 2017.

[14] Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/.

[15] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, page 192–198, New York, NY, USA, 1977. Association for Computing Machinery. ISBN 9781450373555. doi: 10.1145/563858.563893. URL https://doi.org/10.1145/563858.563893.

[16] András Bódis-Szomorú, Hayko Riemenschneider, and Luc Van Gool. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 469–476, 2014.

[17] Nicolas Bonneel, Balazs Kovacs, Sylvain Paris, and Kavita Bala. Intrinsic decompositions for image editing. *Computer Graphics Forum*, 36(2):593–609, 2017.

[18] Adrien Bousseau, Sylvain Paris, and Frédo Durand. User-assisted intrinsic images. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, New York, NY, USA,

2009. Association for Computing Machinery. ISBN 9781605588582. doi: 10.1145/1661412.1618476. URL https://doi.org/10.1145/1661412.1618476.

[19] Chris Buehler, Michael Bosse, Leonard McMillan, and Steven Gortlerand Michael Cohen. Unstructured lumigraph rendering. In *Proc. SIGGRAPH*, 2001.

[20] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM, 2001.

[21] Marco Callieri, Paolo Cignoni, Massimiliano Corsini, and Roberto Scopigno. Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3d models. *Computers & Graphics*, 32(4):464–473, 2008.

[22] Rodrigo Ortiz Cayon, Abdelaziz Djelouah, and George Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *3D Vision (3DV), 2015 International Conference on*, pages 469–477. IEEE, 2015.

[23] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):98, 2017.

[24] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):30, 2013.

[25] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. *CoRR*, abs/1707.09405, 2017. URL http://arxiv.org/abs/1707.09405.

[26] Zhang Chen, Anpei Chen, Guli Zhang, Chengyuan Wang, Yu Ji, Kiriakos N Kutulakos, and Jingyi Yu. A neural rendering framework for free-viewpoint relighting. *arXiv preprint arXiv:1911.11530*, 2019.

[27] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep

learning. In *Proceedings of the NIPS Workshop on Deep Learning and Representation Learning*, 2014.

[28] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. Eurographics, 2008.

[29] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *SIGGRAPH Comput. Graph.*, 15(3):307–316, August 1981. ISSN 0097-8930. doi: 10.1145/965161.806819. URL https://doi.org/10.1145/965161.806819.

[30] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[31] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B. Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82:1–82:10, July 2012. ISSN 0730-0301.

[32] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[33] Paul Debevec. Image-based lighting. *IEEE Computer Graphics and Applications*, 22 (2):26–34, 2002.

[34] Paul Debevec. Virtual cinematography: Relighting through computation. *Computer*, 39(8):57–65, 2006.

[35] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 classes*, page 32. ACM, 2008.

[36] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156. ACM Press/Addison-Wesley Publishing Co., 2000.

[37] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (TOG)*, 37(4):128, 2018.

[38] Stavros Diolatzis, Adrien Gruson, Wenzel Jakob, Derek Nowrouzezahrai, and George Drettakis. Practical product path guiding using linearly transformed cosines. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 2020. URL http://www-sop.inria.fr/reves/Basilic/2020/DGJND20.

[39] Sylvain Duchêne, Clement Riant, Gaurav Chaurasia, Jorge Lopez-Moreno, Pierre-Yves Laffont, Stefan Popov, Adrien Bousseau, and George Drettakis. Multi-view intrinsic images of outdoors scenes with an application to relighting. *ACM Transactions on Graphics (TOG)*, 34(5), November 2015.

[40] Matthew DuVall, John Flynn, Michael Broxton, and Paul Debevec. Compositing light field video using multiplane images. In *ACM SIGGRAPH 2019 Posters*, page 67. ACM, 2019.

[41] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. In *ACM transactions on graphics (TOG)*, volume 23, pages 673–678. ACM, 2004.

[42] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. *Computer Graphics Forum*, 2008.

[43] Jihad El-Sana and Amitabh Varshney. Generalized view-dependent simplification. In *Computer Graphics Forum*, volume 18, pages 83–94. Wiley Online Library, 1999.

[44] Yaron Eshet, Simon Korman, Eyal Ofek, and Shai Avidan. DCSH - matching patches in RGBD images. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 89–96, 2013.

[45] Graham D Finlayson, Steven D Hordley, Cheng Lu, and Mark S Drew. On the removal of shadows from images. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):59–68, 2006.

[46] Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2): 141–151, 2005.

[47] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[48] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019.

[49] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Styles Overbeck, Noah Snavely, and Richard Tucker. Deepview: High-quality view synthesis by learned gradient descent. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. PAMI*, 2010.

[51] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *Proc. of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, pages 1422–1429. IEEE, 2009.

[52] Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. Seamless montage for texturing models. In *Computer Graphics Forum*, volume 29, pages 479–486, 2010.

[53] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1418–1425. IEEE, 2010.

[54] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[55] Marc-André Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagné, and Jean-François Lalonde. Deep parametric indoor lighting estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7175–7183, 2019.

[56] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[57] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. *CoRR*, abs/1505.01749, 2015.

[58] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[59] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *International Conference on Computer Vision (ICCV)*, 2007.

[60] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Siggraph*, volume 96, pages 43–54, 1996.

[61] Maciej Gryka, Michael Terry, and Gabriel J. Brostow. Learning to remove soft shadows. *ACM Transactions on Graphics (TOG)*, 34(5), October 2015.

[62] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2014.

[63] Ruiqi Guo, Qieyun Dai, and Derek Hoiem. Single-image shadow detection and removal using paired regions. In *CVPR, 2011*, pages 2033–2040. IEEE, 2011.

[64] Kaiming He and Jian Sun. Statistics of patch offsets for image completion. In *Computer Vision–ECCV 2012*, pages 16–29. Springer, 2012.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imageNet classification. *CoRR*, 2015.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[67] Paul S. Heckbert. Fundamentals of texture mapping and image warping. Technical Report UCB/CSD-89-516, University of California, Berkeley, 1989. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1989/CSD-89-516.pdf.

[68] Paul S Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *ACM SIGGRAPH Computer Graphics*, 24(4):145–154, 1990.

[69] Peter Hedman. *Viewpoint-free photography for virtual reality*. PhD thesis, University College London, UK, 2019. URL http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.785144.

[70] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics*, 35(6):231, 2016.

[71] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *ACM Transactions on Graphics (TOG)*, 36(6):234, 2017.

[72] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6):257:1–257:15, December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275084. URL http://doi.acm.org/10.1145/3272127.3275084.

[73] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*, pages 94–101. Springer, 1999.

[74] Heiko Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *Computer Vision and Patter Recognition (CVPR)*, pages 2386–2393, 2006.

[75] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-Francois Lalonde. Deep outdoor illumination estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[76] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-Francois Lalonde. Deep outdoor illumination estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[77] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. *arXiv preprint arXiv:1905.03897*, 2019.

[78] Daniel Reiter Horn and Billy Chen. Lightshop: interactive light field manipulation and rendering. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 121–128. ACM, 2007.

[79] Lukas Hosek and Alexander Wilkie. An analytic model for full spectral sky-dome radiance. *ACM Transactions on Graphics (TOG)*, 31(4):95, 2012.

[80] Joel Howard, Bryan S. Morse, Scott Cohen, and Brian L. Price. Depth-based patch scaling for content-aware stereo image completion. In *WACV*, pages 9–16. IEEE Computer Society, 2014.

[81] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Image completion using planar structure guidance. *ACM Trans. Graph.*, 33(4):129:1–129:10, July 2014. ISSN 0730-0301.

[82] Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Nießner. 3dlite: Towards commodity 3d scanning for content creation. *ACM Transactions on Graphics*, 2017, 2017.

[83] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017.

[84] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, July 2017.

[85] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[86] A G Ivakhnenko and V G Lapa. *Cybernetics and forecasting techniques*. Mod. Analytic Comput. Methods Sci. Math. North-Holland, New York, NY, 1967. URL

https://cds.cern.ch/record/209675. Trans. from the Russian, Kiev, Naukova Dumka, 1965.

[87] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[88] Wenzel Jakob. Mitsuba renderer, 2010.

[89] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3121–3128. IEEE, 2011.

[90] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[91] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[92] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4): 143–150, August 1986. ISSN 0097-8930. doi: 10.1145/15886.15902. URL https://doi.org/10.1145/15886.15902.

[93] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep high dynamic range imaging of dynamic scenes. *ACM Trans. Graph.*, 36(4):144–1, 2017.

[94] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):193, 2016.

[95] Yoshihiro Kanamori and Yuki Endo. Relighting humans: occlusion-aware inverse rendering for fullbody human images. *ACM Transactions on Graphics (TOG)*, 37 (270):1–270, 2018.

[96] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)*, 30 (6):157:1–157:12, December 2011. ISSN 0730-0301.

[97] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self Tuning Texture Optimization. *Computer Graphics Forum*, 2015. doi: 10.1111/cgf.12565.

[98] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4):127, 2014.

[99] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[100] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4):78, 2017.

[101] Johannes Kopf, Michael F. Cohen, and Richard Szeliski. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)*, 33(4):78:1–78:10, July 2014.

[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[103] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

[104] Pierre-Yves Laffont, Adrien Bousseau, Sylvain Paris, Frédo Durand, and George Drettakis. Coherent intrinsic images from photo collections. *ACM Transactions on Graphics (TOG)*, 31(6), 2012.

[105] Eric Lafortune. Mathematical models and monte carlo algorithms for physically based rendering. Technical report, 1996.

[106] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 179–195, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01267-0.

[107] Jean-François Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Estimating natural illumination from a single outdoor image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 183–190. IEEE, 2009.

[108] Jean-François Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. In *ACM Transactions on Graphics (TOG)*, volume 28, page 131. ACM, 2009.

[109] Jean-François Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Detecting ground shadows in outdoor consumer photographs. In *European conference on computer vision*, pages 322–335. Springer, 2010.

[110] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1):1–11, 1971.

[111] R. Lange and P. Seitz. Solid-state time-of-flight range camera. *IEEE Journal of Quantum Electronics*, 37(3):390–397, 2001.

[112] Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet. Surface Reconstruction from 3D Line Segments. In *2019 International Conference on 3D Vision (3DV)*, pages 553–563, Québec City, Canada, September 2019. IEEE. doi: 10.1109/3DV.2019.00067. URL https://hal.archives-ouvertes.fr/hal-02344362.

[113] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[114] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5928, 2019.

[115] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 2971–2980, 2018.

[116] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics*, 22(2):234–257, 2003. ISSN 0730-0301.

[117] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[118] Wenfeng Li and Baoxin Li. Joint conditional random field of multiple views with online learning for image-based rendering. In *Computer Vision and Pattern Recognition*. IEEE, 2008.

[119] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4), 2017.

[120] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018 Technical Papers*, page 269. ACM, 2018.

[121] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. *arXiv preprint arXiv:1905.02722*, 2019.

[122] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proc. of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[123] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[124] Céline Loscos, Marie-Claude Frasson, George Drettakis, Bruce Walter, Xavier Granier, and Pierre Poulin. Interactive virtual relighting and remodeling of real scenes. In *Rendering Techniques 99*, pages 329–340. Springer, 1999.

[125] Céline Loscos, George Drettakis, and Luc Robert. Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics*, 6(4): 289–305, 2000.

[126] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *CoRR, abs/1703.07511*, 2, 2017.

[127] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 199–208. ACM Press/Addison-Wesley Publishing Co., 1997.

[128] Stephen R Marschner and Donald P Greenberg. Inverse lighting for photography. In *Color and Imaging Conference*, volume 1997, pages 262–265. Society for Imaging Science and Technology, 1997.

[129] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6315–6324, 2018.

[130] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019.

[131] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322980. URL https://doi.org/10.1145/3306346.3322980.

[132] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[133] Ankit Mohan, Jack Tumblin, and Prasun Choudhury. Editing soft shadows in a digital photograph. *IEEE Computer Graphics and Applications*, 27(2), 2007.

[134] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4080–4089, 2019.

[135] Thomas Nestmeyer, Iain A. Matthews, Jean-François Lalonde, and Andreas M. Lehrmann. Structural decompositions for end-to-end relighting. *CoRR*, abs/1906.03355, 2019. URL http://arxiv.org/abs/1906.03355.

[136] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. Video inpainting of complex scenes. *CoRR*, abs/1503.05528, 2015.

[137] Fred E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4(7):767–775, Jul 1965. doi: 10.1364/AO.4.000767. URL http://ao.osa.org/abstract.cfm?URI=ao-4-7-767.

[138] Baptiste Nicolet, Julien Philip, and George Drettakis. *Repurposing a Relighting Network for Realistic Compositions of Captured Scenes*. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450375894. URL https://doi.org/10.1145/3384382.3384523.

[139] Rodrigo Ortiz-Cayon, Abdelaziz Djelouah, and George Drettakis. A Bayesian Approach for Selective Image-Based Rendering using Superpixels. In *International Conference on 3D Vision (3DV)*, Lyon, France, October 2015. URL https://hal.inria.fr/hal-01207907.

[140] Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. Optix: a general purpose ray tracing engine. In *Acm transactions on graphics (tog)*, volume 29, page 66. ACM, 2010.

[141] Philippe Pébay and Timothy Baker. Analysis of triangle quality measures. *Mathematics of Computation*, 72(244):1817–1839, 2003.

[142] Pieter Peers, Naoki Tamura, Wojciech Matusik, and Paul Debevec. Post-production facial performance relighting using reflectance transfer. In *ACM Transactions on Graphics (TOG)*, volume 26, page 52. ACM, 2007.

[143] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):235, 2017.

[144] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics*, 36, 2017.

[145] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003. ISSN 0730-0301.

[146] Julien Philip and George Drettakis. Plane-based multi-view inpainting for image-based rendering in large scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, page 6. ACM, 2018.

[147] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. 2019.

[148] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A. Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Trans. Graph.*, 38 (4):78:1–78:14, July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323013. URL http://doi.acm.org/10.1145/3306346.3323013.

[149] Liangqiong Qu, Jiandong Tian, Shengfeng He, Yandong Tang, and Rynson W. H. Lau. Deshadownet: A multi-context embedding deep network for shadow removal. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[150] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[151] Capturing Reality. Realitycapture reconstruction software. https://www.capturingreality.com/Product, 2018.

[152] CapturingReality RealityCapture. Realitycapture, 2016. URL \protect{http://https://www.capturingreality.com/}.

[153] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 267–276, 2002.

[154] Gernot Riegler and Vladlen Koltun. Free view synthesis, 2020.

[155] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.

[156] Andres Sanin, Conrad Sanderson, and Brian C Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition*, 45(4):1684–1695, 2012.

[157] Hiroto Sasao, Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya. A study on effect of automatic perspective correction on exemplar-based image inpainting. *ITE Transactions on Media Technology and Applications*, 4(1):21–32, 2016.

[158] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[159] Johannes L Schönberger. Colmap: A general purpose SfM/MVS system, http://colmap.github.io, 2016.

[160] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[161] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[162] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[163] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. *arXiv preprint arXiv:1901.02453*, 2019.

[164] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W. Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *International Conference on Computer Vision (ICCV)*, 2019.

[165] C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37 (1):10–21, jan 1949. doi: 10.1109/jrproc.1949.232969. URL https://doi.org/10.1109/jrproc.1949.232969.

[166] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013.

[167] Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5444–5453. IEEE, 2017.

[168] Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *ACM Transactions on Graphics (TOG)*, volume 27, page 159. ACM, 2008.

[169] Sudipta N. Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE ICCV 2009*, pages 1881–1888, 2009.

[170] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 2006.

[171] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics (TOG)*, 25(3):835–846, July 2006. ISSN 0730-0301.

[172] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017.

[173] Jessi Stumpfel, Chris Tchou, Andrew Jones, Tim Hawkins, Andreas Wenger, and Paul Debevec. Direct hdr capture of the sun and sky. In *Proceedings of the 3rd*

international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, pages 145–149. ACM, 2004.

[174] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. Image completion with structure propagation. In *ACM Transactions on Graphics (ToG)*, volume 24, pages 861–868. ACM, 2005.

[175] Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *arXiv preprint arXiv:1905.00824*, 2019.

[176] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored time-lapse video. In *ACM Transactions on Graphics (TOG)*, volume 26, page 101. ACM, 2007.

[177] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[178] Marshall F Tappen, William T Freeman, and Edward H Adelson. Recovering intrinsic images from a single image. In *Advances in neural information processing systems*, pages 1367–1374, 2003.

[179] Chris Tchou, Jessi Stumpfel, Per Einarsson, Marcos Fajardo, and Paul Debevec. Unlighting the parthenon. In *ACM Siggraph 2004 Sketches*, page 80. ACM, 2004.

[180] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhöfer. State of the art on neural rendering, 2020.

[181] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *arXiv preprint arXiv:1904.12356*, 2019.

[182] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics 2019 (TOG)*, 2019.

[183] Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *Fourth International Conference on 3D Vision, 3DV 2016*, pages 351–359, Oct. 2016.

[184] Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 351–359. IEEE, 2016.

[185] Benjamin Ummenhofer and Thomas Brox. Global, dense multiscale reconstruction for a billion points. *International Journal of Computer Vision*, 125(1):82–94, 2017.

[186] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV 2014*, pages 836–850. Springer International Publishing, 2014.

[187] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195–206, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673524.

[188] Jifeng Wang, Xiang Li, and Jian Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2018.

[189] Yang Wang, Lei Zhang, Zicheng Liu, Gang Hua, Zhen Wen, Zhengyou Zhang, and Dimitris Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1968–1984, 2009.

[190] Henrique Weber, Donald Prévost, and Jean-François Lalonde. Learning to estimate indoor lighting from 3d objects. In *2018 International Conference on 3D Vision (3DV)*, pages 199–207. IEEE, 2018.

[191] Yair Weiss. Deriving intrinsic images from image sequences. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 68–75. IEEE, 2001.

[192] Zhen Wen, Zicheng Liu, and Thomas S. Huang. Face relighting with radiance environment maps. In *CVPR*, 2003.

[193] Andreas Wenger, Andrew Gardner, Chris Tchou, Jonas Unger, Tim Hawkins, and Paul Debevec. Performance relighting and reflectance transformation with time-multiplexed illumination. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 756–764. ACM, 2005.

[194] Paul Werbos and Paul John. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974.

[195] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476, 2007.

[196] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Get out of my pictureinternet-based inpainting. In *BMVC*. British Machine Vision Association, 2009.

[197] Oliver Woodford and Andrew W Fitzgibbon. Fast image-based rendering using hierarchical image-based priors. In *BMVC*, volume 1, pages 260–269, 2005.

[198] Oliver J Woodford, Ian D Reid, Philip HS Torr, and Andrew W Fitzgibbon. Fields of experts for image-based rendering. In *BMVC*, volume 3, pages 1109–1108, 2006.

[199] Oliver J Woodford, Ian D Reid, and Andrew W Fitzgibbon. Efficient new-view synthesis using pairwise dictionary priors. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[200] Jung-Hsuan Wu and Suguru Saito. Interactive relighting in single low-dynamic range images. *ACM Transactions on Graphics (TOG)*, 36(2):18, 2017.

[201] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4):126, 2018.

[202] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 38(4):76, 2019.

[203] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. *ACM Transactions on Graphics*, 2017, 2017.

[204] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224. ACM Press/Addison-Wesley Publishing Co., 1999.

[205] Ya-Ting Yue, Yong-Liang Yang, Gang Ren, and Wenping Wang. Scenectrl: Mixed reality enhancement via efficient scene editing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 427–436. ACM, 2017.

[206] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016.

[207] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

[208] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics*, 33(4):155, 2014.

[209] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.

[210] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4):65, 2018.

[211] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.