Unstructured Image Mosaics

Rahul Garg

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Steven M. Seitz, Chair

Aseem Agarwala

Richard Szeliski

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Unstructured Image Mosaics

Rahul Garg

Chair of the Supervisory Committee:
Professor Steven M. Seitz
Computer Science & Engineering

Image mosaics stitch photos into a single composite with a wide field of view. They are easy to create and can be panned by dragging the mouse, thus enabling simple and effective photorealistic visualizations. However, they are restricted to structured input where the camera motion is limited and the appearance variation across photos can be controlled. In my thesis, I extend mosaics to unstructured cases that include more general camera motion, appearance variation and Internet photo collections, enabling visualization of more complex scenes.

I first develop a mosaicing technique for a general class of photo collections subsuming rotational mosaics. In contrast to prior approaches that stitch a single static mosaic, my approach dynamically composites a mosaic based on the current viewpoint as the user navigates, allowing for distortion-free mosaicing and a broader range of camera motion. Exposure differences, parallax, misalignment between photos and scene motion lead to stitching artifacts in static mosaics. My dynamic approach avoids these artifacts and enhances realism by preserving scene motion and changes in brightness.

In the latter part of my thesis, I focus on highly unstructured collections comprising tourist photos downloaded from the Internet that are not captured with the intention of creating a mosaic. First, I describe an algorithm to discover rotational panoramas

(photos taken from nearly the same viewpoint) and orbits (photos looking at a common object), within these collections. These photo sets can then be browsed by dragging the mouse like traditional mosaics. Second, I focus on extreme variation in appearance of these photos. I prove that any photo of a scene can be represented as a linear combination of a set of basis photos. I show theoretically and empirically that under suitable assumptions, for a scene with k_n distinct orientations and k_ρ different materials, $k_n k_\rho$ basis photos are sufficient to span the space of all possible photos of the scene. I then describe a method to robustly compute these basis photos from Internet photos and show novel applications like removing people and expanding the field of view of a photo.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I wish to thank my advisor Steve Seitz, who not only taught me the significance of methodical thinking and thorough experimentation in doing research, but also imbibed in me the importance of presenting and communicating the results effectively. This thesis would not have been completed without his guidance and effort, and his patience with my mistakes. I thank Rick Szeliski and Aseem Agarwala for reading drafts of this thesis and providing useful feedback. I thank James Fogarty and Cecilia Aragon for serving on my thesis commitee.

I also had the fortune of working with a number of wonderful collaborators – Noah Snavely, Rick Szeliski, Ryan Kaminsky, Hao Du, Deva Ramanan, Ira Kemelmacher and Eli Shechtman. I would like to especially thank Noah who was virtually my second advisor during the first two years. I also wish to thank my undergraduate advisors, Subhashis Banerjee and Manik Varma, who introduced me to computer vision and encouraged me to pursue graduate studies.

The space is too small to acknowledge all the friends in Seattle, who made my stay both enjoyable and memorable. I am also indebted to my friends from high school and undergrad, who stayed in touch despite the distance and endured my rants.

Finally, I cannot adequately thank my parents for their unconditional love and support despite their own problems, including fighting cancer.

Chapter 1

INTRODUCTION

If I could tell the story in words, I wouldn't need to lug around a camera.

– Lewis Hine

How can a user virtually navigate a scene? A system that allows the user to virtually explore a scene should be capable of rendering a realistic view of the scene from any viewpoint. The traditional computer graphics approach to this problem is to create a 3D texture mapped model of the scene and then render the scene from the desired viewpoint. While much progress has been made in this area (Figure 1.1), producing realistic renderings require intensive data capture and manual effort, e.g., acquisition of scene geometry using 3D scanners or modeling using CAD tools, measurement or manual modeling of surface properties like the bidirectional reflectance distribution function (BRDF), modeling illumination in the scene, modeling of natural phenomenon like fog, clouds, water, smoke, etc. Often, the data collection and processing step is not automated and many man-hours are required. Further, accurate modeling of physical interaction of light with the scene elements is computationally intensive; hence many systems approximate these interactions with faster alternatives which may not yield physically accurate renderings. As a result, these approaches have been limited to select areas like video games where it is feasible to spend effort on data capture and processing and have not pervaded common use, e.g., creating a virtual walkthrough of someone's house using a consumer-level hand-held camera.

For a moment, let us consider a simpler problem than navigating the entire scene. Imagine standing at a spot in the scene and looking around – how can one virtually

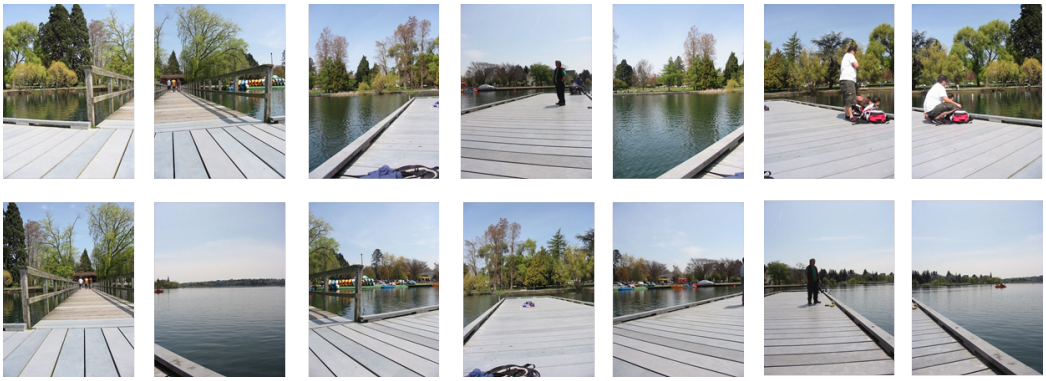
(a) Captured photo (b) Rendered photo

Figure 1.1: (a) A captured photo. (b) A photorealistic rendering from video game Gran Turismo 5 [41]. Intensive data capture and many man-hours are required to yield a rendering as good as a captured photo.

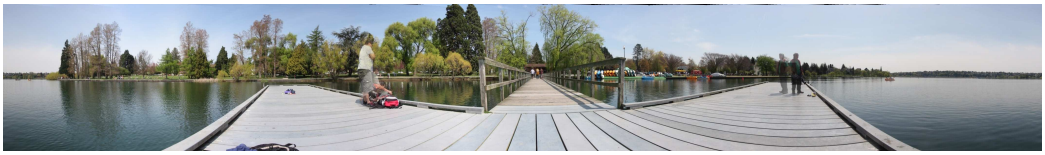
recreate this experience? An easy approach is to capture images looking in different directions from the same spot and stitch them into a single image mosaic [97, 99]. Such a mosaic can then be browsed in an interactive viewer as shown in Figure 1.2 where the user can look in different directions by simply dragging the mouse around. There are clear advantages of using image mosaics for scene visualization:

- Such a visualization is very easy to create – the only input required is a set of photos looking in different directions captured by a hand-held consumer level camera and the rest of the processing happens automatically.

- Individual photos are stitched into a single seamless mosaic – as a result it feels



(a)



(b)

(c)

Figure 1.2: (a) A few photos out of 18 captured from a single spot (b) An image mosaic stitched from the photos. (c) The same image inside the Quicktime VR [18] viewer which allows the user to pan and zoom the image interactively.

like browsing the scene instead of individual photos.

- The only navigation controls needed are mouse drag and zoom.

Recognizing these advantages, the broad goal of my thesis is to generalize and extend image mosaics so as to generate visualizations of a scene that are both easy to create and easy to navigate. While image mosaics have been limited to structured photo collections and seen as a way to capture ultra-wide angle or ultra-high resolution

Figure 1.3: Street side mosaic stitched from 107 photos [1].

images, I seek to generalize them to more unstructured cases and enable virtual navigation of complex scenes.

Due to the ease of creating and navigating mosaics, they have become very popular. Panoramic photography first emerged in the late 19th century – special cameras with curved film holders that employed clockwork drives to scan a line image in an arc were used to create an image over almost 180°. In the mid 1990's image alignment techniques were applied to stitch mosaics from images captured by regular hand-held cameras [61, 97, 18, 99].

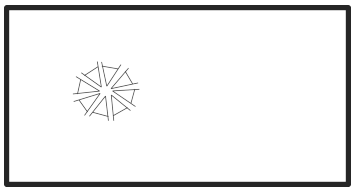
More recently, image mosaics have found their way into many real world applications. Many consumer level cameras have a panorama mode which facilitates capturing a sequence of images and then stitching them into a mosaic. With the advent of smartphones, this has become even easier with panorama capturing apps. In fact, Microsoft's Photosynth app [6] that allows one to capture panoramas using an iPhone was rated the 2^{nd} most popular iPhone app of 2011 [100]. Google streetview [5] and Bing maps [62] capture entire cities using 360° image mosaics captured at every few meters along the streets.

Despite the popularity of image mosaics, they suffer from limitations which impede their applicability, especially for scene exploration and navigation. I discuss three critical limitations here which I seek to address in my work:

1. **Restricted camera motion:** Image mosaics have largely been seen as a way to capture ultra wide angle or 360° images, i.e., they are limited to rotational motion – one can only visualize how the scene appears from a single spot in the

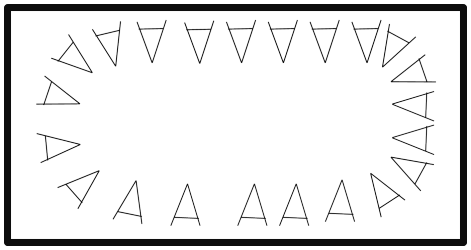
scene. Another kind of mosaics that have recently become popular are mosaics of a planar scene captured by a translating camera [1] (Figure 1.3). Such mosaics can capture long planar scenes, e.g., a street side. However, they are again restricted to the specific case of a planar scene and a translating camera. Can one extend image mosaics to handle more general camera motion?

To give an example, consider an art gallery in the form of a rectangular room



(a) Rotating Camera

(b) Rotational mosaic of a rectangular room



(c) Desired capture

(d) A sequence of zoom outs from my approach

Figure 1.4: (a) A rectangular art gallery with paintings along the wall captured using a single rotational mosaic (b) A rotational mosaic captured inside the rectangular room. Paintings on the sides at the further end appear highly distorted and have a low resolution – it is impossible to uniformly capture all the paintings along the wall due to the oblong shape of the room. (c) On the other hand, a collection of photos taken along the walls of a room captures the room more uniformly. Such a collection ends up being a combination of translational and rotational mosaics but it cannot be stitched into a single mosaic using existing methods. (d) My approach where the user can view all the paintings almost head-on by simply dragging the mouse.

(a) (b)

Figure 1.5: (a) Naive stitching of two images with different exposures lead to artifacts near image boundaries. Image from [103]. (b) In presence of scene motion, blending across images leads to ghosting artifacts. Image from [22].

with paintings along the wall. If one wants to use an image mosaic to capture the paintings along the walls, one option is to capture a rotational image mosaic from the center of the room. However, since the room is rectangular, a single rotational mosaic will not be able to capture all the paintings uniformly; paintings which are viewed side on at the corner of the room appear highly distorted. Further, paintings which are far from the optical center of the panorama have low resolution (Figures 1.4a and 1.4b). One can capture the individual walls as separate translational mosaics where each painting will have a head-on view, but one cannot capture the entire room using a single mosaic with existing automatic mosaicing methods. Intuitively, one should be able to capture a sequence of photos as shown in Figure 1.4c, which capture all the walls uniformly and browse them seamlessly using the same drag and zoom navigation that mosaics provide (Figure 1.4d).

2. **Stitching artifacts: Exposure differences, misalignment, scene motion and parallax:** Stitching together two or more images may lead to artifacts along the seams (Figure 1.5) due to different exposures or color balance, alignment errors, moving objects in the scene, etc. There exists an extensive body of work

[2, 1, 11, 14, 22, 27, 53, 54, 98, 97, 99, 103] to avoid such artifacts. However, the primary goal of past approaches has been to generate a single static seamless image mosaic as opposed to an interactive experience. When the goal is to create a realistic visualization of the scene, different solutions may be employed. E.g., in a scene with varying brightness, our eyes adjust to the brightness as we look around (the inside of a room becomes dark when peering outside a bright window). Ideally, the virtual system should recreate this experience. Similarly, scene motion like waving trees, moving people, may form an integral part of the scene and recreating them in the virtual system will lead to a more realistic reproduction of the scene.

3. **Requires specialized capture:** Image mosaicing approaches have been limited to a single person taking photos with the intention of creating a mosaic. However, there exists a huge amount of imagery on the Internet on photo sharing websites like like Picasa [74], Flickr [33], Facebook [30], etc. To give an idea of the magnitude of the sizes of these collections, approximately 70 billion photos were uploaded to Facebook alone in the year 2010 [38]. Leveraging these large image collections to build mosaics of remote places that one cannot capture oneself offers exciting opportunities and challenges.

   Unfortunately, standard image mosaicing approaches may not be directly applicable to these collections. First, these photos are tourist photos and are not captured with the intention of creating a mosaic. Hence, even in these large collections, it may not be guaranteed, for instance, that there exists a set of photos taken from *exactly* the same location that can be stitched into a rotational mosaic. Even if such a set of photos exists, identifying this set in a collection of thousands of photos is a non-trivial task. Second, the wide variation in appearance across these photos poses a problem for mosaic creation. These photos are captured by different people, by different cameras, at different times

(a) (b)

Figure 1.6: A day time and a night time photo of Trevi fountain downloaded from Flickr [33]. Appearance is very different due to different lighting conditions and people posing in front of the scene.

of the day, and under different weather conditions. The variation in appearance is much larger than the variation in photos captured by a single user over a short time interval for creating a mosaic, and hence standard image mosaicing approaches to compensate for appearance differences may not be applicable here. Figure 1.6 shows two photos of the Trevi fountain demonstrating the wide variation in appearance that can exist between such photos.

Hence, while image mosaics are easy to create and facilitate intuitive drag and zoom navigation, their application is limited due to the above mentioned reasons. In my thesis, I seek to address these limitations of image mosaics, thereby greatly increasing their applicability and generality. My specific contributions are as follows:

1. **Generalized camera motion:** I describe a new kind of mosaics, *dynamic mosaics*, that enable piecewise-perspective mosaicing that preserves straight lines in the scene. Further, they support generalized camera motion which subsumes rotational and translation mosaics.

(a) Camera path (b) Captured photos

Figure 1.7: A collection of 43 photos taken on a staircase capturing paintings on the wall.

Figure 1.8: Dynamic mosaic created from a set of photos taken along a staircase. The top row shows a sequence of zoom-outs corresponding to the upper stairs highlighted at left. The bottom row shows zoom outs for the landing (left). User can smoothly move between different locations simply by dragging the mouse.

To generate an image mosaic, one typically needs to choose a stitching surface on to which all images are composited. The choice of stitching surface depends on the expected type of camera motion. E.g., a cylindrical or a spherical surface is typically used when the camera is purely rotating, and a planar surface is used for a translating camera and planar scenes. In the case of general camera motion, a surface that adapts to the camera motion is needed. E.g., Peleg *et al.* [71] propose an approach that finds such an adaptive surface based on

optical flow from a video sequence. Lin *et al.* [56] use a spatially varying affine transform to stitch images that allows them to stitch images with parallax and motion. However, such approaches which use non-perspective projections introduce unnatural distortion of the scene – straight lines in the scene may become bent in an arbitrary fashion.

I circumvent this problem by relaxing the requirement that we need to generate a single static mosaic. Instead, I develop an interactive viewer that composites a piecewise-perspective mosaic that preserves straight lines on the fly as the user navigates, based on the images that are currently in the user's view. This approach is applicable to a general class of image collections subsuming rotational and translational mosaics that I call *locally stitchable*. Figure 1.7 shows a collection of 43 photos captured by walking down a staircase and Figure 1.8 shows the corresponding dynamic mosaic being viewed inside my viewer. The viewer supports the intuitive drag and zoom navigation of traditional mosaics. This is achieved by formulating the dynamic compositing problem as an optimization problem, that seeks to minimize the distortion of images that are currently visible to the user, and solving it at frame rate as the user navigates.

2. **Dynamic aspects of the scene:** Dynamic mosaicing also recreates dynamic aspects of the scene. E.g., in Figure 1.9, as the user navigates to the part with the bright window, the rest of the scene becomes dark imitating the way the human eye adjusts to the varying brightness in the scene. To this end, I model the exposure and color balance differences across images using a simple model whose parameters are computed as a preprocessing step. Then at render time, these parameters are optimized dynamically according to the current set of images visible to the user in a fashion similar to the dynamic tone mapping approach of Kopf *et al.* [52].

   Similarly, I propose a novel solution to avoid ghosting artifacts which result from

(a) (b) (c)

Figure 1.9: (a) 6 out of 18 input photos. (b) A rendering from the viewer where the interior is well exposed while the window on the right is saturated. (c) As the windows come into view, the brightness of the scene is adjusted automatically; view outside the window is visible.

(a) Frames from a panning video

(b) Dynamic mosaic; jumper moves forward as the user pans from left to right.

Figure 1.10: Frames from a panning video of a triple jumper can be composited into a dynamic mosaic where the jumper moves forward in stroboscopic fashion as the user navigates forward.

misalignment, parallax between images, or scene motion. Instead of choosing a static set of optimal seams for the entire mosaic as in previous approaches, the proposed viewer dynamically chooses a set of seams that is optimal for the *current view*. The seams move as the user navigates, recreating some of the dynamic aspects of the scene. Figure 1.10 shows a particular kind of dynamic mosaic that can be created from the frames of a panning video. Further, the viewer also performs multi band blending [14] across these seams and I develop a blending algorithm that works at frame rate on standard PCs.

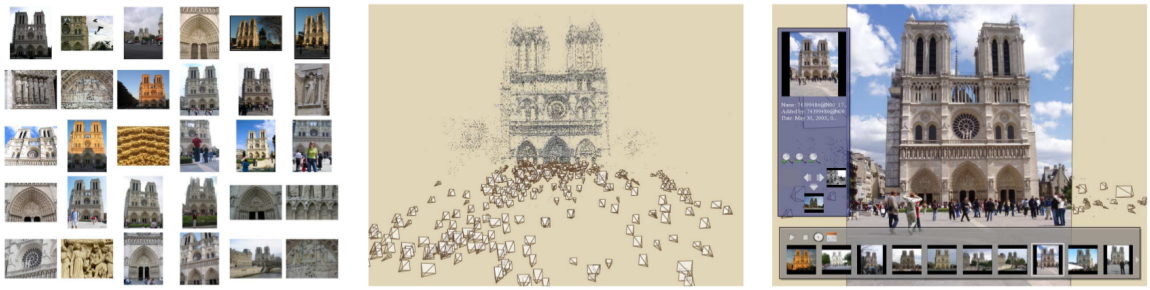
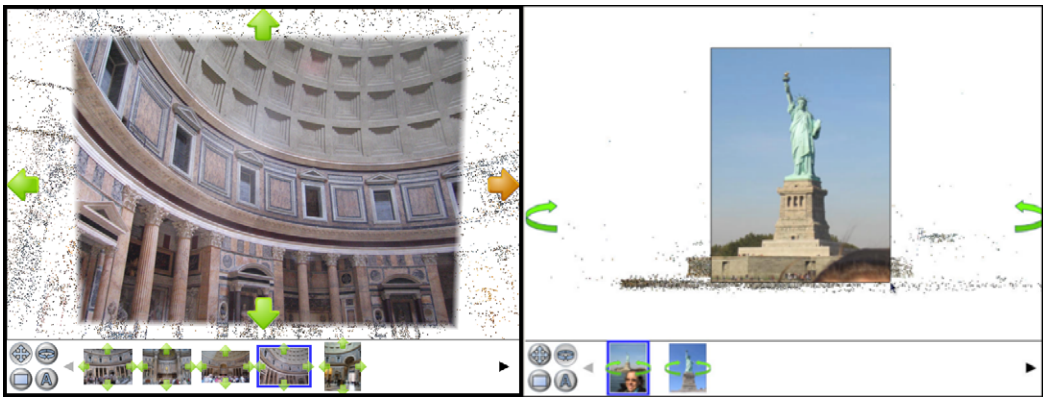


Figure 1.11: *The PhotoTourism system [94]* takes as input a collection of photos of a scene downloaded from the Internet. The system then recovers the 3D pose of each photo and a sparse model of the scene. Using this information the photos can be browsed in an interactive and immersive 3D browser. Figures from *[94]*.

3. **Mosaicing for Internet photo collections:** I explore how image mosaics can be extended to Internet photo collections. The starting point of my work is the PhotoTourism system of Snavely *et al.* [94] that takes as input a collection of photos of a scene downloaded from the Internet, computes the 3D pose of every photo and a sparse reconstruction of the scene, and uses that information to allow the user to explore the photo collection in 3D (Figure 1.11)

   The system shows a single best photo to the user at each time instant and presents the user with photo-centric controls, i.e., the user can move from photo to photo with 3D transitions between them. While these discrete controls work well for the purpose of browsing a large photo collection, they may not work well for navigating and exploring the scene in a way that helps the user to form a mental map of the scene.

   Image mosaics provide free viewpoint navigation with fewer degrees of freedom as opposed to the 6 or 7 degrees typically available in unconstrained free viewpoint 3D navigation. While it is difficult to stitch seamless mosaics from these highly unstructured photo collections (images are not taken from exactly the same viewpoint and there can be extreme appearance variation), I propose a method



(a) Rotational panoramas (b) Orbits

Figure 1.12: (a) Automatic detection of rotational panoramas in Internet photo collections. These panoramas can be selected using the pane at the bottom, and browsed using simple drag and zoom interactions. (b) Orbital controls for the Statue of Liberty. The system detects two orbits situated at different radii from the statue. These can be selected using the thumbnails at the bottom and navigated by simply dragging the mouse left and right.

to automatically discover two types of free viewpoint controls with constrained degrees of freedom for these image collections that are inspired by panoramic navigation:

(a) **Rotational panoramas:** I propose an algorithm to automatically identify groups of images which are taken from *nearly* the same viewpoint. While stitching them together into a single seamless mosaic is still a problem due to significant parallax (they are not captured from exactly the same viewpoint), appearance variation, and occluders (people posing in front of scenes), I provide panorama-like $2D$ navigation controls for such groups of photos (Figure 1.12a). The user can drag in different directions using the mouse to see new photos.

(b) **Orbits:** Many scenes often contain objects or points of interest which are

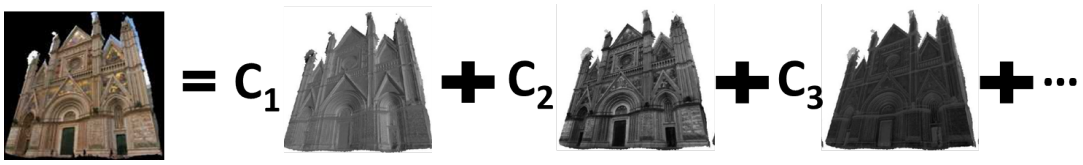


Figure 1.13: Under suitable assumptions, any appearance of a scene can be expressed as a linear combination of a set of few basis appearances.

photographed by tourists from different directions. E.g., a lot of people take photos of the Statue of Liberty from different directions. I develop an approach to identify such groups of photos which are all looking at some common point of interest and are arranged in a circular arc. These orbits can then be browsed seamlessly in a fashion similar to rotational mosaics – dragging the mouse left and right simulates moving along the orbit. (Figure 1.12b).

These controls are discovered automatically from the distribution of photos.

4. **Appearance space of a scene:** Image stitching approaches compensate for minor appearance differences across images like exposure changes and different color balance. However, the variation in appearance for Internet photos is much wider. I propose an approach to model the space of appearance of a scene under different conditions.

   Internet photos provide a sampling of different possible scene appearances (Here, by appearance of a scene, I refer to a photo of a scene taken under a specific set of conditions). While there is a lot of variation across these photos, I claim that these appearances can be modeled as a *linear combination* of a set of few *basis appearances* (Figure 1.13).

   The result is based upon the observation that man-made scenes are not random – they contain (or can be approximated by) only a few distinct surface normal

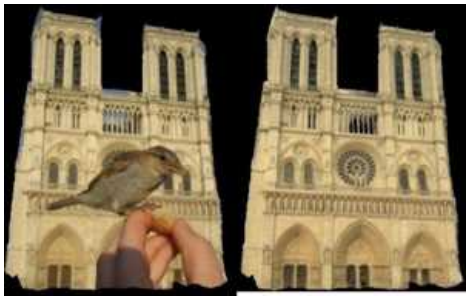


Figure 1.14: I introduce an approach to model the appearance space of a scene from Internet photos. It can then be used to remove occluders from the original photo (left) and fill the hole with new pixels whose appearance is in agreement with the appearance of the rest of the scene (right).

directions and surface reflectance properties (distinct materials in the scene). I first prove a theoretical result stating that for a scene with k_n distinct normal directions and k_ρ different materials, $k_n k_\rho$ basis appearances are sufficient to span the space of all possible appearances of the scene under suitable assumptions. I extend this basic result in a number of interesting directions to span the space of images of a scene with real world illumination and viewing conditions, allowing for general surface reflectance. In addition to theoretical proofs, I also conduct empirical studies to justify the assumptions made. The theory I develop provides justification for the popularity of low dimensional linear models in computer vision. First exploited in the early work on eigenfaces [48, 102], low linear-dimensionality has become the basis for a broad range of successful applications in recognition [72, 66], tracking [42], background modeling [68], image-based rendering [104], BRDF modeling [44, 63], compression and other domains. The new theoretical results suggests that linear models can be used successfully for modeling the appearance of scenes even under uncontrolled conditions, for instance Internet photo collections.

In addition to proving theoretical results, I also develop a practical method to

compute the set of basis appearances corresponding to a scene from Internet photos. The recovered basis appearances can then be used to model the appearance space of a scene and perform tasks such as removing occluders from an image (Figure 1.14). My method relies on accurate 3D models of the scene limiting its applicability to scenes where such models are available. However, with more 3D data becoming available and advances in multi view stereo methods, this limitation will be less severe in future.

The organization of my thesis is as follows. In Chapter 2, I review related work. In Chapter 3, I introduce dynamic mosaics, where I develop a dynamic viewer that composites a mosaic on the fly as the user navigates. These dynamic mosaics are applicable for more general camera motion than is possible with past approaches and also capture dynamic aspects of the scene. In Chapter 4, I develop algorithms to discover image mosaics within Internet photo collections and provide controls to navigate them which are analogous to the dragging controls in a mosaic. In Chapter 5, I discuss the problem of extreme variation in appearance in Internet photo collections. I first present theoretical and experimental results and later develop algorithms to recover basis photos from Internet photo collections. Finally, I conclude in Chapter 6 with future research directions that build upon my work.

Chapter 2

RELATED WORK

An image mosaic is a composite stitched from multiple photos, typically used to create a wide field of view image. Image mosaics date back to late 1700s, first introduced in the context of paintings. I discuss the evolution of image mosaics in four roughly chronological phases: (1) early mosaics captured using special cameras, (2) image mosaics stitched from multiple images using computer vision techniques, (3) image mosaics for interactive scene visualization, and (4) image mosaicing approaches that capture dynamic elements, e.g., motion in the scene, instead of generating a single static image.

I use the terms *mosaics* and *panoramas* interchangeably in my thesis. Initially, the primary intention of image mosaics was to capture wide field of view images that cannot be captured by a conventional single lens camera. Early approaches discussed under Section 2.1 captured panoramas using special cameras with modified optics while approaches in Section 2.2 use computer vision methods to stitch images taken by a conventional camera into a panoramic image. In Section 2.3, I discuss systems where image mosaics are used as building blocks for interactive scene visualization. However, the camera motion is still primarily restricted to rotational motion with discrete transitions between individual mosaics that may be disorienting to the user. In my work, I extend image mosaics to more general camera motion enabling interactive visualization of more complex scenes, and also to unstructured photo collections, e.g., Internet photo collections. Under Section 2.4, I discuss some examples of recent approaches that deviate from the paradigm of generating a single static mosaic and are in line with my work on recreating dynamic aspects of the scene like varying

brightness and scene motion.

2.1 Early Panoramic Images

Panoramic images were first introduced in paintings, with the term panorama coined by Irish painter Robert Barker in 1792 for his panoramic paintings of Edinburgh [69], which were displayed on a cylindrical surface and viewed from inside. Soon, image mosaics became popular in photography as a method for capturing very wide field of view images. The first recorded patent for a panoramic camera is by Joseph Puchberger in Austria in 1843 for a hand-cranked, 150° field of view, 8-inch focal length camera [70]. The camera exposed a sequence of photographic plates which were then pieced together to generate a panoramic image. Panoramic cameras were revolutionized by the invention of flexible film in 1888. The new cameras used a lens that rotated around the camera's nodal point and a curved film was placed behind it. As the lens rotated, a slit exposed a vertical strip of film that was aligned with the axis of the lens.

These early methods relied on hardware solutions to capture panoramic images and were cumbersome to use. Further, they all seek to capture a single static wide angle image that could not be captured using conventional cameras.

2.2 Image Mosaicing via Computer Vision Methods

While panoramic cameras with modified optics were cumbersome to use, development of automatic computer vision techniques to align and stitch overlapping images of a scene captured by conventional hand-held cameras has greatly simplified the process. There are three key steps in creating an image mosaic from a collection of images.

- **Image matching and alignment:** First, one needs to find geometric transforms between different images to align them. In the 1980s, automatic image alignment methods were developed [60, 76, 4, 9] that iteratively improve the

alignment by minimizing an error metric between the corresponding pixels of the two images. However, such methods are prone to getting trapped in local minima. Feature-based methods have become more popular recently due to their superior robustness [12, 11]. These methods work by finding repeatable and distinct features in images, e.g., Scale Invariant Feature Transform (SIFT) [59], and then matching these features across images. Feature correspondences are then used to robustly estimate alignment transforms between images. I refer the reader to the excellent survey article on image mosaicing by Szeliski [98] for more details on image alignment approaches. While most image alignment methods typically assume a simplified camera motion model, e.g., pure camera rotation, they are robust enough to work in case of more general camera motion as well. E.g., PhotoTourism system [94] recovers the full 3D pose of each photo using a feature-based matching method. Hence, image matching and alignment is not the limiting factor when extending image mosaics to more general camera motion.

- **Image compositing:** In order to generate a single composite, images are aligned and placed on a common surface. The surface can then be unwrapped to yield a single image. The compositing surface is usually chosen based on the camera motion involved. E.g., images captured by a rotating camera are often composited onto a cylindrical or spherical surface. I discuss this step in more detail in Section 2.2.1 in order to understand how the choice of compositing surface restricts the range of camera motion.

- **Blending:** A single pixel in the final stitched composite corresponds to a 3D point in the scene. The projection of that 3D point may be seen in multiple images and may appear differently due to different exposures or different color balance across images. Yet, the color of each pixel in the final composite must be chosen to ensure that the stitch is seamless. The problem is further complicated

Figure 2.1: A panorama of a street side captured by a translating camera and stitched on a planar compositing surface [1].

by the fact that there might be alignment errors, parallax and scene motion, and the corresponding pixels across images may not correspond to the same 3D point in the scene. I discuss blending issues and approaches in detail in Section 2.2.2. In Chapter 3, I introduce my dynamic blending approach which avoids stitching artifacts while recreating some of the dynamic aspects of the scene.

2.2.1 Compositing Surface

Given the alignment transforms, all images need to be warped and placed onto a common compositing surface. There are two objectives when choosing a compositing surface.

- All images are geometrically warped to align them on the compositing surface. Appropriate compositing surface geometry ensures that the distortion of individual images generated by warping is minimal and they are uniformly sampled.

- Compositing on a non-planar surface yields a projection in which scene structures may appear distorted, e.g., lines in the scene may get bent. Compositing surface choice should minimize such distortions.

Usually there is a trade-off between these two objectives. I discuss common compositing surface choices used in prior work below.

Planar projection: A planar compositing surface preserves perspective projection (keeps straight lines in the scene straight in the composite) and works well for rotational panorama when the field of view is not very large or for a translational panorama [1], i.e., images of a planar scene captured by a translating camera (Figure 2.1).

However, a plane is not a good choice for wide field of view rotational panoramas as it leads to geometric distortion of the images on the periphery and breaks down for fields of view larger than 180° (Figure 2.2).

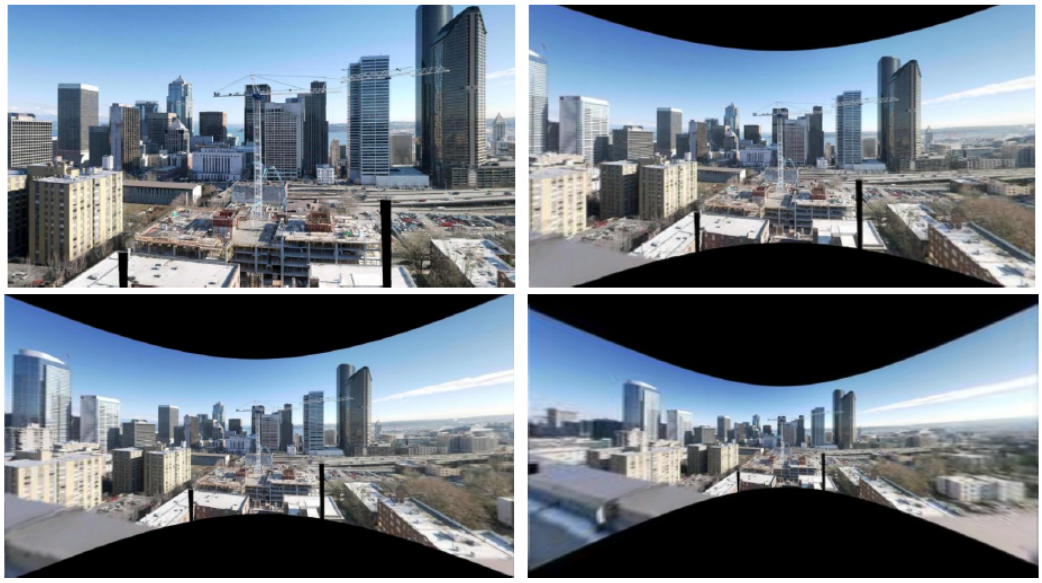


Figure 2.2: Planar projection with different field of view for a rotational panorama. It works well for smaller field of view (top row) but shows distortion near the periphery for larger field of view (bottom row), i.e., the images at the periphery are poorly sampled. Images from [52].

Cylindrical and spherical projection: The most common use-case of a panoramic image is to capture a 360° field of view from a specific point by taking images looking in different directions. For such rotational panoramas, curved surfaces like a cylinder or a sphere (Chen [18], Szeliski [97], Szeliski and Shum [99]) work well. However projecting on to a curved surface introduces unnatural distortion of the scene – e.g., straight

Figure 2.3: Cylindrical projection creates distortion (straight scene lines appear curved). Image from [51].

lines in the scene become curved (Figure 2.3). Zelnik-Manor *et al.* [107], Kopf *et al.* [51] and Carroll *et al.* [15] address this issue by using an interactive method which requires the user to specify constraints based on the scene, e.g., manually specifying straight lines in the scene which need to be kept straight, and the compositing surface is modified accordingly.

Adaptive projections: Peleg *et al.* [71] use an adaptive compositing surface based on the idea that after projecting onto the compositing surface, the optical flow vectors should be identical in magnitude and direction. This approach enables generating mosaics in many general cases including a forward moving camera. However, their approach relies on optical flow computation and is only applicable for videos or densely sampled imagery. Further, since they may end up using arbitrary manifolds for stitching, the composite image may show arbitrary distortions. Nomura *et al.* [67] and Zelnik-Manor *et al.* [106] use only similarity transforms to stitch images into Hockney-style [45] panoramas. While they keep the individual images undistorted, there are significant stitching errors at seams due to the constrained stitching model.

Lin *et al.* [56] use a spatially varying affine transform to stitch images that allows them to stitch images with parallax and motion. While being more flexible than homography based stitching, it can introduce unnatural distortion of the scene as well, e.g., lines may bend.

2.2.2 Blending

After projecting the input images on to a compositing surface, one needs to decide the color of each pixel in the final stitched image. A pixel in the stitched image, which maps to a point on the compositing surface, may be contained in multiple input images. If there is perfect alignment, all images have been exposed identically and have identical color balance, and there are no view dependent effects, then a pixel would look the same in all of the source images and one may choose any one of them. However, that is rarely the case. I discuss below different problems that occur.

Misalignment, parallax and scene motion: Due to these effects, corresponding pixels in different source images after alignment may not correspond to the same 3D point in the scene. A simple way to resolve this problem is to choose a single source image for each pixel in the final composite. This can be achieved by doing Voronoi tessellation of the stitched image which assigns each pixel to the nearest image in the set (Wood *et al.* [105], Peleg *et al.* [71]). However Voronoi tessellation can create discontinuities across *seams*, i.e., boundaries where the source image changes. Agarwala *et al.* [2] describe a seam selection algorithm by posing the source image selection problem as a Markov Random Field over pixels that selects seams which minimize the perceptual difference across them and is based on Graph Cuts [10]. Eden *et al.* [27] build upon this approach and come up with a two step approach that works well even in presence of exposure differences.

Exposure differences, different color balances: Even if the pixels align, they may have different colors in different source images. Feathering (linear blend) across image boundaries is a simple solution that works well even in the presence of slight

misalignments (Szeliski and Shum [99], Chen and Klette [17], Uyttendaele *et al.* [103]). While simple feathering may lead to blurring near seams, Laplacian pyramid blending [14] better preserves high frequency details across images.

Recently, gradient domain approaches have been very successful in image blending. Gradient domain blending was introduced by Perez *et al.* [73] for the two image case and later generalized to the multi image case by Agarwala *et al.* [2]. The approach works by computing the gradient field of the desired image and then reconstructing an image whose gradient fits the computed field. Levin *et al.* [54] provide a good survey of gradient domain image stitching techniques.

Uyttendaele *et al.* [103] explicitly compensate for exposure differences by dividing the image composite into blocks and fitting a quadratic transfer function between the original source image and the image composite corresponding to each block while ensuring that the transfer function changes smoothly over blocks. While being able to handle extreme exposure changes, such a block based approach can also handle spatially varying effects like lens vignetting.

Mosaics often span a large field of view which may have a high dynamic range, e.g., in a mosaic of a room with bright windows, it is difficult to capture both the indoors and outdoors in a single exposure. High quality mosaics can be created by capturing multiple exposures corresponding to each part of the scene. In such cases, one can first estimate a *high dynamic range* (HDR) radiance map and then apply *tone mapping* to reduce it to 8-bit range. Estimating the high dynamic radiance map typically involves estimating a parameterized radiometric transfer function along with the radiance map [61, 25, 65, 82]. Tone mapping HDR images into 8-bit images is a well studied problem and many approaches exist [31, 26, 83, 58].

2.2.3 Summary: Compositing Surface and Blending

While image mosaicing is a fairly mature field, mosaicing has been largely seen as a way to generate ultra wide-angle or ultra high-resolution images. In particular, most

approaches make the assumption of a rotating camera when choosing the compositing surface. Similarly, in case of blending across images, the goal is to generate a single static seamless mosaic. Hence, these approaches do not seek to capture and recreate the dynamic aspects of the scene like varying brightness and motion in the scene.

Further, these approaches work with photo collections that have been captured with the intention of creating a mosaic. In unstructured collections, there might be significant parallax preventing perfect alignment of images. Moreover, if the images are captured by different cameras and under very different conditions, there might be extreme appearance changes across images and traditional blending approaches may fail to produce a seamless stitch.

2.3 Image Mosaics for Interactive Scene Visualization

There exist approaches which use mosaics as building blocks for interactive scene visualization systems. These approaches come under the larger category of Image Based Rendering (IBR), where the goal is to generate a rendering of a scene from images. A detailed survey of IBR techniques can be found in [90].

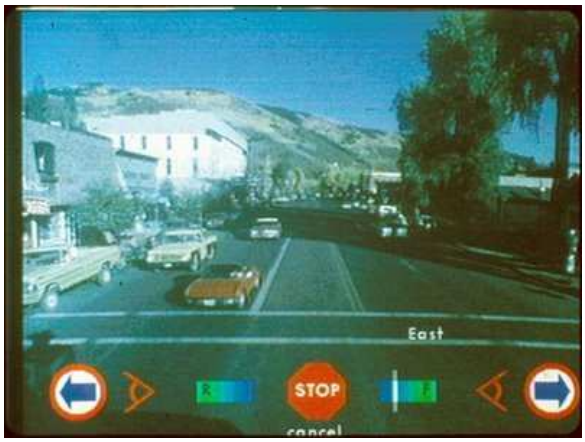


Figure 2.4: Andrew Lippman's Aspen Movie Map Project [57]. The user can navigate city by going forward and backward along the streets and branching off at intersections.

Andrew Lippman's Aspen Movie Map project [57] provides one of the earliest example of creating an interactive 3D walk-through from images. The system enables the user to take a virtual tour of the city of Aspen, Colorado. The streets of Aspen city were filmed at 10 foot intervals and the captured frames are then played in an interactive viewer (Figure 2.4). Building this system was an intensive task requiring a lot of data capture, storage and manual processing.

Systems like Google Street View [5] and Bing Maps [62] can be seen as modern versions of the Aspen project that seek to scale this approach up to the entire world. In these systems, 360° panoramas are captured at every few meters. The user can then freely navigate inside each of these panoramas or jump to adjacent panoramas. While the navigation is smooth and intuitive within a single panorama, the discrete jump from one panorama to the next may be disorienting to the user. Further, these are still captured by specialized cameras mounted on a moving van; it may be difficult to extend the method to images captured by a handheld camera.

One of the first IBR systems to use image mosaics was Quicktime VR [18]. The system stitches the images taken from the same spot into a cylindrical panoramic image which can then be viewed using pan and zoom interaction. Besides cylindrical panoramas, the system also supports cubic and spherical panoramas. The individual panoramas called as *nodes* can be manually linked with other *nodes* through *hot links*. The user can then click on these hot links to navigate the scene. The system also supports a special mode called *object movies* which allows the user to interactively rotate an object by simply dragging the mouse left and right. While data collection was much simpler in this system since capturing cylindrical panoramas is relatively easy, the individual panoramas need to be linked manually. Moreover, as mentioned before, discrete transitions between different mosaics can be disorienting to the user.

There exists a plethora of IBR approaches. However, in many cases, they require elaborate equipment for capture. E.g., Light-field rendering [55] and Lumigraph [40] can render a scene from any viewpoint but require a $2D$ array of cameras (or a single

moving camera) for capture. Among approaches that work with photos captured using hand-held camera, some need dense correspondences between images [19, 88] which is difficult to obtain even using state of the art computer vision methods. The Façade system [24, 23] to render architectural scenes relies on user input to build a 3D model. Another notable approach is the PhotoTourism system of Snavely *et al.* [94] which is a fully automatic system and works with unstructured photo collections downloaded from the Internet. However, it provides photo-centric navigation controls which work well for a photo browser but may not be optimal for exploring a scene. I discuss PhotoTourism system in more detail in Chapter 4 where I discover mosaics in the collection and augment the system with more intuitive controls.

The advantage of using mosaics for scene navigation is that they allow easy capture and intuitive navigation controls. However, current IBR systems simply use mosaics as building blocks with discrete transitions between them which may be disorienting when exploring a scene. My work on extending mosaics to more unstructured collection provides seamless navigation for more complex scenes.

2.4 Image Mosaics with Dynamic Aspects

Most image mosaicing methods seek to generate a single static image composite. E.g., in presence of scene motion, mosaicing methods seek to find optimal boundaries between images so as to avoid any stitching artifacts. However, if the goal is to create a virtual experience of the scene, motion may form an integral part of the experience, e.g., crowd moving about, trees waving in the wind, etc.

Rav-Acha *et al.* [81] propose Dynamosaics which are mosaics with motion created from slowly panning videos of scenes with repetitive motion, e.g., a waterfall. The approach works by stabilizing the video frames in time and then considering slices of the resulting space-time volume as mosaics. Agarwala *et al.* [3] use a different approach to generate similar panoramas from video. Their method is based upon video textures [87] where a video is analyzed for similar frames which are then used

to generate a new seamless video. To generate a video panorama from a panning video, the user first manually tags regions of the scene as static or dynamic and then the video texture approach is applied to dynamic regions of the mosaic resulting in a video panorama that can be played indefinitely in a loop. Correa and Ma [20] also achieve similar results by segmenting out the moving pixels in the scene and then blending them over a mosaic composited from static background scene. However these are again essentially wide angle images with motion, i.e., the user cannot navigate and move about in the scene beyond panning.

Kopf *et al.* [52] propose a system for capturing and viewing gigapixel images. Their viewer also adapts dynamically. First, they use an adaptive compositing surface which curves into a cylinder as the user zooms out to show a larger field of view but flattens out to a planar surface upon zooming in so as to keep the lines in the scene straight. Second, they use a dynamic tone mapping approach which adjusts according to the local part of the mosaic the user is viewing. Another system that uses dynamic adaptive projection is the Street Slide system [50] which works with street level imagery acquired in Bing Maps [62]. The system selects vertical strips from different images taken along a street and then generates a composite by simply placing them alongside each other. The width of the strips is changed dynamically as the user zooms or translates left and right which results in parallax effects.

2.5 Summary

As we saw in Sections 2.1 and 2.2, image mosaics are largely seen as a way to capture ultra wide angle images. IBR systems that do use image mosaics consist of rotational mosaics captured at different locations in the scene connected through discrete transitions between them which may be disorienting to the user (Section 2.3). In Chapter 3, I extend image mosaics to more general camera motion that enables visualization of more complex scenes but with the intuitive and seamless navigation that mosaics provide. Further, I use piecewise-perspective projection that preserves

straight lines in the scene. I achieve this by moving away from the paradigm of generating a single static mosaic and instead building a dynamic viewer which updates the compositing surface as the user navigates. This dynamic approach also enables a novel solution to the blending problem, which not only avoids stitching artifacts but also recreates dynamic aspects of the scene like varying brightness and motion in the scene along the lines of approaches discussed in Section 2.4.

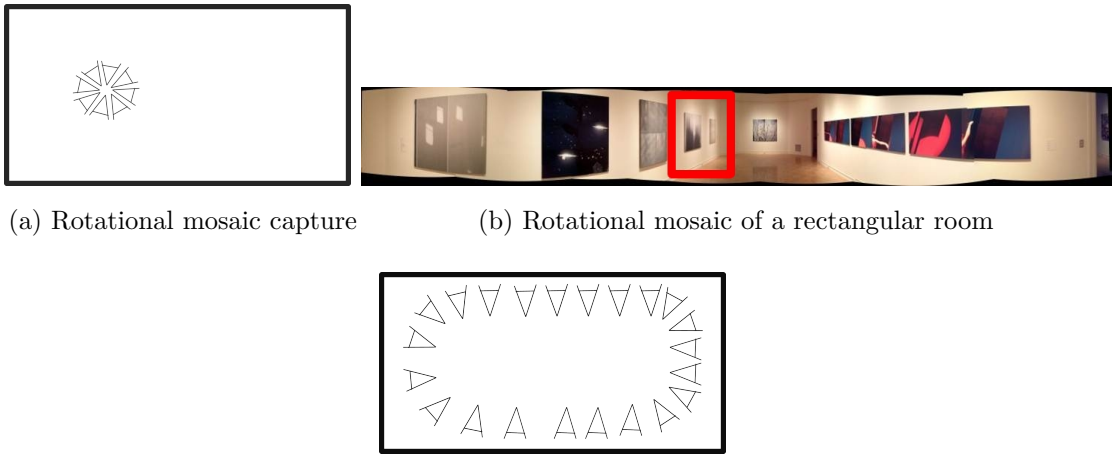
Past mosaics approaches have been targeted towards structured collections that have been captured with the intention of creating a mosaic. While there exist IBR systems that leverage the vast amount of Internet imagery for scene visualization, existing mosaicing approaches cannot be readily applied to such unstructured collections. I introduce an algorithm to discover image mosaics in such collections and aid visualization of the scene (Chapter 4). Further, past approaches to compensate for appearance difference across photos fail in the presence of the extreme variation seen in Internet photo collections. In Chapter 5, I model the space of different possible appearances of such unstructured photo collections and show, both theoretically and empirically, that any photo of a scene can be expressed as a linear combination of set of basis photos. I then describe an algorithm to recover basis photos from Internet photo collections and show applications of the same.

Chapter 3
DYNAMIC MOSAICS

Let us consider the case of a rectangular art gallery with paintings along the wall. Presently, the best way to create a mosaic in such a case is to capture a rotational panorama from the center of the room. However, since the room is rectangular, a single rotational mosaic will not be able to capture all the paintings uniformly; paintings at the corner of the room appear highly distorted an have low resolution (Figures 3.1a and 3.1b). One can capture the individual walls as separate translational mosaics where each painting has a head-on view, but they cannot be stitched automatically into a single seamless mosaic capturing the whole room. Ideally, one will like to capture the walls by moving the camera as shown in Figure 3.1c where we have a head-on view of each painting and the stitch is seamless. However such a collection of images can not be automatically stitched into a single mosaic (without bending straight lines in the scene) using current techniques.

Recent work on non-perspective panoramas enables more general camera motions [77, 71, 56], at the expense of introducing distortions (e.g., straight lines in the scene become curved), by relaxing perspective-correctness. Rather than introducing scene distortion, I propose to relax the requirement that all photos be composited into a *single* mosaic. I compute a collection of local mosaics that can be interactively traversed in a new kind of dynamic scene viewer. Further, each of these local mosaics preserve straight lines in the scene. The key aspects of my approach are:

- **Dynamic Projection:** The viewer uses a dynamic projection model to transition between local mosaics. Transitions are so subtle as to be barely noticeable, resulting in a continuous and nearly distortion-free scene visualization. The



(a) Rotational mosaic capture

(b) Rotational mosaic of a rectangular room

(c) Desired capture

Figure 3.1: (a) A rectangular art gallery with paintings along the wall captured using a single rotational mosaic (b) A rotational mosaic captured inside the rectangular room. Paintings at the further end of the room appear distorted and have a low resolution – it is impossible to uniformly capture all the paintings along the wall with a rotational mosaic. (c) On the other hand, a collection of photos taken along the walls of a room may capture the room more uniformly. Such a collection ends up being a combination of translational and rotational mosaics but cannot be stitched into a single mosaic using existing methods.

approach generalizes to a class of image collections which I call *locally stitchable*, and subsumes rotational and translational mosaics.

- **Dynamic Seam Selection and Blending:** Misalignment between images, scene motion and parallax lead to ghosting artifacts in traditional mosaicing. The proposed viewer dynamically selects and blends across a set of precomputed seams to avoid artifacts and allows us to recreate dynamic aspects of the scene, e.g., people walking, trees waving, etc. as the user navigates.

- **Dynamic Exposure Selection:** Rather than removing exposure differences entirely, the viewer dynamically adjusts the exposure based on the part of the scene currently visible. This imitates the behavior of the human eye, e.g., when

looking at a bright light source, rest of the scene appears dark.

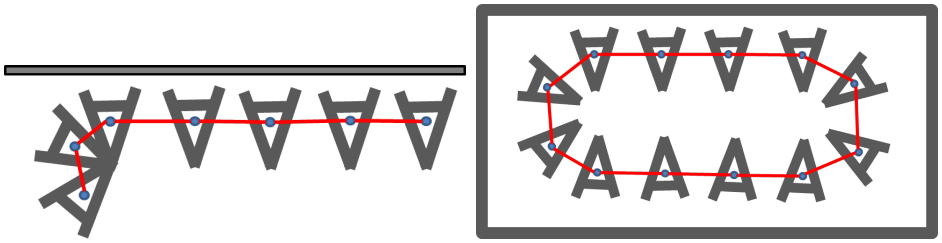
I call such mosaics *dynamic mosaics.* I refer the reader to the video results [84] to see what the output looks like – results are discussed in more detail in Section 3.4.

There are three major challenges that I address. First, I pose the problem of computing a local mosaic as an optimization problem which minimizes the distortion in the current view and can be solved in real time. Secondly, blending techniques (e.g., graph cuts [10], multi band blending [14]) are not fast enough to work in real-time. I come up with a semi-online approach that allows us to do blending in real time while recreating scene motion. Finally, I come up with a method to dynamically compute locally optimal exposure.

Mosaics are a simple form of Image Based Rendering (IBR), a class of techniques that includes more sophisticated methods like lightfields [55, 40], plenoptic modeling [64], Photo Tourism [94], and so forth. A key difference, however, is that mosaics rely only on 2D warps (homographies) and 2D navigation (pan, zoom). This reliance on 2D warps makes mosaics robust enough to work in practice, and the simple 2D navigation appeals to the most naive users. These factors have enabled mosaics to enjoy a level of penetration and widespread use that other IBR methods have not. My work seeks to achieve a greater range of behaviors (generalized camera paths, scene motion, dynamic exposure) than is possible with existing mosaic techniques, while staying in the mosaic framework (2D warps, 2D navigation).

3.1 Overview

Let us first define *locally stitchable image collection*s. I consider two images as *stitchable* if they overlap and can be aligned via a 2D transform, i.e., a 3×3 homography matrix. E.g., images captured by a rotating camera or images of a planar scene are *stitchable*. Consider a graph with images as nodes and an edge between every pair of stitchable images. I call an image collection *locally stitchable* if the resulting graph is connected



(a) (b)

Figure 3.2: Locally Stitchable Collections. (a) A locally stitchable collection that is a combination of translational and rotational motion. (b) Adjacent images need to be only approximately stitchable. This shows an example of a rectangular room whose walls have been captured by a combination of translational and approximately rotational motion. A single rotational mosaic will not provide a uniform sampling of the walls.

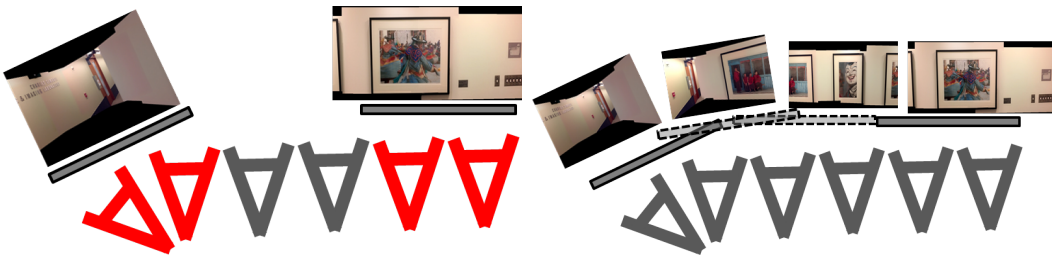
(Fig. 3.2a).

One can relax the definition of stitchablity. E.g., two images can be aligned well if they are taken from *approximately* the same spot or if the scene is *approximately* planar. This allows us to extend the definition to cases like those shown in Figure 3.2b. Any collection which consists of sequences of rotating camera or translating camera (in front of a planar scene) is included in the definition.

I now briefly describe the three key dynamic aspects of my system.

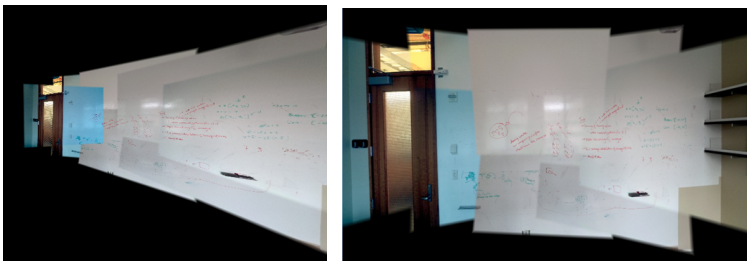
3.1.1 Dynamic Projection

Consider the photo collection shown in Figure 3.3a, a combination of rotational and translational motion. Photos at the left end of the collection can be composited into a rotational mosaic. Similarly photos at the right end can be composited into a translational mosaic. The proposed viewer interpolates between the two mosaics by smoothly varying the projection as the user navigates from one end to the other. This is achieved by selecting a planar projection based on the current view which changes smoothly as the user moves around (Fig. 3.3b).



(a) (b)

Figure 3.3: (a) Photos at either end of the collection can be stitched into local mosaics by projecting them onto different planar surface. (b) My system smoothly interpolates the planar projection as the user navigates allowing smooth transitions between these local mosaics.



(a) (b)

Figure 3.4: (a) An arbitrary planar projection. (b) Optimized planar projection.

How should one select the optimal planar projection for a given view? An arbitrary planar projection may lead to geometric distortion (Fig. 3.4a).

I formulate the planar projection selection problem as an optimization problem that seeks to minimize the geometric distortion of images in view (Fig. 3.4b). A planar projection is represented by a 3×3 homography matrix H. We need to quantify the geometric distortion induced by a homography H. For H to not introduce any affine or perspective distortion, it should preserve the original rectangular shape of the input image. While this can be measured in a number of ways, one way is to see where the horizontal direction $[1, 0, 0]$ and vertical direction $[0, 1, 0]$ are mapped by H. Hence, a measure of distortion can be $||H[1, 0, 0]^T - [1, 0, 0]^T||^2 + ||H[0, 1, 0]^T - [0, 1, 0]^T||^2$ (This

also penalizes $2D$ rotation and uniform scaling which do not introduce any affine or perspective distortion but I talk about that later). Such a penalty can be written in the form $||H^{res} - I||^2$ where H^{res} is simply H where the last column has been replaced by $[0, 0, 1]^T$, i.e., the translation free part of H. This can also be interpreted algebraically – the distance between the translation free component of H and the identity transform. Such a penalty is also consistent with Zorin and Barr's [108] observation that for a linear perspective camera, objects in the center of the image never look distorted.

Given this measure of geometric distortion, the viewer chooses a projection that minimizes a weighted sum of distortions of the images in view. I show in Section 3.3.1 that such a penalty measure leads to a linear objective which can be solved and continuously updated at frame rate as the user navigates.

3.1.2 Dynamic Seam Selection and Blending

Finding optimal seams between images to avoid ghosting artifacts due to scene motion and parallax is a common approach [103, 22, 2, 1]. However, in our case we do not have a single mosaic for which we can precompute seams beforehand. As it is impractical to compute seams in real time for each local mosaic we render, I precompute seams for a set of local mosaics and then transition between those seams at render time based on user's viewpoint.

Such an approach has two advantages. First, the seams are *locally optimal,* i.e., they prefer images which show least distortion in the current view. Second, as I crossfade between different local seams as the user navigates, it recreates the dynamic aspects of the scene like parallax and scene motion. While the basic formulation computes seams whose purpose is to minimize the stitching artifacts, one can add additional constraints to explicitly control how seams are correlated with the motion in the scene (Sec. 3.4.1). Further, I also implement real time multi band blending across these seams (Sec. 3.3.4).

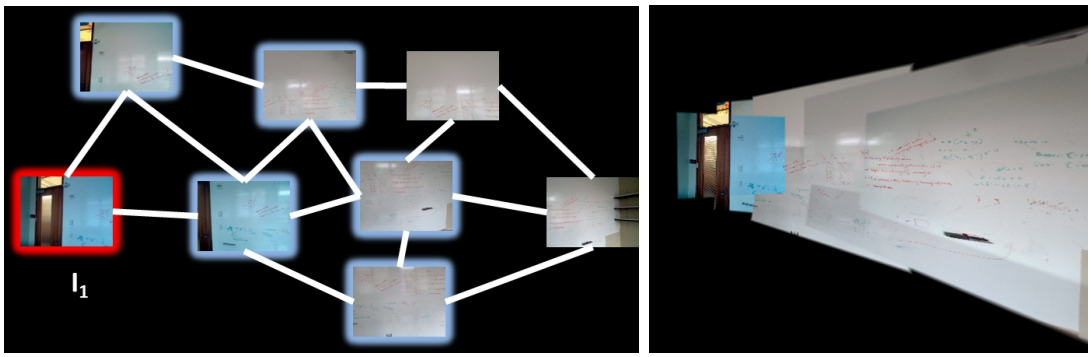
3.1.3 Dynamic Exposure

Cameras in auto-mode change exposure dynamically based on scene brightness. Exposure differences lead to artifacts when stitching a single static mosaic and sophisticated exposure adjustment and blending algorithms are required to avoid such artifacts [103, 1, 11, 52]. However, exposure differences across images provide valuable cues about variation in brightness of the different parts of the scene. The system estimates these differences in an offline process and uses them to dynamically adjust the exposure in the online viewer as the user navigates providing a more realistic experience.

Brown and Lowe [11] use a simple linear gain model, i.e., a multiplicative factor to model exposure changes which is equivalent to multiplying the color channels of an image by a single scalar. I relax this model by assuming that three independent scalars can be used for the three color channels. This allows me to correct for not only exposure changes but also color balance. This is equivalent to using a diagonal transform color correction model which suffices for many cases [32]. I compute these scalars using an approach similar to Brown and Lowe [11] (Section 3.2.4). This simple model in conjunction with seam selection and blending avoids almost all of the artifacts.

While rendering, Kopf *et al.* [52] use dynamic tone mapping of HDR images based on the local part of the mosaic the user is viewing. Analogously, I compute locally optimal diagonal transform as the user navigates. The transform is optimal in the sense that it tries to preserve the original exposure and color balance of the mosaic currently in view.

I now proceed to give technical details of the approach which can be broken down into two parts – offline processing and online viewer.



(a) (b)

Figure 3.5: (a) The neighbor set S_1 (images with blue border) corresponding to image I_1. It consists of images that can be projected onto the plane of I_1 to generate a local mosaic. (b) The local mosaic generated by compositing images in the set S_1 onto the plane of I_1.

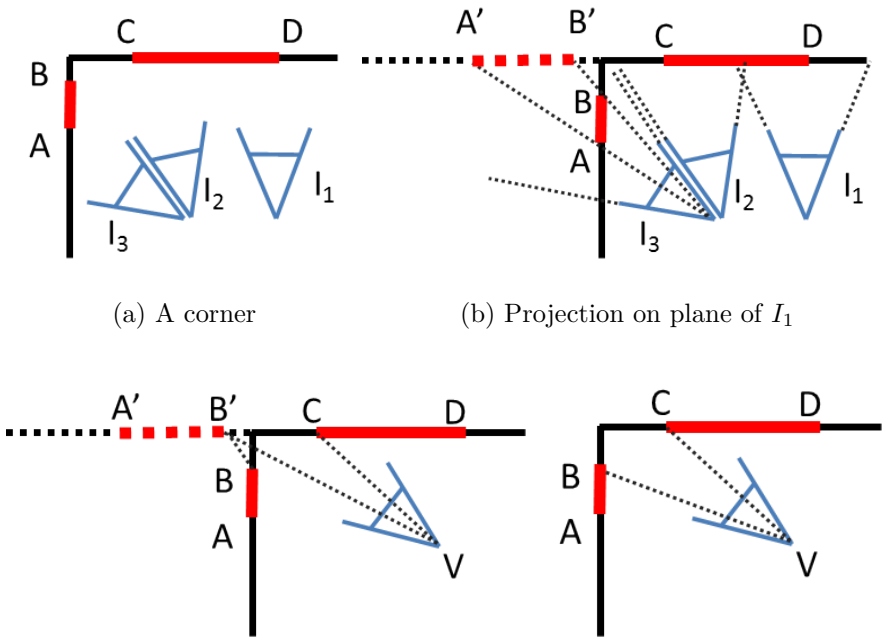
3.2 Offline Processing

3.2.1 Image Matching

I match images pairwise using SIFT features and RANSAC [43]. I declare two images to be *stitchable* if the homography found by RANSAC has at least 40 inliers. Since the graph of images is connected w.r.t. stitchability relation (locally stitchable), I can determine the homography between every pair of images by finding the shortest path between them and chaining homographies along the path. This may lead to accumulation of errors along long paths but images connected by long paths are not likely to overlap and hence such errors do not lead to stitching artifacts in practice.

3.2.2 Local Mosaics

For each image I_i, I define the set of neighbors S_i as images that can be stitched to I_i to generate a local mosaic (Figure 3.5a). Let us say that I_j is *compatible* with I_i if the optical axis of I_j intersects the plane of I_i in the positive direction, i.e., I_j can be



(a) A corner (b) Projection on plane of I_1

(c) Projection seen through virtual camera V (d) Single perspective view

Figure 3.6: A local mosaic is a piecewise perspective projection of the scene that preserves straight lines. (a) A corner captured by a locally stitchable collection. AB and CD are two straight lines on two different planes. (b) The local mosaic corresponding to S_1 obtained by projecting all images onto the plane of I_1. Line segment $A'B'$ is the projection of AB. (c) The local mosaic seen through a virtual camera V. V sees AB and CD as straight lines but under different perspective projections. (d) True single perspective view of the scene from V

projected onto the plane of I_i without flipping the image. Then, I define S_i as the largest connected set of images in the graph containing I_i which are compatible with I_i. Algorithmically, I run breadth first search starting from I_i to build the set S_i.

Images in the set S_i can be stitched to I_i to generate a local mosaic (Figure 3.5b). I prove that each such local mosaic is a piecewise-perspective projection of the scene that preserves straight lines.

Notice that if the images in the neighbor set S_i correspond to a rotating camera or if the scene is planar, then the resulting local mosaic is a single perspective projection.

Hence, the only interesting case is a non planar scene captured by a camera which is not purely rotating. Such a scene can be captured by a locally stitchable collection if a rotating camera covers the planar discontinuities in the scene (Figure 3.6a). Let AB and CD be two line segments in the scene on two different planes. The local mosaic corresponding to S_1 is generated by projecting all images onto the plane of I_1, thus projecting AB as $A'B'$ (Figure 3.6b). Now, while rendering, S_1 is viewed through a virtual camera V (Section 3.1.1) which sees the image $A'B'$ of AB instead of the true AB (Figure 3.6c and 3.6d). However, $A'B'$ is also a straight line. Hence, while the two planes of the corner appear under different perspective transforms in V, straight lines in the scene, which are confined to these planes, remain straight. The observation is generalizable as long as the homographies between *stitchable* images correspond to real world planes (holds if the image matching does not make errors). Zelnik-Manor *et al.* [107] also use such piecewise-perspective projections that are selected interactively to reduce distortion in spherical panoramas.

3.2.3 Optimal Seams via Graph Cuts

I precompute optimal seams for each of the local mosaics. At render time, as the user transitions between different local mosaics, I crossfade between corresponding seams.

Optimal seams should avoid stitching artifacts by minimizing pixel difference across them. Further, they should choose pixels from images which appear less distorted geometrically. I formulate an objective which captures these two goals and minimize it using Markov Random Field (MRF) optimization.

The optimization computes a labeling L over pixels in the local mosaic where $L(p) = j$ if pixel p of the mosaic is assigned to $I_j(p)$. This approach is similar to that of Agarwala *et al.* [1] but we use a different data term which encourages images with low distortion. Again, I use $||H_j^{res} - I||$ to quantify distortion where H_j is the homography that aligns I_j with I_i. I define the data term as

$$D(p, L(p)) = \begin{cases} \infty & \text{if } p \notin I_{L(p)} \\ ||H^{res}_{L(p)} - I||^2 & \text{otherwise} \end{cases} \tag{3.1}$$

where p is a pixel in the local mosaic and $p \notin I_{L(p)}$ implies that p projects outside the bounds of image $I_{L(p)}$. The second term encourages seamless transitions between images and has been used previously [2, 1, 53]

$$V(p, L(p), q, L(q)) = ||I_{L(p)}(p) - I_{L(q)}(p)||^2 + ||I_{L(p)}(q) - I_{L(q)}(q)||^2 \tag{3.2}$$

Note that $I_{L(p)}(p)$ refers to the pixel in $I_{L(p)}$that is the image of p in the mosaic. Further, we adjust the pixel values using the exposure compensation factors we compute in Section 3.2.4. The MRF optimization solves for a labeling that minimizes $\sum_p D(p, L(p)) + \lambda \sum_{p,q} V(p, L(p), q, L(q))$. We use $\lambda = 0.5$. Computed seams are stored as binary masks; for each $I_j \in S_i$, I have a mask M_{ij}. At run time, I do not need to store and load all the local mosaics, the system uses these single channel binary masks along with the input images to composite local panoramas on the fly.

3.2.4 Global Exposure Compensation

Brown and Lowe [11] first find pairwise gain factors for overlapping images by considering corresponding pixels, and then use pairwise factors to estimate per image gain factors by minimizing a global objective function. I use a similar approach to estimate per channel gain factors for each image. To estimate pairwise gain factors, I consider all corresponding pixels and use RANSAC to estimate the most common triplet of channel wise gain (r_{ij}, g_{ij}, b_{ij}). Such an approach is more robust than the mean used in Brown and Lowe's approach [11]. To estimate gain per image for each channel, e.g., r_i for red channel of I_i, I consider all pairwise equations $\frac{r_j}{r_i} = r_{ij}$ and solve using least squares for each color channel independently after taking log (and setting $r_1 = 1$). Brown and Lowe [11] use a slightly different objective which requires them to use a prior to encourage gains close to 1. Further, I weigh the equations by the number of

RANSAC inliers to reflect the confidence in each equation. At run time, I use the estimated values to compensate for exposure differences between images and compute a locally optimal exposure value as the user navigates (Sec. 3.3.3).

3.3 Online Viewer

3.3.1 Dynamic Projection

The user interacts with the mosaic using drag and zoom interaction. The viewer updates the projection to provide the best view of the images currently visible in real time as the user drags the mouse.

I first describe how dragging works. Since I use a planar projection at all times, I can model the transformation of each image by a homography. Suppose the current transforms applied to the images $I_1, I_2, ..., I_n$ are $H_1, H_2, ..., H_n$ respectively, i.e., these are the optimal homographies for the current view (initialization is done by setting $H_1 = I$ and using the pairwise homographies computed in Sec. 3.2.1), and the user drags by $(\delta x, \delta y)$. Let $H^{trans}(\delta x, \delta y)$ be the homography that induces a translation of $(\delta x, \delta y)$. I first apply this homography to the current view, i.e., the new transformations of images are $H^{trans}(\delta x, \delta y)H_1, ..., H^{trans}(\delta x, \delta y)H_n$. Let H_i refer to the updated transformation $H^{trans}(\delta x, \delta y)H_i$. Solving for a new planar projection is equivalent to solving for a homography H that will be applied to all the images, i.e., the geometric transforms of the images under this new projection will be $HH_1, HH_2, HH_3, ..., HH_n$. I want to choose an H that minimizes the distortion of images.

As described in Section 3.1.1, distortion of a single image in the collection can be measured by $||(HH_i)^{res} - I||$. However, the global homography H only seeks to minimize the distortion in the images and not translate the image composite. Hence $H(0,2) = H(1,2) = 0$, i.e., the last column of H is $[0,0,1]^T$ and H is already of the form H^{res} and the distortion can be written as $||HH_i^{res} - I||$. I compute importance weights for the images (explained later) and minimize the weighted sum of distortions.

Mathematically,

$$H_{opt} = \arg\min_{H} \sum_{i=0}^{n} w_i ||H H_i^{res} - I||^2 \tag{3.3}$$

However, it is a non linear optimization problem because of the dehomogenization operation, i.e., the bottom right entry of the product HH_i^{res} needs to be 1. However, algebraically, the above objective is similar to

$$H_{opt} = \arg\min_{H} \sum_{i=0}^{n} w_i ||H^{-1} - H_i^{res}||^2 \tag{3.4}$$

as they both encourage the product HH_i^{res} to be close to I. Since H_i^{res} are known and can be normalized by setting the bottom right element to 1, it yields a simple least squares objective and leads to a simple closed form solution

$$H_{opt}^{-1}(a,b) = \frac{\sum_{i=1}^{n} w_i H_i^{res}(a,b)}{\sum_{i=1}^{n} w_i} \tag{3.5}$$

H_{opt} can be computed by inverting the solution obtained from Eq. (3.5). In practice I avoid the explicit inversion and directly compute the products $H_{opt}H_i$ from H_{opt}^{-1} by solving the linear equation $H_{opt}^{-1}X = H_i$ which is more numerically stable. This computation is fast enough to be done in real-time on standard PCs. Note that H_{opt} does not induce any translation, i.e., the origin or the center of the mosaic remains stationary. However, it had moved exactly by$(\delta x, \delta y)$ before the application of H^{opt} due to application of $H^{trans}(\delta x, \delta y)$. This is the behavior that the user expects to see on dragging by $(\delta x, \delta y)$, i.e., the center of the mosaic moves by $(\delta x, \delta y)$.

I now describe how I compute importance weights for different images that are used in the optimization step. Distortion of images near the center of the viewport would be more noticeable. Hence I weight such images higher. Let (x_{i0}, y_{i0}) be the location of the center of I_i under transformation H_i. In all of our computations, I normalize the image dimensions to be $[-0.5, 0.5] \times [-0.5, 0.5]$. Our viewport is $[-1, 1] \times [-1, 1]$ with the origin at the center of the viewport. Then I define the weight of I_i as

$$w_i = \begin{cases} 0 & \text{if } |x_{i0}| > 0.5 \text{ or } |y_{i0}| > 0.5 \\ max(0.5 - |x_{i0}|, 0.5 - |y_{i0}|) & \text{otherwise} \end{cases} \tag{3.6}$$

Such a definition of weights ensures that they change smoothly as the user moves around ensuring that the optimal projection also changes smoothly. If the user is zoomed in very far, it might happen that all weights are zero. In that case, I set the weight of the image that is closest to the origin to be 1. I refer to the image with the highest weight as the *central image*. Any image that does not belong to the neighbor set S of the central image is set to have a weight zero. Finally the weights are normalized such that $\sum_i w_i = 1$.

Application of H_{opt} changes the transforms of the images and consequentially the weights. Ideally one should solve this in an iterative fashion until convergence. However, such an approach is too slow for interactive real time rendering. Hence, I instead amortize the iterations over render cycles, i.e., at each render cycle I compute the weights and solve for an optimal H_{opt}.

Zooming: Dragging interaction works by allowing the user to control the translation of the center of the mosaic while the optimization is invariant to it. Hence, zoom interaction should work similarly – allow the user to zoom in or zoom out with optimization being invariant to uniform scaling. However, the objective in Eq. (3.3) penalizes uniform scaling of images.

To make the objective invariant to scaling, I first compute the *scale factor* of each image. For that I need to associate a scaling factor $scale(H)$ with a general homography H. Consider a unit square centered at the origin transformed by the homography. I define the scale factor as the square root of the area of the transformed square. Since I normalize the image dimensions to $[-0.5, 0.5] \times [-0.5, 0.5]$, this measures the change in area of the image under that transformation. For our image collection, for every pair of stitchable images (I_i, I_j), I find this pairwise scaling factor $s_{ij} = scale(H_{ij})$ where H_{ij} is the homography that warps I_j to I_i. I then find the scale factor s_i for each image relative to I_1 in a way similar to how I find the exposure gain factors. More precisely, I set $s_1 = 1$ and solve for $s_{2,}s_{3,}..., s_n$ given the pairwise constraints $\frac{s_i}{s_j} = s_{ij}$. It reduces to a least squares system after applying log.

Now, when solving for the best planar projection in our objective, I normalize for the scale factor, i.e., we want the product HH_i^{res} to be close to

$$\begin{bmatrix} s_i & 0 & 0 \\ 0 & s_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

instead of identity. Correspondingly, I change the objective to

$$H_{opt} = \arg\min_{H} \sum_{i=0}^{n} w_i ||H^{-1} - H_i^{res} S_i^{-1}||^2 \tag{3.8}$$

where S_i is the matrix defined in Eq. (3.7) whose inverse is trivial to compute since it's a diagonal matrix. To allow the user to zoom in or zoom out of the mosaic, the viewer maintains a current scale factor s_f which is updated based on user input and I use

$$S_i = \begin{bmatrix} s_i s_f & 0 & 0 \\ 0 & s_i s_f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

in our objective.

To support images at different scales, I also change our weighing function – give more weight to the image that is at the right resolution for the current scale factor. Hence, I now define the weight as $w_i' = \frac{w_i}{1+|log(s_i s_f)|}$, i.e., amplify the weights of the images for which $s_i s_f$ is close to 1.

3.3.2 Cross Fades between Seams

The system uses the seams corresponding to the current *central image* (image with the maximum weight), i.e., if the current central image is I_i it uses the seams corresponding to S_i. As the central image changes, the system smoothly transition between different graph cut seams. The renderer works by keeping an alpha mask A_i [75] corresponding to each image I_i which are updated continuously. I initialize each A_i to be zero everywhere.

(a) (b) (c)

Figure 3.7: Binary masks from graph cuts. (a) An image I_j in the collection. (b) Binary mask M_{mj} for the image I_j corresponding to local mosaic S_m computed using graph cuts. (c) Binary mask overlaid on the image.

Let I_m be the current central image. Let M_{mj} denote the mask corresponding to image $I_j \in S_m$ (Figure 3.7). While the original mask computed using graph cuts is binary, it can have real values due to blending across seams (Section 3.3.4). Then at each render cycle, I update the current masks as

$$A_i(x,y) = \begin{cases} A_i(x,y) - \beta & \text{if } M_{mj}(x,y) < A_i(x,y) - \beta \\ A_i(x,y) + \beta & \text{if } M_{mj}(x,y) > A_i(x,y) + \beta \\ M_{mj}(x,y) & \text{otherwise} \end{cases} \tag{3.10}$$

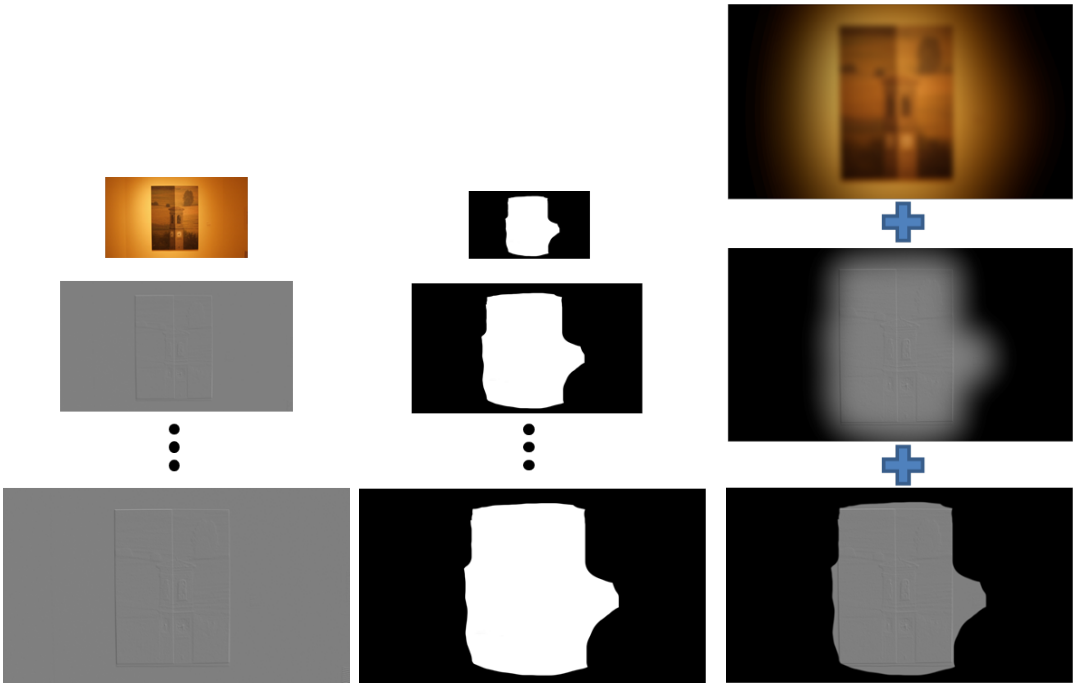
M_{mj}'s are loaded as textures and these updates are done in hardware using a custom shader. The parameter β controls the speed of crossfade.

3.3.3 Dynamic Global Exposure Compensation

I solve for scalar gain factors for each color channel independently. E.g., for the red channel, I use

$$r_0 = \arg\min_r \sum_i w_i (log\frac{r}{r_i})^2 \tag{3.11}$$

where r_i is the gain factor corresponding to I_i computed in Section 3.2.4, and similarly for the green and blue channels. The resulting exposure gain that is applied to I_i is $\frac{r_0}{r_i}$. From the objective, one can see that the gain corresponding to images with higher



(a) (b) (c)

Figure 3.8: (a) Laplacian pyramid corresponding to an image. (b) Corresponding pyramid for the binary mask. (c) Binary masks are applied to corresponding levels of the pyramid. Finally, we accumulate different levels of pyramids across images to generate the final rendering.

weights be closer to 1. The weights are the same as those used in Eq. (3.5). The above objective permits a closed form solution of the form

$$r_0 = \prod_i r_i^{w_i} \tag{3.12}$$

Further, I do not apply the compensation immediately and add a delay in a way similar to Kopf *et al.* [52] to simulate the behavior of human eye which takes some time to adjust to the change in lighting.

3.3.4 Real-time Multi Band Blending

I also implement real time multi band blending [14] across seams to remove any remaining artifacts (Figure 3.8). I precompute the Laplacian pyramid corresponding to each of the images and store different levels of the pyramid as individual textures. I add an offset of 255 to the difference images to ensure that all values are positive and then compress them to range $[0, 1]$ before loading them as textures. As higher levels of pyramid are stored at lower resolution, memory requirements are bounded.

At render time, for each I_i I build a Gaussian pyramid of the corresponding alpha mask A_i. This is done on the GPU using OpenGL's texture downsampling. Final rendering is then produced by accumulating the renderings corresponding to each level of the pyramid in accumulation buffer after adding back proper offsets. OpenGL's texture upsampling is used to upscale each rendering to the full resolution. Further, for each level, I also apply exposure compensation factors (Sec. 3.3.3) while adjusting for offset and compression.

Using too many pyramid levels may lead to quantization artifacts because of the limited resolution of accumulation buffer. Hence I use a two level scheme with a downsampling factor of 7 similar to Brown *et al.* [12]. Further, multi band blending is ineffective if the seams are very close to the image boundary as the blending alpha is truncated by image edges. Hence, I encourage seams that are away from the boundaries by penalizing pixels near the boundary in graph cuts and divide each pixel value in the final composite by the composited alpha value in case there is truncation and alpha value does not sum to 1.

3.4 Results

I show some results here while video results can be found online [84]. Please view the videos as it is difficult to capture the dynamics of the system in still figures. The renderer works at interactive rates on standard PCs.

(a) Camera path (b) Captured photos

Figure 3.9: A collection of 43 photos taken on a staircase capturing paintings on the wall.

Figure 3.10: Dynamic mosaic corresponding to the staircase collection. The top row shows a sequence of zoom outs corresponding to the part shown on the left. The bottom row shows zoomouts for another part. User can smoothly move between different parts simply by dragging the mouse.

Figure 3.9 shows a collection of 43 photos. Traditional image stitching software is unable to stitch them into a single mosaic. However, in my system the user can browse the scene by using simple drag and zoom interaction (Fig. 3.10).

(a) (b) (c)

Figure 3.11: (a) A collection of 31 photos of a street side. (b) In presence of parallax, stitching artifacts are unavoidable at image edges. (c) However, seam selection is able to mask such artifacts.

(a) (b)

(c)

Figure 3.12: A sequence of zoomouts from our viewer.

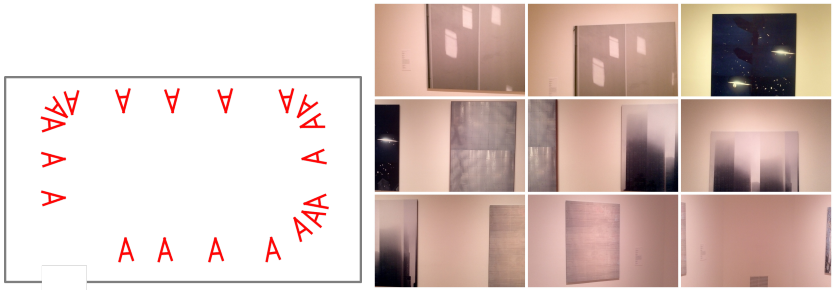
The system can also handle photos at different scales. The user is able to smoothly zoom in with the system choosing the images at the right resolution. The weighing function takes into account scale of the image and hence gives higher weight to images which are at the current scale of the viewer. Please view the video to see the results [84].

(a) (b)

Figure 3.13: Two renderings corresponding to the left end of Fig. 3.12 bottom. As the user drags to this part of the mosaic, the system smoothly updates the rotation to show as horizontal as possible view.

Figure 3.11a shows photos of a street side shot using a hand held camera. Because of significant parallax between images, I solve for a similarity transform between images instead of a full homography to reduce the degrees of freedom. Misalignments due to restricted matching model and parallax are minimal and are compensated by seam selection and blending (Fig. 3.11c). Fig. 3.12 shows the collection being browsed in our viewer. Unlike the system of Agarwala *et al.* [1], the camera motion is not restricted to translation; some of the taller buildings have been captured by tilting the camera up and down (Please see the video [84]). Further, dynamic seams emphasize parallax similar to the Street slide system of Kopf *et al.* [50].

Stitching such long facades can often lead to curved mosaics if the camera is not horizontal or not looking head on at the facade as can be seen on the left of Fig. 3.12 bottom. However, because I penalize in-plane rotation in the objective in Eq. (3.3), the viewer smoothly rotates the mosaic as the user navigates making the current view as horizontal as possible (Fig. 3.13).



(a) A rectangular art gallery (b) Some of the captured photos

Figure 3.14: A rectangular art gallery captured by a hand held cellphone camera.

Figure 3.15: The gallery being viewed inside the dynamic mosaic viewer

Figure 3.14 shows an example of 33 photos of a rectangular art gallery captured using a hand held cell phone camera and Figure 3.15 shows the gallery being viewed as a dynamic mosaic. Moving along the walls of the room gives uniform coverage of the paintings which a single rotational panorama cannot provide.

(a) (b) (c)

Figure 3.16: (a) 6 out of 18 input photos. (b) A rendering where the interior is well exposed while the window on the right is saturated. (c) As the windows come into view, a shorter exposure is selected; view outside the window is visible.

Figure 3.16 shows a rotational mosaic that adjusts dynamically to the varying brightness in the scene.

3.4.1 Dynamic Motion Mosaics

(a) (b) (c)

Figure 3.17: (a) 2 out of 18 photos from a rotating camera; the person moves between shots. (b),(c) Renderings from the viewer which stitches the images into a larger field of view. Dynamic seams recreate the motion without cutting through the person as the user pans from right to left.

Dynamic seams allow us to recreate the motion in the scene (Fig. 3.17). While my formulation automatically chooses seams to minimize artifacts, I allow the user to add additional constraints to achieve more targeted effects like the stroboscopic effect shown in Fig. 3.18c where the dynamic mosaic shows the history of the jump up to

the current time with time increasing as the user pans from left to right. This mosaic is created from a panning video where the location of the jumper was marked in each frame. This can be achieved by adding a constraint in local graph cuts. Assuming that the images are indexed by the frame number, then while computing the seams corresponding to S_i, I add a constraint to choose pixels corresponding to the jumper from I_j if $j \leq i$ and not choose such pixels if $j > i$. The method can also be seen as video summarization analogous to [16].

(a) Frames from a panning video (b) Masked location of jumper in frames

(c) Dynamic mosaic; jumper moves forward as the user pans from left to right.

Figure 3.18: Dynamic motion mosaic created from a panning video. The user specifies the location of jumper in the frames using a rectangular mask.

3.5 Discussion

I described an approach for creating dynamic mosaics of scenes from photos. These visualizations provide intuitive drag and zoom navigation like traditional mosaics while allowing for more general camera motion and being dynamic in nature. The new paradigm of dynamic mosaics allows one to capture a wider variety of scenes with more flexible camera motion and creates a more immersive experience of the scene by modeling the dynamic elements in the scene.

However, there is room for improvement. A more versatile appearance compensation model that models spatially varying effects like vignetting would be useful. Currently, homographies between images are computed in a greedy fashion, i.e., to compute homographies between a pair of images that don't overlap, I concatenate the homographies along the shortest path instead of ensuring that the computed homographies are consistent along all paths. Such an approach may not yield a globally optimal solution and may prevent *loop-closing* in certain cases, e.g., in a 360° panorama. Hence, an approach that computes globally optimal homographies will circumvent this problem. As I found out in my experiments, extending the bundle adjustment approach [101] to this problem yields an unstable optimization problem because of greater number of parameters in a homography as compared to 2 or 3 parameter model typically used in stitching a rotational mosaic.

Another desirable feature is to be able to show the summary of the entire collection if the user is zoomed out all the way. Since I use a planar projection, zooming out all the way may not be possible in all cases, e.g., for a 360° panorama. Image collages [106, 67] can lay out a collection on a planar surface at the expense of stitching artifacts. Peleg *et al.* [71] relax perspective-correctness and find adaptive manifolds for stitching mosaics from video sequences. However, it can result in arbitrary bending of scene lines and does not work for photos. Adaptive distortion of the stitching surface as the user zooms out is a direction worth exploring as future work.

While dynamic mosaics allow more general camera motion than possible with past approaches, a single dynamic mosaic cannot capture a very complex scene like an entire house. However, one can imagine a system where individual rooms are captured using dynamic mosaics with seamless transitions between them. Ultimately, the approach should allow easy capture and navigation of arbitrary scenes, e.g., a house with multiple rooms, where dynamic mosaics for individual rooms are connected to each other in an intuitive fashion that conveys the scene structure.

Chapter 4

MOSAICING FOR INTERNET PHOTO COLLECTIONS

In Chapter 3, I introduced dynamic mosaics that enable creating mosaics for more general photo collections than is possible with past mosaicing approaches. However, the approach is still restricted to photo collections that have been captured by a single user with the intention of creating a mosaic. How can one handle photo collections which are more unstructured, e.g., tourist photos of a scene downloaded from the Internet?

Creating scene visualizations from Internet photo collections has distinct advantages:

- The amount of imagery on the Internet has grown exponentially over the years. E.g., a search for a popular tourist place like *Pantheon* on Flickr [33] returns almost $200,000$ results (Figure 4.1). Developing mosaicing techniques for such photo collections will allow one to leverage this vast amount of imagery that already exists for a large number of scenes.

- The photos on the Internet have been captured under a wide variety of conditions, e.g., by different people, from different viewpoints, using different cameras, at different times of the day and under different weather conditions. Such collections capture the whole range of different possible appearances of the scene, which is hard to emulate if all photos are captured by a single person over a short period of time. Visualizations created from such collections are more diverse, displaying different possible appearances of the scene.

- Tourist photos are captured by different people and contain crowdsourced infor-

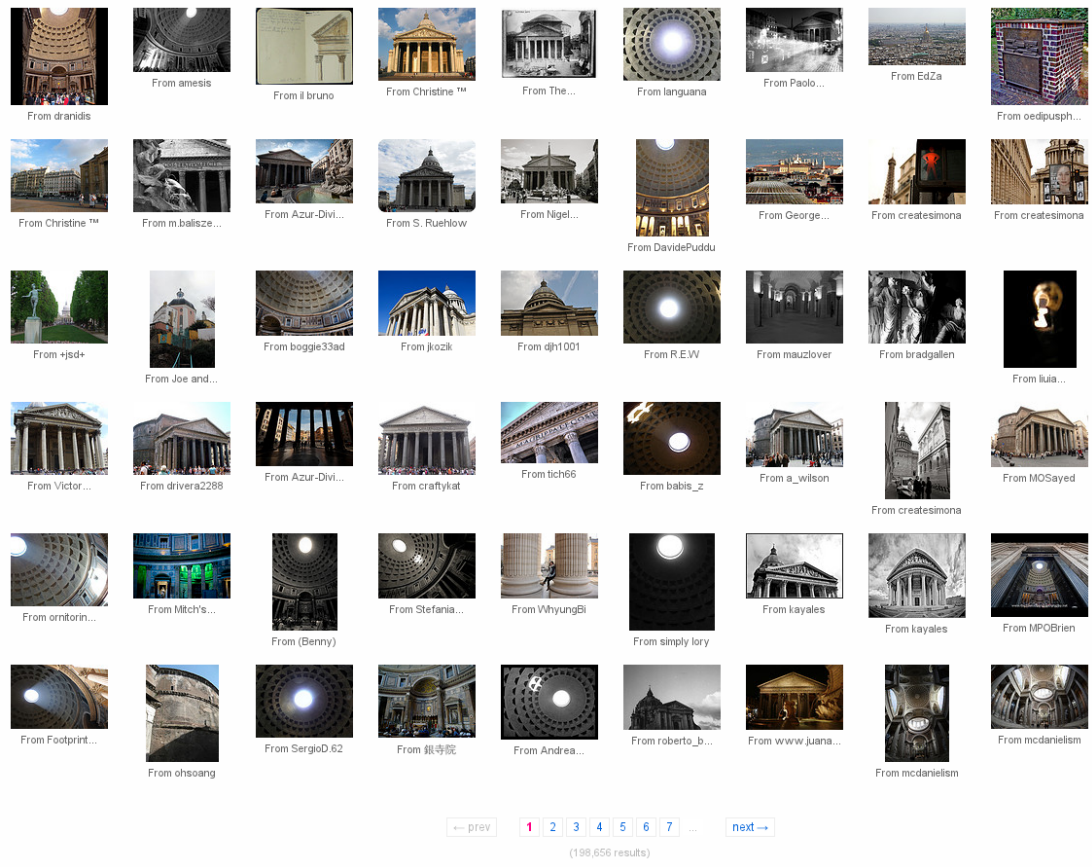


Figure 4.1: Flickr search for Pantheon. Almost 200,000 results are returned.

mation about which parts of the scene are interesting, e.g., a part of the scene that has been captured many times over indicates that a majority of people found the view worth photographing. Incorporating this information automatically into navigation controls will allow the user to easily discover interesting parts of the scene.

Given these advantages, how can one visualize a scene using Internet photo collections? Snavely *et al.* [94] introduced the PhotoTourism system, an interactive and immersive 3D photo browser for this purpose (Figure 4.2). The system takes as input a collection of photos, finds the 3D pose of each photo and then allows the user to navigate the

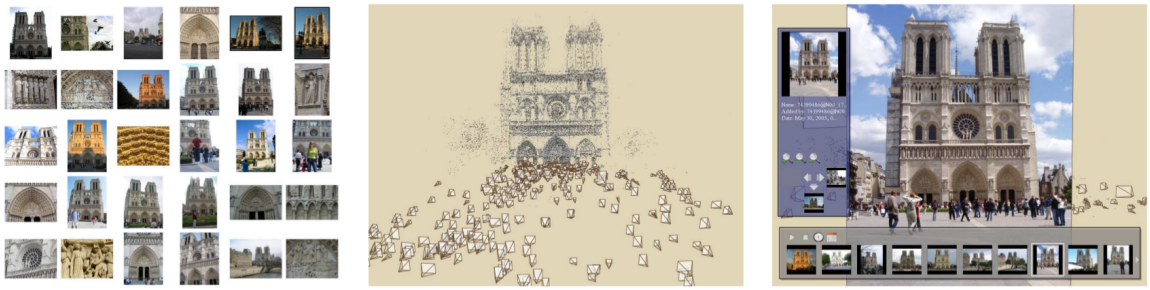


Figure 4.2: *The PhotoTourism system [94]* takes as input a collection of photos of a scene downloaded from the Internet. The system then recovers the 3D pose of each photo and a sparse model of the scene. Using this information the photos can be browsed in an interactive and immersive 3D browser. Figures from *[94]*.

photos in 3D. However, the system does not allow the user to move about freely in the 3D scene. Instead, the controls are photo-centric allowing the user to move from one photo to the next relying on a 3D transition between the photos to convey how they are arranged relative to each other in space. In particular, one can see photos to the left or right of the current photo. One can step back or step in to bring up a photo at a desired resolution. One can also drag a box around an object of interest and the system will select the best photo of that particular object in the scene. While it's true that 3D transitions between different photos convey how they are arranged in space, the user still moves from one photo to another in contrast to the seamless navigation that an image mosaic provides. Hence, while the system is general and allows arbitrary collection of photos and works well as an advanced photo browser with discrete navigation controls, it lacks continuous navigation and hence may not be ideal for exploring and navigating a scene.

In this chapter, I seek to automatically discover image mosaics in these collections that can then be used to augment the controls to aid scene navigation. There are two main challenges that need to be addressed. First, these collections are large, often containing thousands of photos. Hence, the approach needs to be scalable. Second,

these collections have not been captured with the intention of creating mosaics and are very diverse. Hence traditional stitching approaches may fail. E.g., if one wants to find rotational mosaics within these large collections, it's unlikely to find a group of photos taken from *exactly* the same spot.

I develop algorithms to identify two kinds of image mosaics in such collections – rotational mosaics and orbits. Rotational mosaics are similar to conventional mosaics captured by a rotating camera and correspond to viewpoints in 3D where there are photos looking in different directions. In addition, I also discover *orbits* which are motivated by the idea of object movies [18]. Such scenes often contain points or objects that are often photographed by people from different directions. E.g., Figure 4.3 shows an overhead view of the Statue of Liberty and recovered camera locations for photos downloaded from Flickr [33] are overlaid. Almost all cameras are looking at the statue and can be grouped into two distinct orbits, one closer to the statue and one further out into the water consisting of photos taken from boats. I seek to automatically detect such points of interests in the scene and associated orbits. These orbits can then be browsed seamlessly in a fashion similar to rotational mosaics – dragging the mouse left and right simulates moving along the orbit.

My work on extending image mosaics for Internet photos forms a part of the Pathfinder user interface described in [93]. While the PhotoTourism viewer was essentially a photo browser in many ways, Pathfinder seeks to provide controls that make the exploration of scenes easier. The system provides free viewpoint navigation allowing the user to move freely in 3D space. However, these photo collections often have a non uniform spatial distribution; there are parts of the scene which are photographed often and have a high density of photos while other parts have fewer photos. Free viewpoint navigation has 6 or 7 degrees of freedom and it may be difficult for the user to discover interesting parts of the scene, i.e., regions with high density of photos, using these controls directly. Hence, we augment the system with additional controls to guide the user to these regions – rotational mosaics, orbits and canonical

Figure 4.3: Overhead view of the Statue of Liberty with distribution of photos around it. Most photos are looking at the statue and these can be arranged into two distinct orbits – one near to the statue and the other one further out into the water.

images [91].[1]

4.1 Preliminaries and Notation

I assume that I have access to the scene reconstruction, i.e., 3D pose of each photo and a sparse reconstruction of the scene which can be computed automatically using the approach described in [94]. I refer to the database of images by $\mathcal{I}$, a single image by I and the associated camera by C. The camera intrinsics and extrinsics are known from the reconstruction.

Snavely *et al.* [93] introduce free viewpoint navigation in their Pathfinder interface where the user can freely control the virtual camera and the system selects the *best* photo to be shown to the user given the virtual camera. To select the best photo

[1]This is joint work with Noah Snavely and first appeared in [93]. My contribution was automatic discovery of rotational mosaics and orbits.

for a viewpoint v, the system first scores each photo by defining a reprojection score $S(I, v)$ for an image I and the viewpoint v which measures how well the image I can be rendered from viewpoint v. Then, the best photo is simply chosen as the photo with the highest score. This scoring function is described in Appendix A in detail and I use it in detecting orbits and rotational mosaics.

4.2 Orbits

An orbit consists of an object of interest at the center together with a number of photos looking at it from different directions. Since most tourist photos are taken by people standing on the ground or some other plane, I detect planar orbits whose plane is parallel to the ground plane (The ground plane can be estimated from the reconstruction of the scene as described in [92]). Further, I constrain myself to detecting circular orbits which can fit most real world situations.

4.2.1 Detection

Formally, I define an orbit to be a distribution of views positioned on a circle and converging on (looking at) a single point, denoted the convergence point p_{focus}. I constrain p_{focus} to lie on a vertical axis o passing through the center of the circle and perpendicular to the ground plane (since we detect planar orbits). The height of p_{focus} determines the tilt at which the object of interest is viewed. Because full 360 degrees view distributions are uncommon, I allow an orbit to occupy a circular arc. I define a good orbit to be one that satisfies the following objectives, as illustrated in Figure 4.4:

- **quality**: maximize the quality of rendered views everywhere along the arc.
- **length**: prefer arcs that span large angles.
- **convergence**: prefer views oriented towards the center of the orbit.
- **object-centered**: prefer orbits around solid objects (as opposed to empty space).

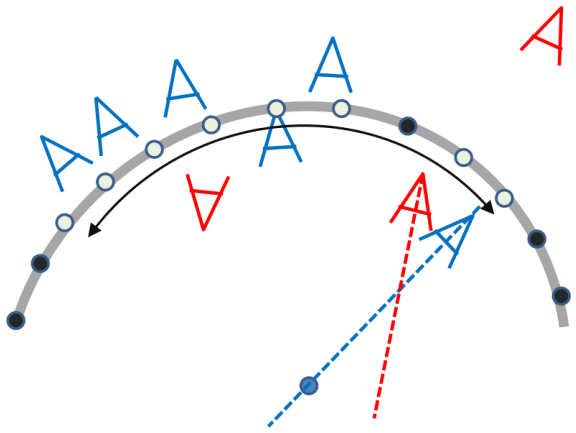


Figure 4.4: *Scoring a candidate orbit.* An orbit is evaluated by regularly sampling viewpoints along the candidate arc (here, the arc is shown in gray, and the samples are shown as small circles drawn on top of the arc). For each sample position, I want to find a nearby image with a high reprojection score that is oriented towards the orbit center (thus eliminating the red cameras). The light samples score well on these objectives while the black samples do not. I search for large arcs where the sum of the sample scores is high, and that do not contain large low-scoring gaps. Here, the optimal subset of the arc is shown with the curved black arrow.

Given these objectives, the problem of detecting orbits involves 1) defining a suitable objective function, 2) enumerating and scoring candidate orbits, and 3) choosing zero or more best-scoring candidates.

I first define my objective function for evaluating orbits. An orbit is fully specified by a center orbit axis o and an image I_j; the image defines the radius of the circle (the distance of the camera center from o), and the convergence point p_{focus} on the orbit axis (p_{focus} is the closest point on the axis o to the optical axis of camera C_j). Assume further that C_j is the point on the arc midway between the endpoints of the arc. The geometry of such an orbit is illustrated in Figure 4.5.

I define the orbit scoring function, $S_{orbit}(o, I_j)$, to be the sum of individual view scores, $S_{orbit}(o, I, \theta)$, sampled at positions θ along the arc. To compute $S_{orbit}(o, I_j, \theta)$

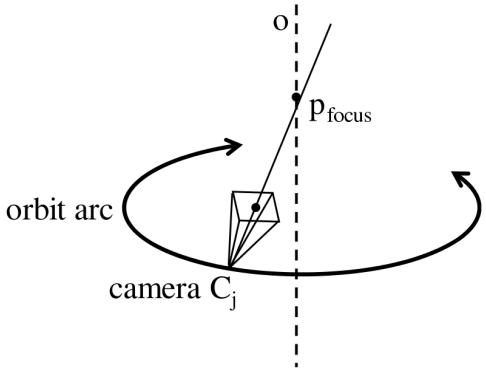


Figure 4.5: An orbit can be defined by an orbit axis and a camera. A vertical orbit axis orbit o, combined with a camera C_j (which defines the orbit radius), defines a family of orbit arcs around the axis. This family consists of all arcs of a circle centered on the axis which passes through the camera center (and for which the camera center is at the middle of the arc). The focal point p_{focus} is the point on the axis o closest to the ray through the center of C_j.

at a sample viewpoint $v(\theta)$ (the viewpoint on the arc at angle θ from I_j), I look for support for that view in the set of database images $\mathcal{I}$. In particular, I score each image $I_k \in \mathcal{I}$ based on (a) how well I_k can be used to synthesize a view at $v(\theta)$ (estimated using the reprojection score $S_{proj}(I_k, v(\theta))$ explained in Appendix A), and (b) whether I_k is looking at the orbit axis (the convergence score). $S_{orbit}(o, I_j, \theta)$ is then the score of the best image I_k at $v(\theta)$:

$$S_{orbit}(o, I, \theta) = \max_{I_k \in \mathcal{I}} \{S(I_k, v(\theta)) \cdot f_o(I_k)\}. \tag{4.1}$$

The convergence score $f_o(I_k)$ is defined as:

$$f_o(I_k) = \max(0, 1 - \frac{\psi}{\psi_{max}}) \tag{4.2}$$

where $\psi = \text{angle}(\mathbf{v}(C_k), p_{\text{focus}} - \mathbf{p}(C_k))$, i.e., the angle between the viewing direction $\mathbf{v}(C_k)$ of image I_k and the ray from the optical center $\mathbf{p}(C_k)$ of I_k to p_{focus} (we use a value of $\psi_{max} = 20°$). This term downweights images for which p_{focus} is not near the

center of the field of view.

I place a few additional constraints on the images I_k considered when computing $S_{\text{orbit}}(o, I, \theta)$:

- p_{focus} must be in the field of view of I_k.
- The tilt of I_k above the ground plane is less than 45° (orbits with large tilt angles do not produce attractive results)
- There are a sufficient number (we use $k_n = 100$) of 3D points from the sparse reconstruction of the scene visible to I_k whose distance from I_k is less than the orbit radius. I enforce this condition to ensure that I find orbits around an object (as opposed to empty space).

I compute $S_{orbit}(o, I_j, \theta)$ at every degree along the circle, i.e., $-180 < \theta \leq 180$. For simplicity, I refer to these sample scores as $s(\theta)$. A good orbit will have a long arc of relatively high values of $s(\theta)$. Simply summing the values $s(\theta)$, however, could favor orbits with a few sparsely scattered good scores. Instead, I explicitly find a long chain of uninterrupted good scores centered around I_j , then add up the scores on this chain. I define this chain as the longest consecutive interval $[-\theta_L, \theta_L]$ such that the maximum $s(\theta)$ in each subinterval of width 15° is at least $\epsilon = 0.01$. This definition allows for small "gaps," or intervals with low scores, in the arc. If this longest chain subtends an angle less than 60° , the score of the orbit is zero. Otherwise, the score is the sum of the individual scores in the chain:

$$S_{\text{orbit}}(o, I_j) = \sum_{k=-L}^{L} s(\theta_k) \tag{4.3}$$

Now that we have an objective function, we need a way to enumerate candidate orbits (mainly for sake of efficiency; we could score all possible orbits). My strategy for finding such orbits operates in two stages. First, I compute a set of good candidate orbit axes, by finding axes in 3D space that are the convergence points of many

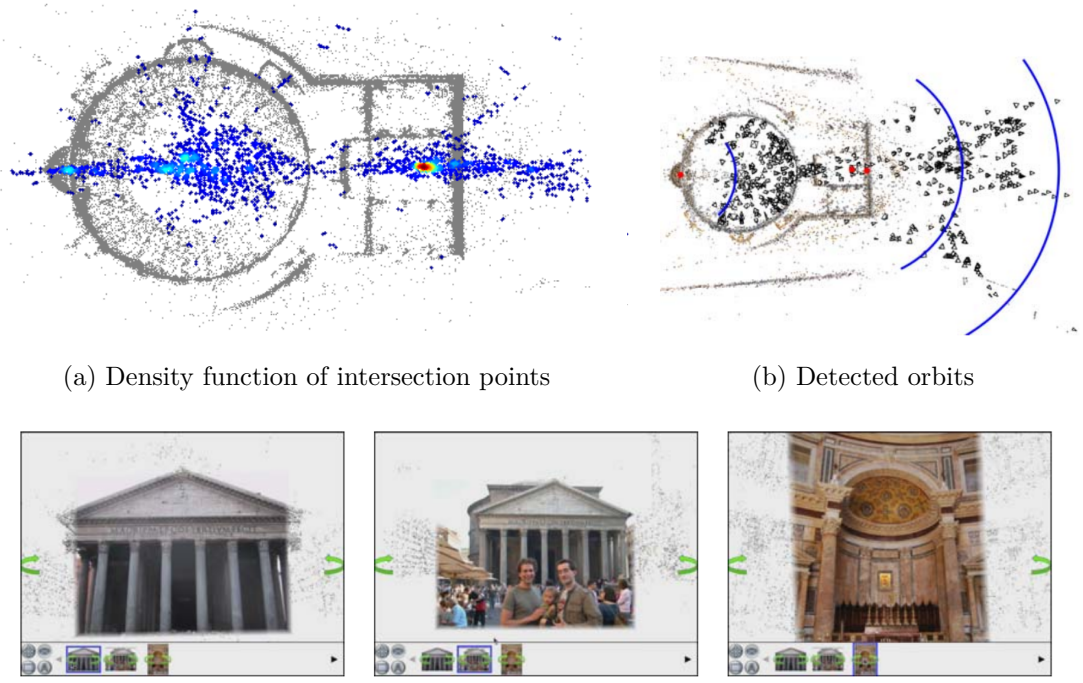
database images. Second, I create and score candidate orbits by pairing each candidate axis with each database image I_j.

To identify a set of candidate orbit axes, I take an approach similar to that of Epshtein *et al.* [28] and use the idea that an object of interest will often occupy the center of the field of view of an image. I first project all cameras onto the ground plane and consider the 2D intersections of the optical axes of all pairs of cameras (discarding intersection points which lie in back of either camera, or which are not approximately equidistant from both cameras). This projection onto the ground plane is possible since all axes are assumed to be vertical. I then compute the density D of these intersection points at each point x:

$$D(x) = \text{number of intersection points within distance } w \text{ of } x. \tag{4.4}$$

(I use $w = 0.02$, although this value should ideally depend on the scale of the scene). The local maxima in this density function identify vertical axes in the scene which are at the center of many different photos, and are thus potentially interesting. A plot of the density function for the Pantheon dataset along with three detected orbits is shown in Figure 4.6.

Next, I find the point with the highest density $D_{\max}$, then select all local maxima (points which have the highest density in a circle of radius w) which have a density at least $0.3D_{\max}$. These points form the set of candidate orbit axes. The next step is to find arcs centered at these axes. I form a set of candidate orbits by considering all pairings of orbit axes o and input images I_j . I only accept candidate orbits that satisfy the three constraints enumerated above, i.e., that the point of convergence is in the field of view, the tilt is less than 45° , and a sufficient number of points are visible in front of the orbit radius. I then evaluate $S_{orbit}(o, I_j)$ for each such suitable combination of orbit axis and image.



(a) Density function of intersection points

(b) Detected orbits

(c) Detected orbits in viewer

Figure 4.6: *Density function and detected orbits for the Pantheon data set.* (a) The set of intersection points of viewing axes is superimposed on the point cloud of the Pantheon. The color of each point corresponds to the density of points in its neighborhood (a red point has the highest density, a dark blue point the lowest). There are several clear maxima of this function, including a point just behind the front facade and a point at the altar (at the extreme left of the figure). (b) The three final detected orbits shown as blue arcs centered at the red orbit points. (c) Images from each of the three detected orbits.

I now have a set of orbits and a score for each orbit. To form a final set of orbits, I select the orbit with the highest score, remove it and all similar orbits from the set of candidates, then repeat, until no more orbits with a score of at least 0.5 times the maximum score remain. Two orbits are deemed similar if the area of intersection of the two circles defined by the orbits is at least 50% of their average area. An example of detected orbits for the Pantheon collection is shown in Figure 4.6. In this case,

three orbits were detected, two around the outer facade, at different distances, and one around the altar in the interior. Note that the furthest detected orbit arc is actually some distance behind the reconstructed cameras. The reason why this orbit does not pass closer to the camera centers is the constraint that it should not be too similar to an existing orbit (in this case, the inner orbit around the facade, which is the highest-scoring orbit). Furthermore, this orbit still has a high camera score, as translating a viewpoint backwards from an image along the viewing direction does not affect the angular deviation of rays nearly as much as translating the viewpoint sideways.

When computing viewpoint scores for orbit samples, I use a default vertical field of view of 50 degrees and a default screen resolution of 1024×768 (for the purposes of computing the field of view and resolution scores, see Appendix A for details).

4.2.2 Navigation

Discovered orbits are shown to the user as thumbnails in the bottom pane of the interactive viewer (Figure 4.7). The orbital motion is constrained to ring around the orbital axis. On selecting an orbit from the thumbnail panel, the axis is set to the axis of the discovered orbit and the virtual camera is set to be at a distance equal to the radius of the discovered orbit. Afterwards, the user can simply drag the mouse left and right to move along the orbit. Hence, only 1 degree of freedom is exposed to the user while in orbit, making it very easy to navigate. Further, the system chooses a proxy plane that passes through the orbit point and is perpendicular to the virtual camera axis. Such a plane stabilizes the object at the center of the orbit [93]. I refer the reader to [93] for more details on the rendering.

I detect orbits in a number of datasets which are mentioned in [93]. Two particular datasets which were found to be particularly suitable for orbits are the Statue of Liberty dataset (388 photos) and the Venus De Milo dataset (461 images). The system detects two orbits for the State of Liberty and one for Venus De Milo (Figure 4.7).

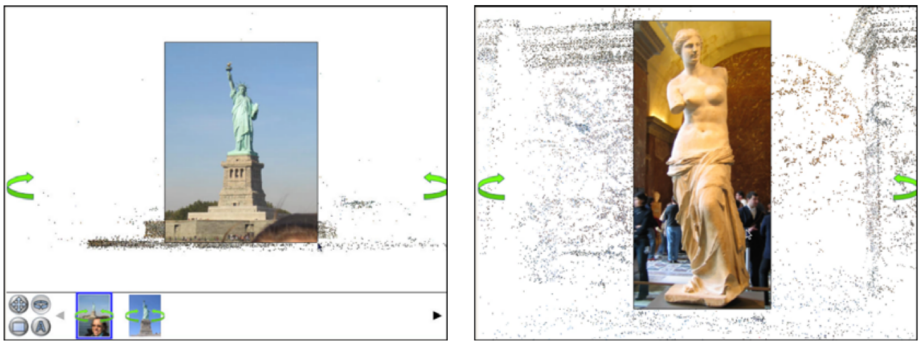


Figure 4.7: Detected orbits for the Statue of Liberty and the Venus De Milo datasets. Two orbits are detected for the Statue and one for the Venus. These can be accessed easily using the thumbnail panel at the bottom. Arrows on the sides indicate the direction in which the user can rotate to see more photos.

For the Statue of Liberty, one of the orbits captures the statue from up close whereas the other orbit is composed of photos taken from further out in the water from boats. The system also detects three orbits for the Pantheon dataset (602 photos) which is a complex scene consisting of both interior and exterior views (Figure 4.6). One of the detected orbits consists of a semi circular arc centered around the altar inside while the other two orbits consist of exterior views and are situated at different radii.

4.3 Rotational Panoramas

4.3.1 Detection

A rotational panorama, referred to as panorama in the subsequent text, consists of a set of images taken close to a nodal point. Similar to orbits, a good panorama has good views available from a wide range of directions. To find panoramas in a scene, I first consider each image I_j to be the center of a candidate panorama, and compute a panoramas score $S_{\text{pano}}(I_j)$ for each candidate. $S_{\text{pano}}(I_j)$ is computed as the sum of view scores S for a range of viewing directions around I_j:

$$S_{\text{pano}}(I_j) = \sum_{\phi=-5^\circ}^{25^\circ} \sum_{\theta=0^\circ}^{360^\circ} S_{view}(v(I_j, \theta, \phi)) \tag{4.5}$$

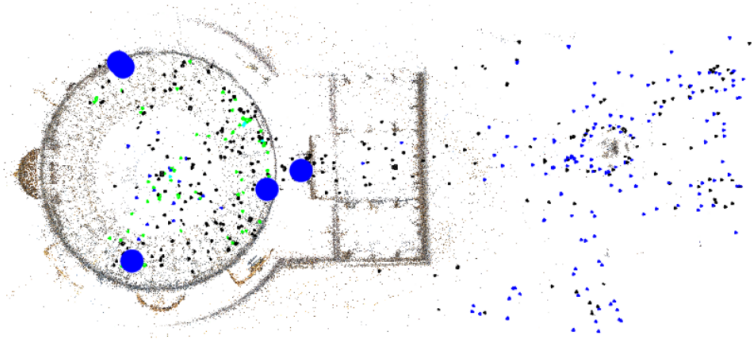


Figure 4.8: *Panoramas detected in the Pantheon dataset.* The panorama detector found five panoramas in the Pantheon collection, shown in this image as blue circles (the two panoramas near the top of the image are quite close to each other). For this scene, it is common for people to stand near the circumference of the interior and take photos looking across the building in different directions.

where $v(I_j, \theta, \phi)$ is the viewpoint located at the optical center of image I_j with viewing direction given by angles θ (pan) and ϕ (tilt). Once each candidate panorama has been scored, I select a set of final panoramas from among this set. To do so, I select the top scoring candidate panorama I^*, remove all images that have a non-zero reprojection score for some view $v(I^*, \theta, \phi)$, then repeat this process until no remaining candidate's score is above a threshold. The Panoramas detected in the Pantheon collection are shown in Figure 4.8, and as images in Figure 4.9.

4.3.2 Navigation

Similar to the orbits, the discovered panoramas are exposed to the user as thumbnails at the bottom of the viewer and can be easily accessed by clicking on them. While these photos are taken quite close to each other, there is significant parallax between them which prevents stitching them into a single seamless mosaic. Hence, I still show a single photo to the user but the user can simply drag left/right and up/down to see more photos associated with the panorama. The arrows on the sides of the screen indicate which direction the user can pan in to see more photos. Again, only 2 degrees

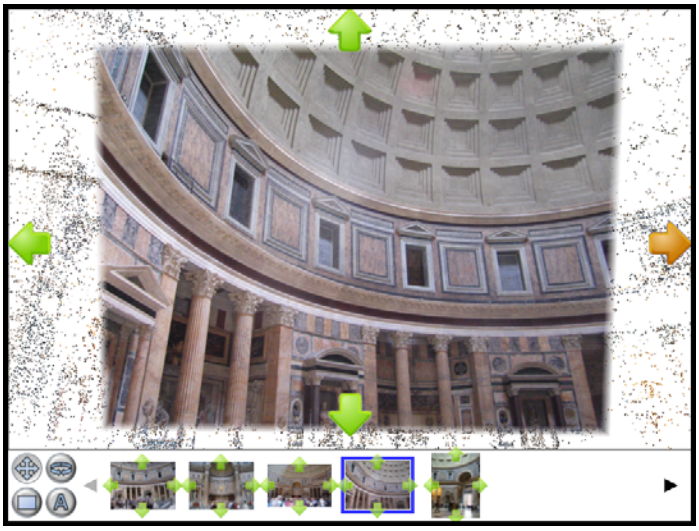


Figure 4.9: Pathfinder user interface with detected panoramas. The scene viewer displays the currently photo in the main view, and shows suggested controls in the thumbnail pane at the bottom of the screen. Arrows on the sides of the screen indicate which directions the user can pan to find more views.

of freedom are exposed to the user making it very easy to navigate the scene. The Pantheon dataset (601 images) was found to be most amenable to panorama detection as lot of people capture photos of the inside looking in different directions.

4.4 Discussion

I detect two kinds of image mosaics in Internet photo collections, rotational panorama and circular orbits, that permit easier navigation controls. These controls, together with other controls introduced in [93] transform the PhotoTourism viewer into a tool for continuously exploring and navigating a scene.

While these controls are intuitive and fit many real world scenes, there are limitations. E.g., one can generalize circular orbits to linear paths, ellipses, polygons or more free-form shapes. One can also detect locally stitchable image sets discussed in Chapter 3 in these large collections. But besides algorithmic difficulty in identifying

more general constructs, detecting a large number of specialized cases would also clutter the user interface. Hence, detecting all simplified controls using a single unified approach and then presenting them in an uncluttered coherent interface to the user is a challenge.

The Pathfinder interface [93] still shows a single photo at a time, crossfading between photos as the user navigates. While Internet photo collections are varied and capture the entire variation in appearance of a scene, transitioning between photos where the appearance is wildly different (day and night photos for instance) or photos with occluders (people posing in front of the scene) is visually jarring and can be disorienting to the user. While Snavely *et al.* [93] implement a simple color compensation method which models an affine transformation per color channel, compensating between two images which are very different, say an image taken in sunny weather and an image taken in cloudy weather is very difficult. Hence, compensating for appearance changes and handling occluders is important for visually pleasing transitions – a problem I consider in the following chapter.

Chapter 5

APPEARANCE MODELING FOR INTERNET PHOTO COLLECTIONS

Internet photo collections, e.g., from Flickr [33], exhibit a wide variance in their appearance, having been taken at different times of the day, under very different weather conditions, by different cameras, and from different viewpoints. The PhotoTourism system [94] relies on 3D transitions between photos to convey how the photo are arranged in space and impart a sense of structure of the scene. the panoramic and orbital controls introduced in the previous chapter make the navigation task easier but still rely on good transitions between photos so as to ensure seamless navigation. Hence, understanding and modeling the variation in appearance of these photo collections is an important problem.

How can one model the variability in appearance of these photos? A key property of man-made scenes is that they are not random. In particular, man-made scenes tend to be dominated by a small number of surface orientations. Similarly surface properties like the Bidirectional Reflectance Distribution Function (BRDF), which encodes the reflectance properties, can be complex, but many real BRDFs can be well-approximated by a low-rank linear basis [63]. Similar considerations apply for illumination; for example, studies have shown that the space of daylight spectra is roughly two- or three-dimensional [96]. Based on these observations, I claim that the appearance of such scenes can be approximated by a linear model, i.e., each possible appearance of the scene can be represented by a linear combination of a set of fixed basis appearances. To this end, I first prove a number of theoretical results justifying the claim. Afterwards, I develop a method for modeling the appearance of Internet

photos and show applications of the same.

I seek to find the *dimensionality* of the appearance space of Internet photos. More specifically, suppose you stacked all photos taken of a particular scene as rows in a matrix – what is the rank of that matrix?[1]

It is well known that certain types of image collections tend to be low-rank in practice, and can be spanned using linear combination of a small number of basis views computed using tools like Principle Component Analysis (PCA) or Singular Value Decomposition (SVD). First exploited in early work on eigenfaces [48, 102], these rank-reduction methods have become the basis for a broad range of successful applications in recognition [72, 66], tracking [42], background modeling [68], image-based rendering [104], BRDF modeling [44, 63], compression and other domains.

In spite of the wide-spread use of rank-reduction on images, however, there is little theoretical justification that appearance space should be low-rank in general. An exception is the case of Lambertian scenes, for which a number of elegant results exist. Shashua [89] proved that three images are sufficient to span the full range of images of a Lambertian scene rendered under distant lighting and a fixed viewpoint, neglecting shadows. Belhumeur and Kriegman [8] considered the case of attached shadows, observing that the valid images lie in a restricted range of 3D subspace which they called the *illumination cone*. Basri and Jacobs [7], and Ramamoorthi and Hanrahan [79] independently showed that the illumination cone is well approximated with 9 basis images. Ramamoorthi more recently [78] improved this bound to 5 images, bringing the theory in line with empirical studies on the dimensionality of face images [29].

Very little is known, however, about the dimensionality of images of *real-world scenes*, composed of real shapes, BRDFs, and illumination conditions. I introduce new theoretical upper bounds on the dimensionality of scene appearance (improving

[1]By dimensionality, I refer to linear dimensionality.

on previous results by Belhumeur and Kriegman [8]). While I make a few limiting assumptions (distant lighting, distant viewer, no cast shadows, interreflections or subsurface scattering), these results bring the theory to the point where it can capture much of the extreme variability in these Internet photo collections. Further, many of the results are still seen to hold empirically even when these assumptions are violated.

The highlights include a factorization framework for analyzing dimensionality questions, introduced in section 5.1. Using this framework, I prove new upper bounds on the number of basis images, allowing for variable illumination direction and spectra, viewpoint, BRDFs, and convolution effects (e.g., blur). Importantly, all prior low-rank results for Lambertian scenes [89, 8, 7, 79, 78] do not apply under variations in light spectrum (even if the images are grayscale). I introduce new results that allow the light spectrum to vary in certain ways (Section 5.1.3), greatly broadening the scope of application (e.g., to outdoors). In Section 5.2, I perform experiments on BRDF databases to empirically verify some of the assumptions made[2]. Finally in Section 5.3, I demonstrate that low rank linear models can be used to model the appearance of outdoor scenes in Internet photo collections and conclude by showing how low-rank linear models can be used for a number of applications like reconstructing images, removing occluders, and expanding the field of view (Section 5.4).[3]

5.1 Theory

In this section, I present theoretical results. I first introduce a new framework to analyze the factorization of images (Section 5.1.1) which yields new insights and results in Section 5.1.2. Finally, I introduce wavelength (Section 5.1.3) bringing the theory closer to the real world images captured by cameras.

Throughout the discussion, I assume that images are lit by distant light sources and observed from distant viewpoints. I ignore indirect illumination effects like transparent

[2]These experiments were done in collaboration with Hao Du.

[3]A concise version of results in this chapter was published in [35].

and translucent materials, interreflections, cast shadows and subsurface scattering. The theory does account for attached shadows, however. Initially, I also make the assumption that images are taken from a fixed viewpoint, which I relax later on. Similarly, I begin by considering grayscale images captured at a constant illumination spectrum across images and talk about more complex and practical cases in Section 5.1.3.

5.1.1 Four Factorizations of the Image Matrix

Suppose we are given a set of n-pixel images of a scene, $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_m$ taken under varying illumination conditions. For now, assume that all images are taken from the same viewpoint, i.e., the corresponding pixels across images correspond to the same 3D scene point. Consider the $m \times n$ matrix $\mathbf{M}$ obtained by stacking $\mathbf{I}_1, \mathbf{I}_2, .., \mathbf{I}_m$ as rows of the matrix. Each row of $\mathbf{M}$ is an image, and each column describes the appearance of a single pixel, say x, under different illumination conditions, referred to as the *profile* of the pixel and denoted by $\mathbf{P}_x$, where $\mathbf{P}_x(i) = \mathbf{I}_i(x)$.

Consider a factorization of $\mathbf{M}$ into the product of two rank-k matrices:

$$\mathbf{M}_{m\times n} = \mathbf{C}_{m\times k}\mathbf{D}_{k\times n}. \tag{5.1}$$

Such a factorization may be obtained by PCA or SVD, for instance. We present four different interpretations of such a factorization, shown in Figure 5.1.

5.1.1.1 Basis Images

First, the rows of $\mathbf{D}$ can be interpreted as basis images, denoted by $\mathbf{B}^I$, and the rows of $\mathbf{C}$ can be interpreted as coefficients. This interpretation, shown in Figure 5.1(a), is commonly used. For instance, in work on eigenfaces [48], the eigenvectors obtained from PCA comprise the basis images (assuming mean subtracted data). Here each

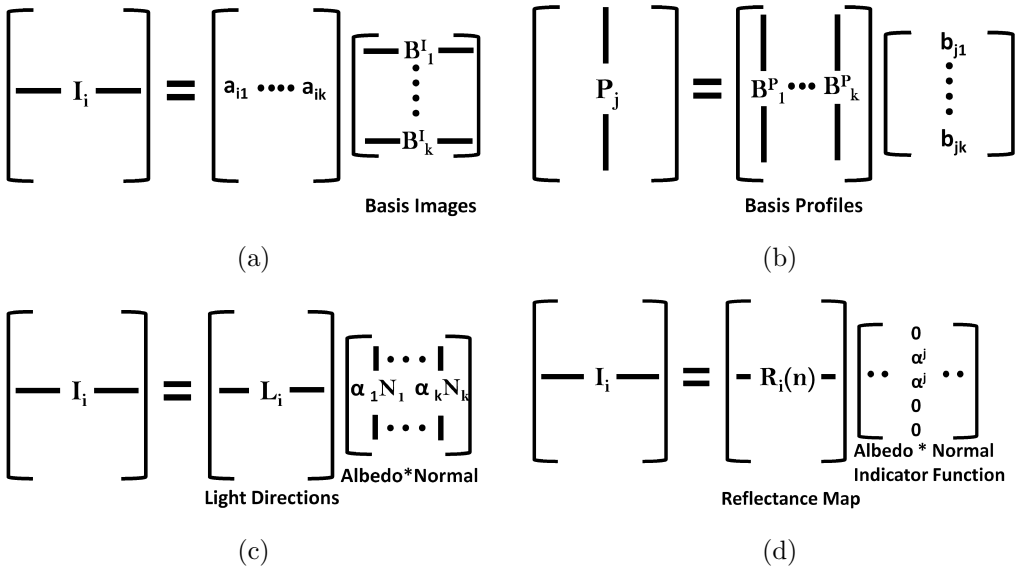


Figure 5.1: Four interpretations of factorization of image matrices. First, each image can be expressed as a linear combination of a set of basis images (a). Alternatively, the profile of each pixel can be expressed as a linear combination of a set of basis profiles (b). In the case of a Lambertian scene, the basis profiles and basis images assume special meaning (c). Finally, (d) shows the reflectance map interpretation.

image $\mathbf{I}_i$ is a linear combination of basis images:

$$\mathbf{I}_i = \sum_{j=1}^{k} a_{ij}\mathbf{B}_j^I. \tag{5.2}$$

5.1.1.2 Basis Profiles

Another way to interpret this factorization is that each column (profile) $\mathbf{P}_j$ of $\mathbf{M}$ can be interpreted as a linear combination of columns of $\mathbf{C}$, with coefficients determined by columns of $\mathbf{D}$, as shown in Figure 5.1(b). In this interpretation, the columns of $\mathbf{C}$ form basis profiles, denoted by $\mathbf{B}^P$:

$$\mathbf{P}_j = \sum_{i=1}^{k} b_{ji}\mathbf{B}_i^P. \tag{5.3}$$

5.1.1.3 The Lambertian Case

For Lambertian scenes, neglecting *any* shadows, the rank of $\mathbf{M}$ is 3 [89] , and the basis profiles and the basis images assume a special meaning, shown in Figure 5.1(c). $\mathbf{D}$ is a $3 \times n$ matrix, where the j^{th} column of $\mathbf{D}$ encodes the normal times the albedo at the j^{th} pixel in the scene. $\mathbf{C}$ is a $m \times 3$ matrix where the i^{th} row encodes the lighting direction times the light intensity for the i^{th} image. Hence, the basis images represent scene properties (normals and albedos) and the basis profiles encode illumination properties. In particular, each basis profile contains the light intensity along a coordinate axis for each image.

5.1.1.4 The Reflectance Map Interpretation

The reflectance map [46], is defined for an image of a scene with a single BRDF as a function $R(\hat{\mathbf{n}})$ that maps scene normals to image intensity. $R(\hat{\mathbf{n}})$ can be encoded as an image of a sphere with the same BRDF as the scene and taken from the same viewpoint under identical illumination conditions.

We can alternately define $R_i(\hat{\mathbf{n}})$ using the rendering equation which under our assumptions, can be written as

$$\mathbf{I}_i(x) = \int_\Omega \alpha^x \rho^x(\omega', \omega) L_i(\hat{\mathbf{n}}^x, \omega')(-\hat{\omega'} \cdot \hat{\mathbf{n}}^x)_+ \mathbf{d}\omega' \tag{5.4}$$

where the integral is over a hemisphere of inward directions ω', ω is the viewing direction for point x, ρ^x is the reflectance function at point x (evaluated at ω', ω), $L_i(\hat{\mathbf{n}}^x, \omega')$ is the light arriving from direction ω' for image $\mathbf{I}_i$, and $\hat{\mathbf{n}}^x$ is the normal at x. The + subscript on the dot product indicates that it is clamped below to 0 to account for attached shadows.

Given this, we can define $R_i(\hat{\mathbf{n}})$ as

$$R_i(\hat{\mathbf{n}}) = \int_\Omega \rho(\omega', \omega) L_i(\hat{\mathbf{n}}, \omega')(-\hat{\omega'}.\hat{\mathbf{n}})_+ \mathbf{d}\omega' \tag{5.5}$$

where ρ^x has been replaced by ρ, as R_i represents a scene with a single BRDF.

Let us denote by $\mathbf{R}_i$ the image of the sphere when taken under identical illumination conditions as in image $\mathbf{I}_i$. Then we can write

$$\mathbf{I}_i{}^T = \mathbf{R}_i{}^T\mathbf{D} \tag{5.6}$$

where $\mathbf{D}$ is defined as:

$$\mathbf{D}(j,k) = \begin{cases} 1 & \text{if } \hat{\mathbf{n}}^k = \hat{\mathbf{m}}_j \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

where $\mathbf{D}(j,k)$ represent the value in the j^{th} row and k^{th} column of $\mathbf{D}$, $\hat{\mathbf{m}}_j$ is the normal at the j^{th} pixel of $\mathbf{R}_i$ and $\hat{\mathbf{n}}^k$ is the normal at the k^{th} pixel in the scene. The k^{th} column of $\mathbf{D}$ can be thought of as a *normal indicator function* $\mathbf{v}_k$.

It often happens that the BRDF is same across the scene save for a scaling factor (the albedo). The reflectance map factorization can also incorporate per pixel albedos if we define the k^{th} column of $\mathbf{D}$ as $\alpha^k\mathbf{v}_k$ where α^k is the albedo of the k^{th} pixel.

Now, observing that $\mathbf{D}$ does not depend on i, one can write $\mathbf{M} = \mathbf{CD}$ where $\mathbf{C}$ contains the reflectance maps $\mathbf{R}_i$'s stacked as rows.

5.1.2 Upper Bound on Rank of M

Belhumeur and Kriegman [8] proved that, given an arbitrary scene with a single material and k_n distinct normals, the space of images of the scene taken from a fixed, distant viewpoint with distant lighting and no cast shadows is *exactly* k_n-dimensional. This result justifies the use of linear models for real-world scenes. For instance, many man-made scenes consist of large planar regions (such as walls and ground), and therefore contain only a small number of distinct normals. Curved surfaces may also be approximated by piecewise planar surfaces.

I first show how an upper bound of k_n can be seen to hold true for a scene with a single BRDF using the reflectance map interpretation of the factorization. Note that this upper bound is the same as that derived by Belhumeur and Kriegman [8].

However, I later extend our result under a number of different and more general settings.

From the reflectance map interpretation $\mathbf{M} = \mathbf{CD}$, it is easy to see that only k_n rows of $\mathbf{D}$ will be non-zero when the number of distinct normals in the scene is k_n. Hence, $rank(\mathbf{D}) \leq k_n$, which gives us an upper bound of k_n on $rank(\mathbf{M})$ as well. Belhumeur and Kriegman [8] derive the same result but via a different route.

Now consider the more general case where there are k_ρ materials and k_n normals in the scene. In this case, I first define reflectance maps corresponding to every BRDF for each image, i.e.,

$$\mathbf{I}_i{}^T = \sum_{l=1}^{k_\rho} \mathbf{R}_{il}{}^T \mathbf{D}_l \tag{5.8}$$

where $\mathbf{D}_l$ now encodes the distribution of normals corresponding to the l^{th} material, i.e.,

$$\mathbf{D}_l(j,k) = \begin{cases} \alpha^k & \text{if } \hat{\mathbf{n}}^k = \hat{\mathbf{m}}_j \text{ and } \rho^k = \rho_l \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

Hence, $\mathbf{M}{=}\sum_{l=1}^{k_\rho} \mathbf{C}_l\mathbf{D}_l$, which implies that $rank(\mathbf{M}){\leq}k_\rho k_n$. More precisely

$$rank(\mathbf{M}) \leq \sum_{l=1}^{k_\rho} N(l) \tag{5.10}$$

where $N(l)$ is the number of orientations corresponding to material l. Hence we have proven the following:

Theorem 1 *Consider a scene with k_ρ different BRDFs and k_n distinct normals. Consider the images $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_m$ of the scene obtained from a fixed distant viewpoint under different distant illuminations $L_{f_1}, L_{f_2}, \ldots, L_{f_m}$. Assuming that there are no cast shadows, the rank of the matrix M obtained by stacking $\mathbf{I}_1, \mathbf{I}_2..\mathbf{I}_m$ as rows is at most $k_\rho k_n$.*

It is also instructive to write $\mathbf{M} = \sum_{l=1}^{k_\rho} \mathbf{C}_l\mathbf{D}_l$ in the form $\mathbf{M} = \mathbf{CD}$ so that basis images and basis profiles can be explicitly defined. This can be done by stacking $\mathbf{C}_l$

side by side, i.e., $\mathbf{C} = [\mathbf{C}_1|\mathbf{C}_2|..|\mathbf{C}_{k_\rho}]$ and stacking $\mathbf{D}_l$ one over another. Finally, we can remove all zero rows from $\mathbf{D}$ and corresponding columns from $\mathbf{C}$ leaving at most $k_\rho k_n$ rows in $\mathbf{D}$, which correspond to basis images, and basis profiles are the remaining columns of $\mathbf{C}$. The columns of $\mathbf{D}$ are of the form $\mathbf{d}_k = \alpha^k \mathbf{v}_k$ where $\mathbf{v}_k$ is a $0, 1$ vector that can be thought of as a *normal-material indicator function*.

The result may be modified to accommodate anisotropic BRDFs as well. For anisotropic materials, one needs to parametrize by both the *orientation* and the normal. Hence, one can derive the same bound where k_n now refers to the number of distinct orientations times normals in the scene.

In the following sections, I extend this result to a number of common scenarios.

5.1.2.1 Linear Families of BRDFs

While the world is composed of diverse materials, it has been argued [80, 63] that the space of BRDFs is low dimensional. I also verify this by conducting experiments on the CUReT [21] database of BRDFs (Section 5.2.2).

Thus, I now generalize to the case when ρ^x is contained in the linear span of $\{\rho_1, \rho_2, \ldots, \rho_{k_\rho}\}$, i.e., $\rho^x = \Sigma_{l=1}^{k_\rho} c_l(x)\rho_l$. In this case, $\mathbf{I}_i$ can be represented as a sum of matrix products, ${\mathbf{I}_i}^T = \sum_{l=1}^{k_\rho} {\mathbf{R}_{il}}^T \mathbf{D}_l$, where

$$\mathbf{D}_l(j,k) = \begin{cases} \alpha^x c_l(x) & \text{if } \hat{\mathbf{n}}^k = \hat{\mathbf{m}}_j \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

Hence the upper bound of $k_\rho k_n$ still holds, i.e., the rank is bounded by the dimensionality of BRDF family times the number of normals.

5.1.2.2 Low-dimensional BRDFs

Certain BRDFs tend to be *low-dimensional*. For example, three basis images suffice to span the images of a Lambertian scene captured under different lighting conditions, in the absence of shadows. Formally, I call a BRDF K-dimensional if the rank of the

matrix $\mathbf{C}$ obtained by stacking reflectance maps obtained under arbitrary sampling of illumination conditions is always at most K. In the presence of such materials, the upper bound on dimensionality may be reduced to $\sum_{i=1}^{k_\rho} K(i)$, where $K(i)$ is the rank of the i^{th} BRDF.

I used the CUReT database for estimating the dimensionality of each material in the database and found that for 49 of the 61 material, the reconstruction error is less than 10% using 9 basis vectors (Section 5.2.1).

5.1.2.3 Varying Viewpoint

Given images taken from different viewpoints, it is trivial to extend the upper bound to $k_v k_\rho k_n$ where k_v is the number of distinct viewpoints. However a much better bound of $k_\rho k_n$ holds true if we know the pixel corresponding to a point x' in the scene in every image. This correspondence can be found, for instance, if the camera parameters of each image and the 3D geometry of the scene are known. Using this correspondence, we can rearrange the pixels in each image so that the x^{th} pixel in every image corresponds to the same scene point. I assume that every scene point is seen by every image (I relax this assumption in Section 5.3.1). I again consider the rank of the matrix $\mathbf{M}$ obtained by stacking these rearranged images. The argument for Theorem 1 still holds, with R_{il} now defined as:

$$R_{il}(\hat{\mathbf{n}}) = \int_\Omega \rho_l(\omega', \omega_\mathbf{i}) L_i(\hat{\mathbf{n}}, \omega')(-\hat{\omega'}.\hat{\mathbf{n}})_+ \mathbf{d}\omega'. \tag{5.12}$$

where $\omega_\mathbf{i}$ is the viewing direction for image i.

5.1.2.4 Filtered Images

Many real-world images are blurry due to camera shake, or have been otherwise filtered (e.g., software sharpening). We extend the above result to filtered images.

Consider the family of images obtained by convolving image $\mathbf{I}(x, y)$ by an arbitrary $K \times K$ kernel $\mathbf{F}$. Each resulting image can be expressed as $\mathbf{I_{conv}}(x, y) =$

$\sum_{i=1}^{K}\sum_{j=1}^{K}\mathbf{F}(i,j)\mathbf{I}(x-i,y-j)$. Since the space of each of the shifted images $\mathbf{I}(x-i,y-j)$ is at most rank $k_\rho k_n$, it follows that the space of all filtered images of the scene is at most rank $K^2 k_\rho k_n$.

An important special case is the family of radially symmetric filters (e.g., blur, sharpen). These filters can be spanned by a few *basis filters* (The basis filters are simply circles of varying radii.)

Suppose that the family of filters we are concerned with can be spanned by k_f basis filters. Consider convolving each of the $k_\rho k_n$ basis images with each of the k_f basis filters to yield $k_f k_\rho k_n$ images. Any filtered image can then be expressed as a linear combination of these filtered basis images. Hence, the bound reduces to $k_f k_\rho k_n$.

Note that it also takes into account images of different sizes. To see this, one can assume that all different sized images of the scene have been obtained by subsampling an appropriately blurred high resolution image of some fixed resolution. Even though we are applying different blur kernels to these images, the blurred images lie on a linear subspace. Subsampling them to some common lower resolution will also result in a linear subspace.

5.1.3 Light Spectra

Up until now, I assumed that all measurements are done at a particular wavelength of light, and that the spectrum of light is constant over all images. I now consider the case when the camera sensors and light spectra vary between images. Surprisingly, in general, the appearance space of a simple Lambertian scene with a single infinite plane can have unbounded dimension, even for grayscale images. That is because albedos, which were before treated as fixed scalars for every pixel, are now functions of wavelength, allowing the scene to have arbitrary appearance for different wavelengths. In the general case, using a linear response model, we have that

$$\mathbf{I}_i(x) = \int s_i(\lambda)\mathbf{I}_i(x,\lambda)d\lambda \tag{5.13}$$

where $s_i(\lambda)$ is the spectral response of the sensor i and $\mathbf{I}_i(x, \lambda)$ is the intensity of light of wavelength λ arriving at the sensor. I begin by analyzing the general case, then discuss results for some common special cases.

5.1.3.1 The General Case

Consider the matrix $\mathbf{M}$ obtained by stacking images $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_m$ captured by arbitrary sensors. I claim that the rank of $\mathbf{M}$ is bounded by $k_\rho k_n k_\alpha$, where k_α is the number of distinct albedos in the scene.

This result can again be derived from the reflectance map interpretation. I define a reflectance map corresponding to every pair of albedo and BRDF in the scene, with the incidence of normals encoded in the $\mathbf{D}$ matrix. More precisely, ${\mathbf{I}_i}^T = \sum_{h=1}^{k_\alpha} \sum_{l=1}^{k_\rho} \mathbf{R}_{ihl}^T \mathbf{D}_{hl}$ where $\mathbf{R}_{ihl}$ is the image of a sphere with BRDF ρ_l and albedo α_h captured under identical illumination conditions and by the same sensor as $\mathbf{I}_i$, and

$$\mathbf{D}_{hl}(j,k) = \begin{cases} 1 & \text{if } \hat{\mathbf{n}}^k = \hat{\mathbf{m}}_j \text{ and } \alpha^x = \alpha_h \text{ and } \rho^x = \rho_l \\ 0 & \text{otherwise} \end{cases} \tag{5.14}$$

Again, we can write $\mathbf{M} = \sum_{h=1}^{k_\alpha} \sum_{l=1}^{k_\rho} \mathbf{C}_{hl} \mathbf{D}_{hl}$ by stacking up the $\mathbf{R}_{ihl}$'s,. It follows that $rank(\mathbf{M}) \leq k_\rho k_n k_\alpha$.

More generally, the albedos in a scene (as a function of wavelength) may be spanned by k_α basis albedos. It can be shown in a fashion similar to Section 5.1.2.1 that the bound of $k_\rho k_n k_\alpha$ extends to this case as well.

5.1.3.2 Light Sources with Constant Spectra

Belhumeur and Kriegman [8] showed that images of a Lambertian scene lit by light sources of identical spectra can be spanned by three basis images in the absence of shadows. I do a similar analysis in a more general setting.

Assume that (1) BRDFs do not depend on λ (save for a scale factor, the albedo), (2) all light sources within and across images have the same spectra (but with varying

intensity and direction), and (3) all images are captured by identical sensors with spectral response $s(\lambda)$. Under these assumptions, the bound of $k_\rho k_n$ can be seen to hold true.

Under assumption (2), we can write $L_i(\hat{\mathbf{n}}, \omega', \lambda)$ as $K(\lambda)L'_i(\hat{\mathbf{n}}, \omega')$ and hence,

$$\mathbf{I}_i(x, \lambda) = K(\lambda) \int_\Omega \alpha^x(\lambda)\rho^x(\omega', \omega)L'_i(\hat{\mathbf{n}}, \omega')(-\hat{\omega'}.\hat{\mathbf{n}}^x)_+ \mathbf{d}\omega' \tag{5.15}$$

We can write $\mathbf{I}_i(x, \lambda) = K(\lambda) \sum_{j,k} a_{jk}(i)\mathbf{B}^I_{jk}(x, \lambda)$ by invoking the basis image representation for the expression in the integral (Theorem 1), where the number of basis images $\mathbf{B}^I_{jk}$'s is at most $k_\rho k_n$. Note that the coefficients do not depend on λ as wavelength dependent albedos are encoded in the basis images. Substituting into Eq. (5.13), we get

$$\mathbf{I}_i(x) = \sum_{j,k} a_{jk}(i) \int s(\lambda)K(\lambda)\mathbf{B}^I_{jk}(x, \lambda)d\lambda \tag{5.16}$$

which implies that $\mathbf{I}_i(x) = \sum_{jk} a_{jk}(i)\mathbf{B'}^I_{jk}(x)$ where the new basis images are obtained by integrating over λ, i.e., $\mathbf{B'}^I_{jk}(x) = \int s(\lambda)K(\lambda)\mathbf{B}^I_{jk}(x, \lambda)d\lambda$. Hence, these images can also be spanned by at most $k_\rho k_n$ basis images.

At first, these assumptions might appear too restrictive. I tested assumption (a) using the CUReT database and found strong support for it (Section 5.2.3). If albedos and camera spectral responses are unconstrained, the scene may have an unbounded rank. However, if the camera responses are similar, assumption (c) is a reasonable approximation. Other assumptions may be relaxed by extending the result. For instance, consider the case where a scene is lit by k_L light sources, each with its own spectrum that stays constant across all images. This can model outdoor illumination, which is often approximated as a combination of sunlight and skylight, each with its own spectrum [96]. Here, the bound can be seen to be $k_\rho k_n k_L$ by writing the illumination in the i^{th} image in the form $\sum_{l=1}^{k_l} K_l(\lambda)L_{li}(\omega', \lambda)$.

Similarly, consider the case when $K(\lambda)$ varies from image to image but lies in a linear subspace of dimension k_s. For illumination in outdoor scenes, the spectra is

well approximated by a two or three-dimensional subspace [96]. The bound can be shown to be $k_\rho k_n k_s$ in this case, by writing $K_i(\lambda) = \sum_{l=1}^{k_s} c_l(i) K_l(\lambda)$.

5.1.3.3 RGB Images

Images captured by conventional cameras contain three color channels. Consider RGB images $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_m$, where I concatenate the channels together. Assume that each channel is captured by a separate sensor that is identical across all images,

$$\mathbf{I}_i^c(x) = \int s_c(\lambda) \mathbf{I}_i(x, \lambda) d\lambda \tag{5.17}$$

Consider the matrix $\mathbf{M}^c$ obtained by stacking channel c of all images. Under the assumptions of Section 5.1.3.2, we know that the rank of this matrix is bounded by $k_\rho k_n$ and it can be written as $\mathbf{M}^c = \mathbf{C}\mathbf{D}^c$ (the coefficients are embedded in the matrix $\mathbf{C}$ while the basis images $\mathbf{B'}_{jk}^I$ are stacked up in $\mathbf{D}$). Because $\mathbf{C}$ does not depend on c (From Eq. (5.16), we can see that $s_c(\lambda)$ is encoded in the basis images, i.e., $\mathbf{D}^c$), the rank of the matrix $\mathbf{M}$ obtained by concatenating the channels and stacking them is also bounded by $k_\rho k_n$ (we can write $[\mathbf{M}^1|\mathbf{M}^2|\mathbf{M}^3] = \mathbf{C}[\mathbf{D}^1|\mathbf{D}^2|\mathbf{D}^3]$).

In fact, we can go further and show that profiles corresponding to a particular pixel are identical across channels save for a scaling factor, i.e., there exists $k_c(x)$ for each channel such that $\mathbf{P}_x^c / k_c(x)$ is same for all c. This can be seen by substituting for $\mathbf{I}_i(x, \lambda)$ from Eq. (5.15) in Eq. (5.17) and writing:

$$\mathbf{P}_x^c(i) = k_c(x) \int_\Omega \rho^x(\omega', \omega) L_i'(\hat{\mathbf{n}}, \omega') (-\hat{\omega'}.\hat{\mathbf{n}}^x)_+ \mathbf{d}\omega' \tag{5.18}$$

where

$$k_c(x) = \int s_c(\lambda) K(\lambda) \alpha^x(\lambda) d\lambda \tag{5.19}$$

5.1.4 Summary

I started by proving an upper bound of $k_\rho k_n$ in Theorem 1 and then showed that the same bound holds for images taken from different viewpoints and for linear families of

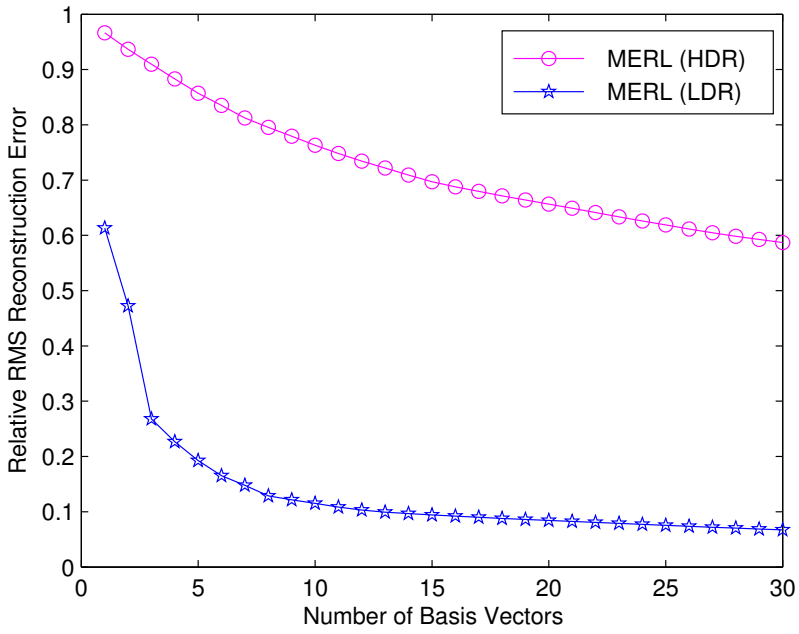


Figure 5.2: Reconstruction Error vs Number of basis images for HDR and LDR images rendered using materials in the MERL database. A hundred 50×50 images of a sphere of each material were rendered under different illumination conditions. Basis images were computed using this collection of all $10,000$ images (100 materials, 100 images per material).

BRDFs. Then I showed that certain BRDFs allow the bound to be lowered. Afterwards, it was shown how filtered images can be handled in my theory. Finally, I introduced wavelength. While in the most general case, the theoretical bound can shown to be $k_\rho k_n k_\alpha$, the bound of $k_\rho k_n$ holds under certain assumptions.

5.2 Experiments on BRDF Databases

To empirically validate the assumptions introduced in Section 5.1, I performed experiments on BRDF databases to ascertain the range of materials present in real world images. I looked at two different databases of BRDFs – the MERL BRDF database [63] which has 100 different materials, and the CUReT database [21], which has 61 different materials. CUReT is more representative of real world materials (e.g., paper,

grass, cloth, etc.) whereas MERL is restricted to machined and painted spheres. Since my goal is to understand the dimensionality of real-world scenes, I chose to focus on CUReT. Also, the prevalence of specularities in MERL in combination with high dynamic range (HDR) capture, makes approximating through linear models much more difficult, as the highlights alone capture the vast majority of image energy. I found, however, that converting the images to a standard 8-bit-per-channel dynamic range (LDR) (and clamping highlights to 255) yields a reasonable fit for MERL database (See Figure 5.2). I use the relative RMS error (here, and in all the subsequent results) to gauge the accuracy of the fit, and is measured as RMS value of the error divided by the RMS value of the original data. Here, to convert HDR images to LDR images, I quantized the range of each image so that 90% of the pixels fall in the range $[0, 255]$ and clamped the pixels outside this range to 255. These results imply that even for this dataset, linear models work reasonably well for real world LDR images. CUReT database is used for the results in the following sections.

5.2.1 Appearance Space of each Material

I first analyze how many basis images are required to span the appearance space of each material. For every material, a hundred 50×50 images of a sphere of that material were rendered. Each image was lit by a distant directional white light whose direction was randomly chosen (uniformly distributed over the front facing hemisphere). The rendered images were reduced to grayscale and SVD was used to compute the basis images corresponding to each material. The green (middle) curve in Figure 5.3 shows the fall of *relative RMS error* as the number of basis images used to model the appearance are increased. The curve is averaged over all 61 materials for CUReT. From the graph, we can see that the error falls to less than 10% after only 6 basis images.

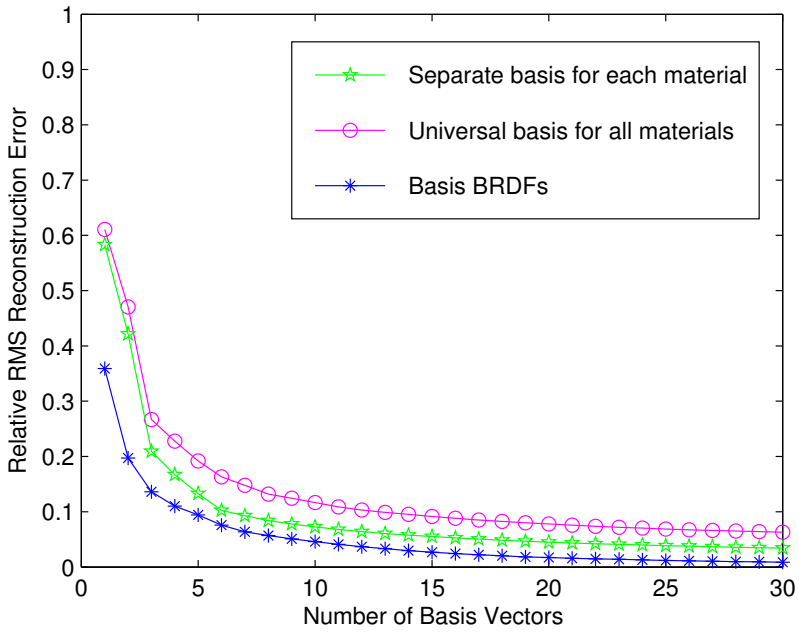


Figure 5.3: The fall in relative RMS error vs number of basis vectors for CUReT database.

5.2.2 Appearance Space of all Materials

In Section 5.1.2.1, I showed that the upper bound on the dimensionality is reduced in case the BRDFs in a scene are contained in a low dimensional linear subspace.

To gauge the range of materials present in the CUReT database, I computed *basis BRDFs* (treating each BRDF as a vector). The BRDFs corresponding to the three color channels were concatenated to form a single large vector. The blue (bottom) curve in Figure 5.3 shows the reconstruction error vs number of basis BRDFs. Note that here the reconstruction error refers to the error in reconstructing the original BRDFs, and not the rendered images.

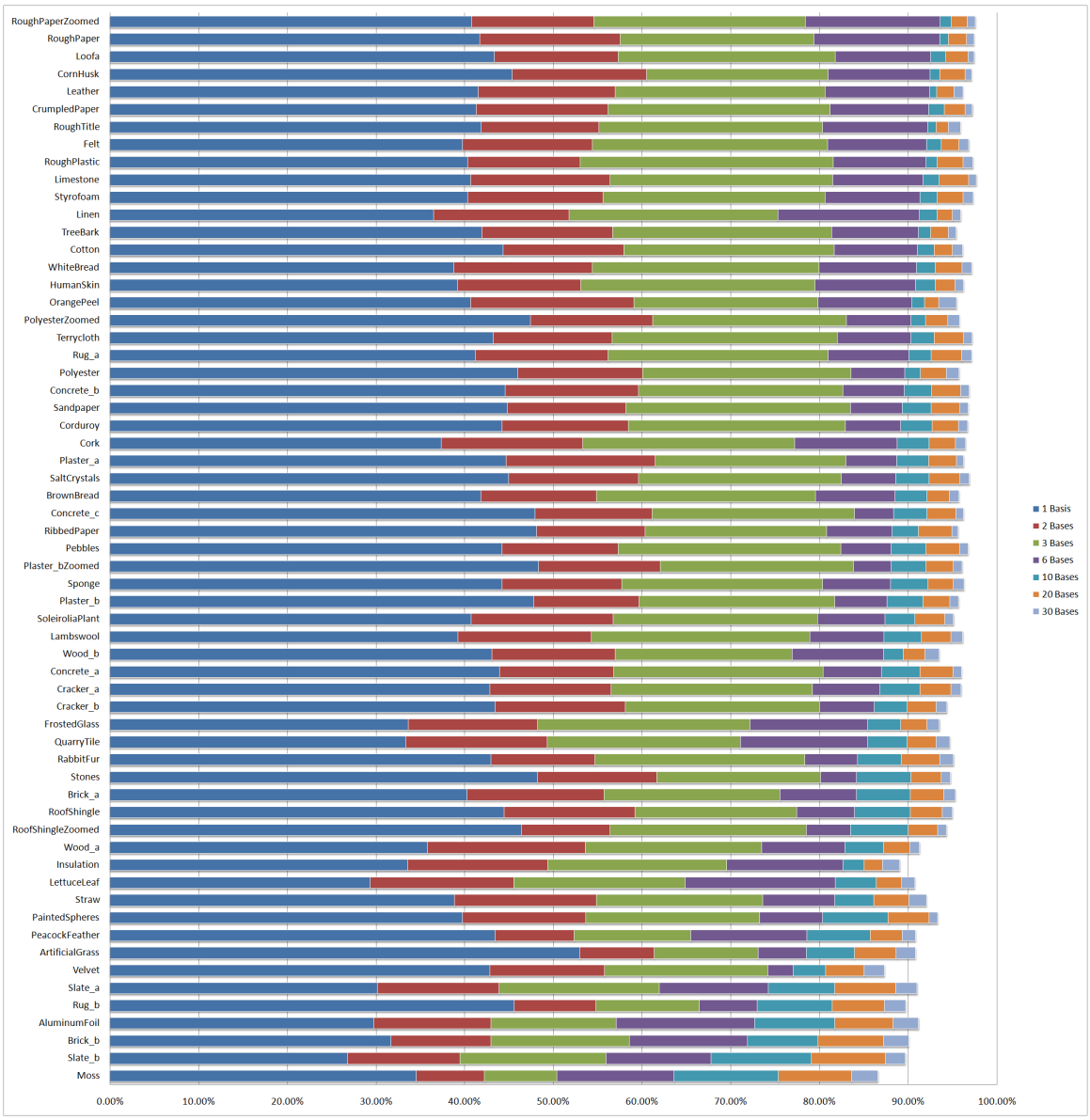


Figure 5.4: Reconstruction accuracy achieved for each material in the CUReT database using $1, 2, 3, 6, 10, 20$ and 30 basis images respectively. The basis was computed using the set of all images of all materials (universal basis). The first six basis images can be seen in Figure 5.5. The materials have been sorted according to the reconstruction accuracy using six basis images. Best seen in color.

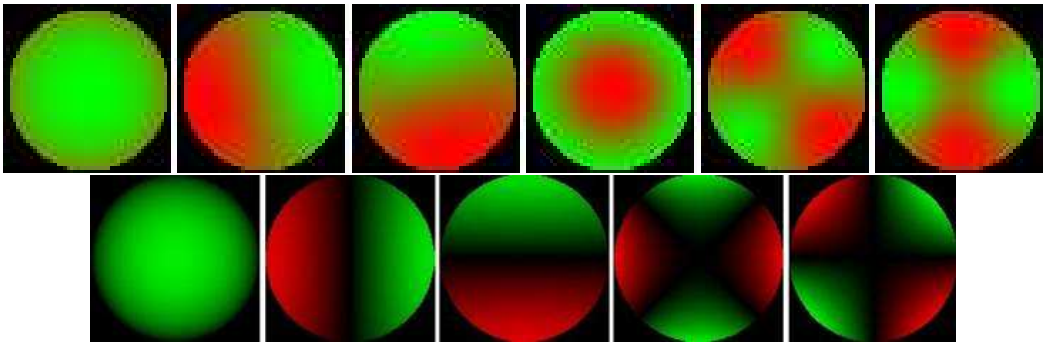


Figure 5.5: The top row shows the first six *universal basis* images computed by performing a SVD over all sphere images rendered using materials in the CUReT database. Five of the six are remarkably similar to the 5 basis images used by Ramamoorthi [78] to span the appearance space of a Lambertian sphere (bottom row).

I also run SVD over *all* 6100 *sphere images* rendered in the previous section (61 materials, 100 images per material) to calculate *universal basis images* and measure the reconstruction error versus the number of basis images (The pink (top) curve in Figure 5.3). The curve is marginally above the curve obtained by calculating a separate basis for each material indicating that the same basis can be shared across a large number of materials.

Reconstruction accuracy of each material is shown in Figure 5.4 (Materials are sorted according to the reconstruction accuracy using six basis images). The first six *universal basis* images are shown in the top row of Figure 5.5, five of which are very similar to the basis images used by Ramamoorthi [78] to span the appearance of a Lambertian sphere. These results show that the Lambertian basis augmented with one additional basis gives an average reconstruction accuracy of 85% for the CUReT database. Some example reconstructions are shown in Figure 5.6. This result is remarkable as it suggests that the non-Lambertian component is relatively insignificant for a wide range of real world materials.

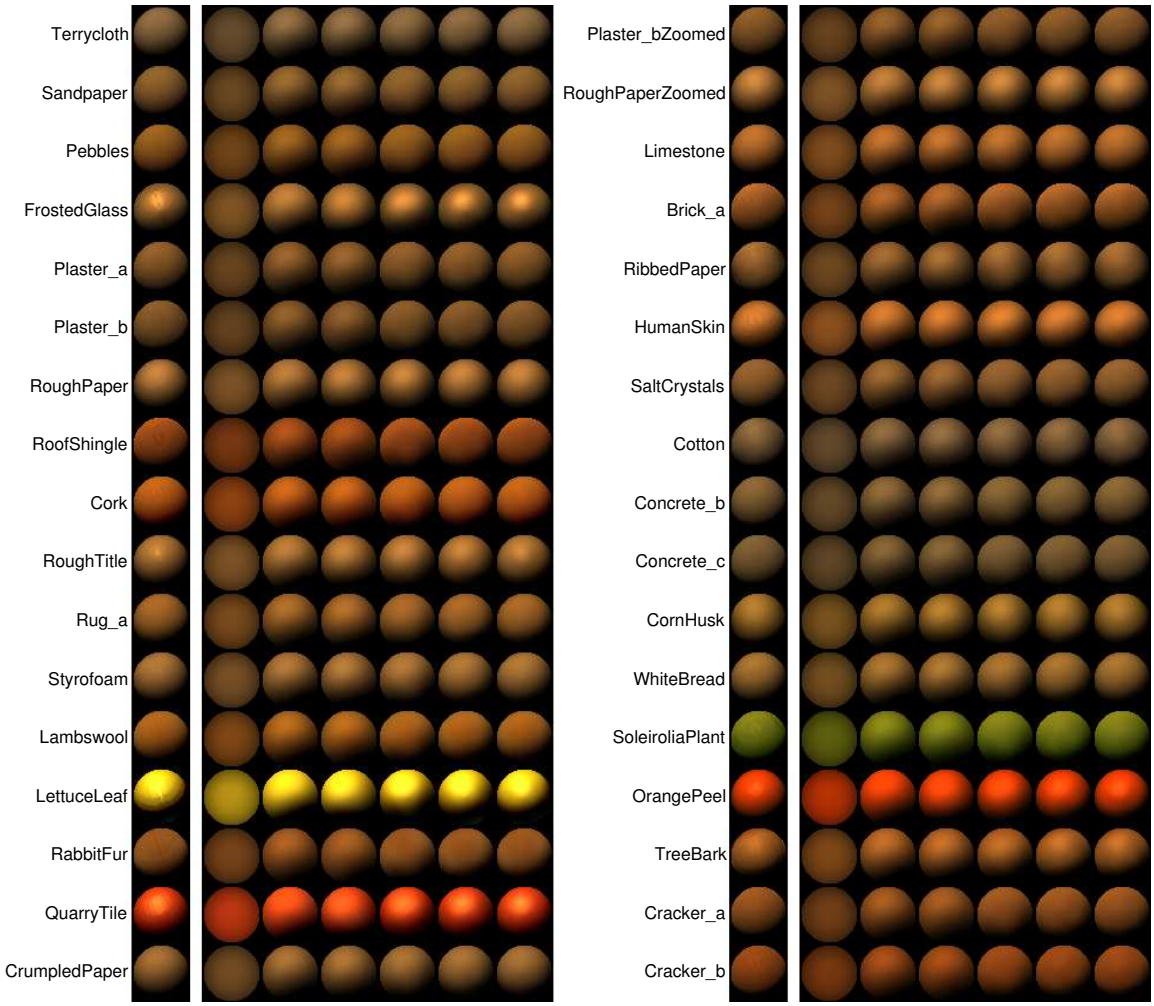


Figure 5.6: Some samples from the CURET BRDF database and corresponding reconstructions using $1, 3, 5, 10, 20$ and 30 basis images. A single basis was learned from images of all the materials. The reconstructions look visually similar (except for smoothing of highlight in some cases) indicating that the same basis may be shared across multiple materials.

5.2.3 BRDF across Color Channels

I also test the assumption made in Section 5.1.3.2, i.e., the BRDF of a material does not depend on wavelength. For each material, I consider the $3 \times N$ matrix obtained by stacking the BRDFs of the 3 channels, where N is the number of samples in each

BRDF for each channel. The mean ratio of first singular value to the second singular value was found to be 49.61 with a minimum of 4.55, indicating that for almost every material, the matrix is close to rank 1 and hence the BRDFs can be approximated as being the same across color channels save for a scaling factor.

It is true that the CUReT database contains samples of real world materials and it does not have many specular materials. With specular BRDFs, it is often the case that the Lambertian part of the BRDF is responsible for the the color of the object while the specular lobe has a much wider spread across different wavelengths. However, the theory can still explain it by treating the BRDF as a linear combination of a specular and a Lambertian BRDF and they are then allowed to scale independently with wavelength.

5.3 Linear Modeling of Internet Photo Collections

The results in Section 5.1 show that linear models well approximate a broad range of real world images. Much of the previous application of linear models has been to images captured in the lab under controlled conditions. Here, I apply linear models to the more challenging case of photos of popular locations downloaded from photo sharing websites. The difficulties here stem from the wide variation in the scene appearance. Moreover, the images are captured using many different cameras and viewpoints.

The organization of this section is as follows. In Section 5.3.1, I give an overview of how I compute the linear model and what representation I use. In Section 5.3.2, I describe in detail how I process the three channels of color images and what assumptions I make. In Section 5.3.3, we give quantitative and qualitative evaluation of the results. Finally in Section 5.4, I demonstrate a couple of novel applications of these linear models.

5.3.1 Basis Computation

Because the input photos are taken from different viewpoints, I first find pixel correspondences. I use the Structure from Motion (SfM) system of Snavely *et al.* [94] to recover the camera parameters. The 3D reconstruction uses the multi view stereo method of Goesele *et al.* [37]. The 3D models are simplified using qslim [36] to a mesh with $\sim 300,000$ faces. I use a simple representation where I associate a color corresponding to each mesh vertex. Images in this representation (which can be thought of as a texture map), can be treated in a fashion similar to images taken from a fixed viewpoint with mesh vertices assuming the role of pixels. However, a single image covers only a part of the scene, i.e., there is missing data in each texture map. To compute basis vectors with missing data, we use the EM based method of Srebro and Jaakkola [95] to compute SVD. However, the algorithm was found to be sensitive to initialization when the amount of missing data is large. I use the method of Roweis [85] which fills the missing data using EM based sensible PCA, to initialize.

Internet photo collections are often dominated by people and other occluders who block the background scene. As my focus is modeling the scene and not the people, I start by manually identifying clean occluder-free images from which I compute a clean basis. I will show later how to handle occlusions in other images using this basis in Section 5.4.2.

5.3.2 Processing Color Images

I cannot directly apply the ideas in Section 5.1.3.3 to these color images as the assumption of identical spectra and identical sensors does not hold for Internet photo collections. One option is to process each color channel independently and compute a separate basis for each channel. The selected clean set still has some outliers (e.g. cast shadows) and processing the three channels independently produces *rainbow* artifacts shown in Figure 5.7 due to inconsistent fits between color channels. Instead, I make

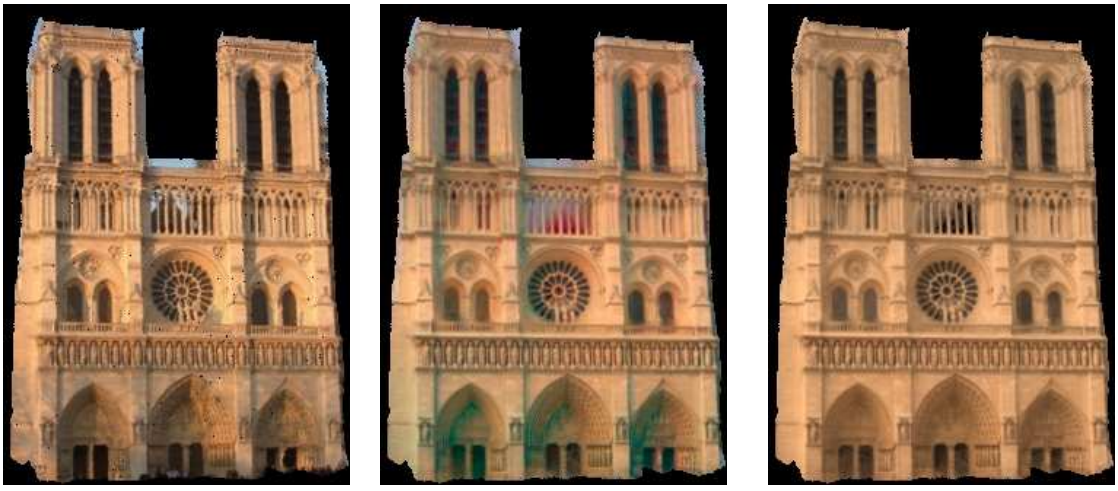


Figure 5.7: The image on the left is the original image. The middle image shows the reconstruction obtained using 5 basis images when processing the color channels independently. Notice the false color in regions of shadow or incorrect geometry (for instance, the doors and the columns above the rose window). The figure on the right is the reconstruction obtained by the described approach.

some simplifying assumptions that allow us to reconstruct the other channels given the reconstruction of one. Hence, I choose to process only the green channel of these images.

First, under the assumptions made in Section 5.1.3.2 and 5.1.3.3, the profile of a pixel x corresponding to, say the red channel, $\mathbf{P_x^{red}}$ can be written as $\mathbf{P_x^{red}} = f^{red}(x)\mathbf{P_x^{green}}$ where

$$f^{red}(x) = \frac{k_{green}(x)}{k_{red}(x)} \tag{5.20}$$

with $k_{red}(x)$ and $k_{green}(x)$ defined by Eq. 5.19. However, Internet photos are not captured by identical cameras and the spectrum of light is also different for different photos (for instance, the spectrum of sunlight in the evening is very different from the spectrum at noon). I give a more rigorous argument later, but intuitively, the combined effect of these two factors (camera and light spectra) can be thought of as individual

scaling applied to the entire channels of an image, i.e., $\mathbf{P_x^{red}}(i) = g^{red}(i) f^{red}(x) \mathbf{P_x^{green}}(i)$ where $g^{red}(i)$ is the scaling applied to the red channel of the i^{th} image and depends upon the camera sensor and the light spectrum of that particular image. $f^{red}(x)$ can be thought of as a measure of the red color of the pixel (relative to the green channel of the image).

Now, if we can recover g^{red} for all images and f^{red} for all pixels, then we can recover the red channel of any image given the reconstruction of the green channel. I found that the following simple method works well in practice for recovering g^{red} and f^{red}. I assume that there exists a dominant value of $g^{red}(i)$ across images, say g^{red}_{dom}. Note that $g^{red}(i)$ depends on the interaction of the camera sensor and the illumination. So, this assumption translates to saying that a large number of photos are taken by similar cameras under lighting conditions with similar spectra. This is plausible as the spectrum of natural illumination for a given scene remains largely constant for a large part of the day. Also, one can safely assume that a large number of cameras will have similar response curve. Hence, we can recover $f^{red}(x)$ by

$$f^{red}(x) g^{red}_{dom} = median_i \left(\frac{\mathbf{P_x^{red}}(i)}{\mathbf{P_x^{green}}(i)} \right) \tag{5.21}$$

where the median is taken over i, i.e, along the profile. Once, we have recovered $f^{red}(x)$ (up to the factor g^{red}_{dom}), one can also recover $g^{red}(i)$ by

$$\frac{g^{red}(i)}{g^{red}_{dom}} = median_x \left(\frac{\mathbf{I_i^{red}}(x)}{f^{red}(x) g^{red}_{dom} \mathbf{I_i^{green}}(x)} \right) \tag{5.22}$$

where now the median is taken over x, i.e., over the pixels of image i. The blue channel can also be processed similarly.

Figure 5.8 shows the relative RMS error in reconstructions of the three color channels for the Notre Dame and Arc De Triomphe datasets where the basis was only computed for the green channel and the other channels were reconstructed using the method outlined above. The curves corresponding to the red and blue channels are

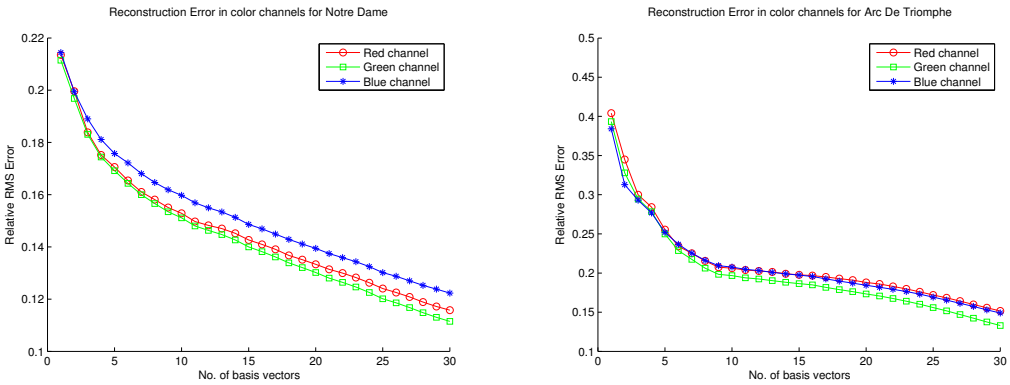


Figure 5.8: Relative RMS error in red, blue and green channels for the Notre Dame and Arc De Triomphe datasets. The blue and red curves are only marginally above the green curve indicating that the accuracy achieved in the reconstruction of the red and blue channel by the described approach is similar to the accuracy in the green channel (via SVD).

seen to be only marginally above the green channel curve, supporting the case of my approximation.

Let us now formally see under what physical conditions the above intuition is correct. I still make the assumption that in a *particular* image, the spectrum of light is the same for all light sources, which allows us to write $L_i(\omega', \lambda)$ as $K^i(\lambda)L'_i(\omega')$ (the illumination can still vary across images). Even though illumination in outdoor scenes is often modeled using two distinct light sources – sunlight and skylight which have different spectra, one can assume that one of them dominates, i.e., the intensity of one is much stronger than the other. E.g., sunlight will dominate on a clear sunny day while skylight will dominate on an overcast day. Repeating the analysis of Section 5.1.3.2 (where we now have $K^i(\lambda)$ instead of $K(\lambda)$), it can be seen that $\mathbf{P^{c_1}_x(i)}/k_{c_1}(x,i) = \mathbf{P^{c_2}_x(i)}/k_{c_2}(x,i)$, where

$$k_{c_l}(x,i) = \int s_{c_l}(\lambda)K^i(\lambda)\alpha^x(\lambda)d\lambda \tag{5.23}$$

i.e., k_{c_l} depends on i as well, unlike in Equation (5.19). We want to be able to write

the above integral in the form $f^{c_l}(x)g^{c_l}(i)$. An assumption under which this holds true is when the albedos remain constant over the range of λ over which the support of spectral responses varies and hence can be taken out of the integral. This essentially translates to saying that the support of the spectral response of, say the red sensor, stays in a small neighborhood of a particular wavelength across different cameras.

5.3.3 Evaluation

I present results on 6 datasets: Notre Dame Cathedral (212 images), Statue of Liberty (318 images), Arc De Triomphe (268 images), Half Dome, Yosemite (95 images), Orvieto Cathedral (228 images), and the Moon (259 images). An image from each of these datasets is shown in the first column of Figure 5.9. Note that the sky and background has been segmented out as we only use a 3D model corresponding to the scene. The Moon presents an interesting case due to its retro-reflective BRDF. I am able to register the Moon images using SfM (There exists sufficient parallax for SfM to work [47]) and then fit a sphere to the 3D points obtained.

All images were gamma corrected assuming $\gamma = 2.2$. As mentioned in Section 5.3.1, I use a manually selected clean set of images for computing the basis. Also, as described in Section 5.3.2, I only need to compute the basis for a single color channel and rest of the color channels can be reconstructed from that. I used the green color channel to compute a basis for each dataset. The reconstructions look reasonably good visually even with three or four basis vectors. With ten basis vectors, some of the finer details such as specularities and self shadowing are also modeled well (I use a basis of size ten to generate results in Section 5.4). There is little visual improvement in the reconstructions after 10 bases though the numerical error continues to fall, but stays at 12% even for 30 basis vectors. Figure 5.11 shows the fall of relative RMS error vs the number of basis images (for the green channel of images). This error can be explained by the fact that even the *clean* set of images have a lot of noise. E.g., Half Dome's view is almost always partially occluded by trees.

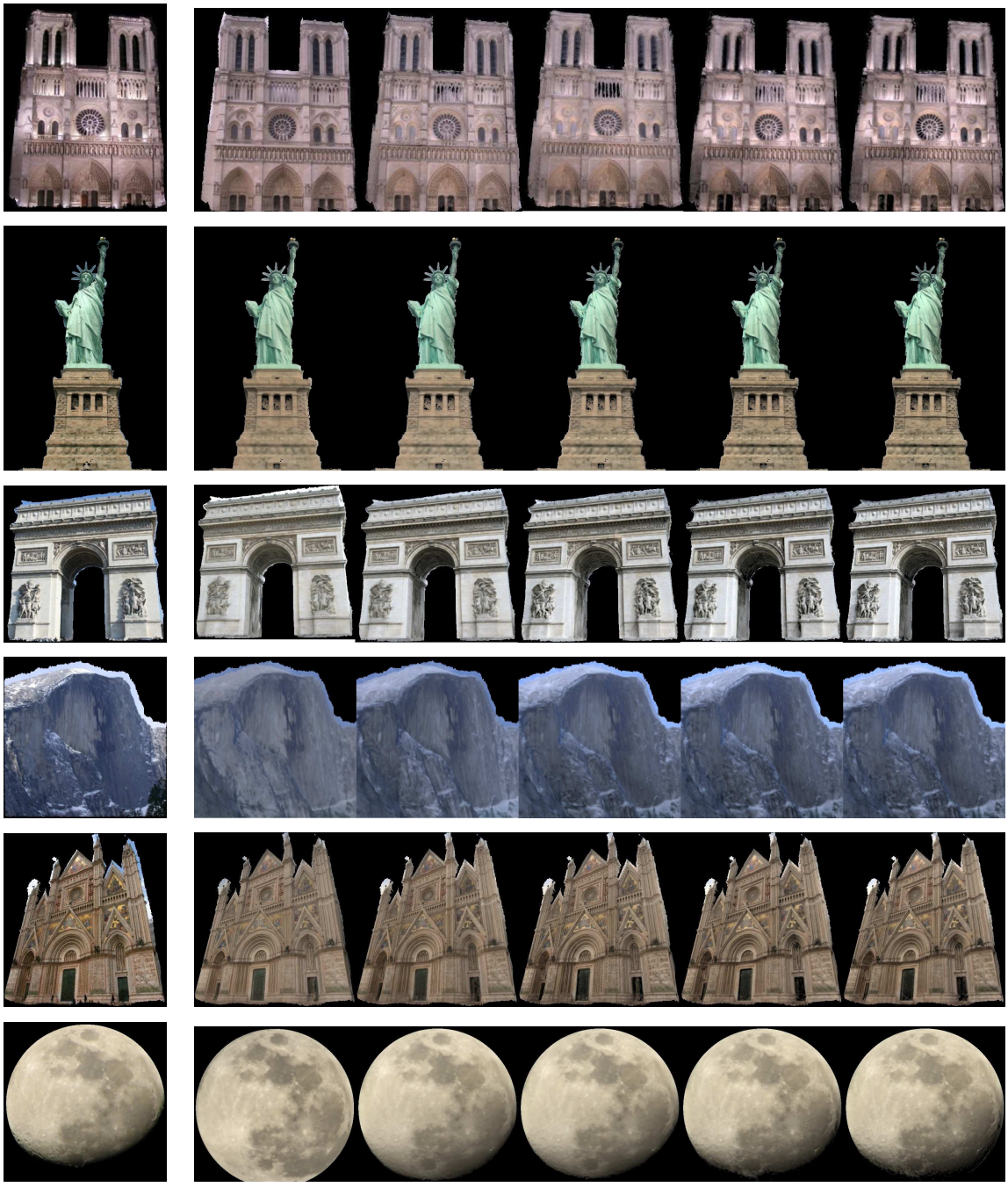
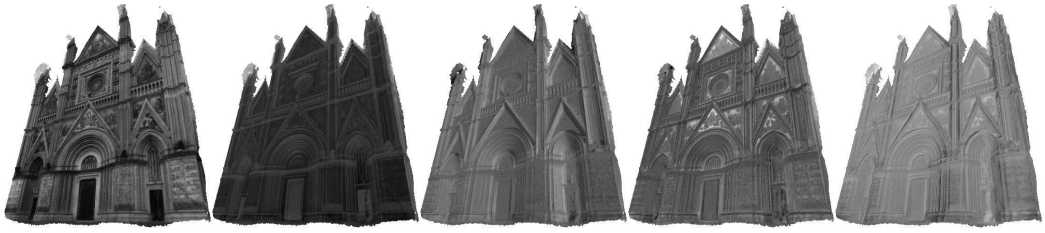


Figure 5.9: Internet datasets and reconstructions. The first columns shows an image from the dataset. The following columns show corresponding reconstructions using $1, 3, 5, 10$ and 20 basis images.



Basis 1 Basis 2 Basis 3 Basis 4 Basis 5

Figure 5.10: First 5 basis images for Orvieto. Basis 1 resembles the mean. Bases 2 and 3 model shading, and Bases 4 and 5 specularities.

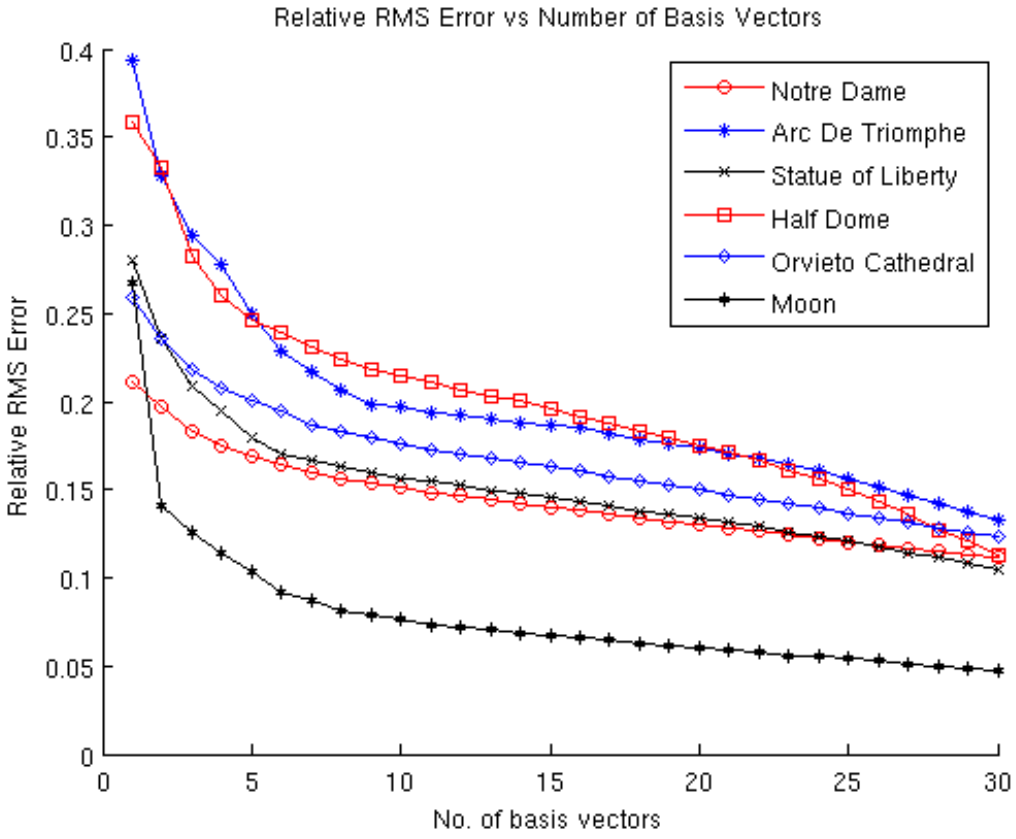


Figure 5.11: Relative RMS Error vs number of basis images for different datasets with a clean set of manually selected images. Even though the reconstruction error is around 12% (due to noise, occluders, etc.) for most datasets even with 30 basis images, reconstructions with only 10 basis images look visually similar.

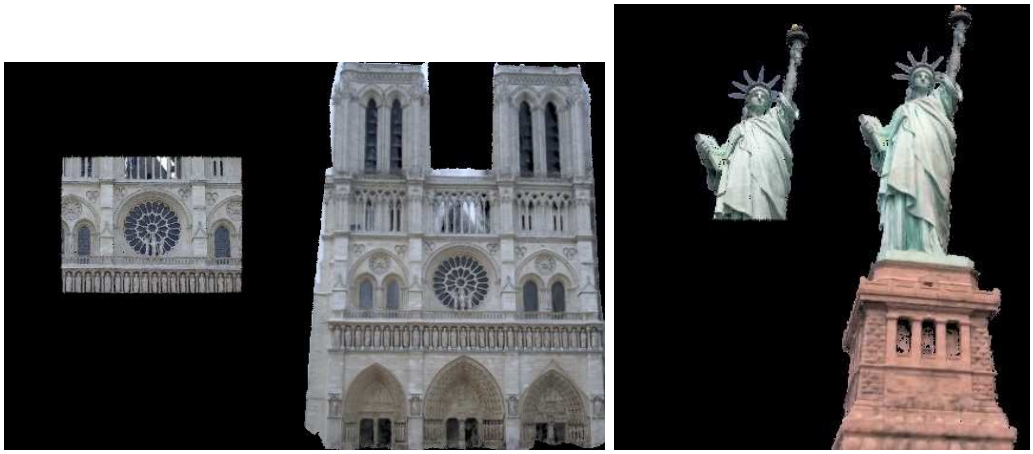
Figure 5.9 shows an example image from these datasets and the corresponding reconstruction for $1, 3, 5, 10$ and 20 basis vectors respectively. The top row (Notre Dame), shows that it becomes possible to model the appearance of night scenes using a larger basis. While such scenes violate our assumptions of distant lighting, as the night-time illumination consists of light sources placed close to the scene, the configuration of light sources is fixed across all night images and hence can be modeled by a single additional basis. The results in the second and third rows (Statue of Liberty and Arc De Triomphe) demonstrate that it is possible to approximate cast shadows using a larger basis even though the shadow boundaries appear blurry in the reconstruction. The linear model does not work well for the Half Dome dataset (the fourth row), as there are drastic appearance changes (such as seasonal snow). An image of Orvieto Cathedral, whose facade is highly specular, is approximated in the fifth row. Figure 5.10 shows the first 5 basis images. While the first basis image simply looks like the mean image, the second and the third model diffuse shading. The fourth and fifth bases seem to model view dependent effects (highlights). Note that the facade of the cathedral is planar and under the assumptions of distant viewer and distant lighting, the specular highlight should ideally cover the whole facade. The presence of specular highlights only on a portion of the facade implies that the viewer is close to the scene which is a violation of our assumption of distant viewer. But as was the case in night scenes, a particular configuration of near-viewpoint and the lighting direction can be modeled by a single additional basis image. For the Moon in the last row, the appearance is modeled well using the first basis, while subsequent bases explain the shadows and the *texture at the terminator* [49].

5.4 Applications

I now show a few novel and interesting applications of linear scene appearance modeling. For all the results shown in the chapter , I empirically chose a basis of size 10.

5.4.1 View Expansion

As was mentioned in section 5.3.1, a single image might cover only a part of the scene. However, since the basis computation method can interpolate missing data, the derived basis images (and hence the reconstructions) cover the entire scene allowing us to hallucinate how the parts of the scene, not visible in the original image, would have appeared under similar illumination conditions. The results of this view expansion approach are shown in Figure 5.12.



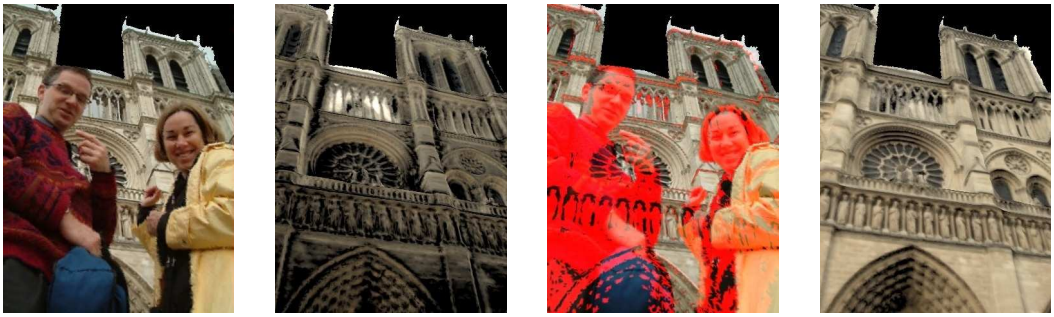
(a) (b)

Figure 5.12: View Expansion: The left image in each pair shows the input image with limited viewing area. The right image shows the reconstructed image with an expanded field of view.

5.4.2 Occluder Removal

Given a basis, we can solve for the coefficients given a new image. I choose a projection approach that is robust to outliers in the image.

Figure 5.13: Occluder removal, where the occluder is removed and the scene behind is rendered under the same illumination conditions by robustly solving for basis image coefficients. In each pair, the left image is the input image while the right image shows the corresponding reconstruction with the occluder removed.



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

Figure 5.14: (a) shows the input image. (b) shows the result of robust projection. It does not work well as the number of outlier pixels is large. (c) shows the precomputed outliers (in red). (d) shows the result of robust projection with precomputed outliers.

This allows us to handle occluders. More precisely, in order to project a new image onto k basis images, I use a RANSAC approach where k pixels are sampled randomly and k coefficients are computed. The number of pixels that lie within a threshold of the original pixel values in the reconstruction obtained using these k coefficients are counted as *inliers*. Finally, the sample with the largest number of inliers is chosen and the estimate of coefficients is refined using all the inliers. Again, I first reconstruct the green channel, and then reconstruct red and blue channels from it (as explained in Section 5.3.2). The reconstruction constructed from the basis using these robustly computed coefficients will be free of occluders. Some results are shown in Figure 5.13.

In presence of very large occluders where the number of outlier pixels is large, the RANSAC based approach may fail. In that case, using the color information, we can also precompute outliers. Assuming we have $f^{red}(x)$ computed for the scene (from the set of images used to compute the basis), given a new image, we can compute $g^{red}(i)$ using Equation 5.22. With this information, one can mark pixels where $|\mathbf{I_i^{red}}(x) - f^{red}(x)g^{red}(i)\mathbf{I_i^{green}}(x)|$ (and a similar expression for the blue channel) is beyond a certain threshold as outliers. Figure 5.14 shows an example.

5.5 Discussion

While I have shown that linear models can be used to model the appearance of scenes for Internet photo collections, this chapter also makes important theoretical contributions which have a much wider scope:

- A simple factorization framework for analyzing dimensionality of image collections.

- New upper bounds on the number of basis images, allowing for variable illumination direction and spectra, viewpoint, BRDFs, and convolution effects (e.g., blur). As in prior work, we assume distant viewers, distant illumination and ignore cast shadows. The results are motivated by models of shape (particularly for man-made scenes), BRDFs, and light spectra that approximate real world scenes.

- Bounds that take into account the illumination spectrum. Prior low-rank results for Lambertian scenes [89, 8, 7, 79, 78] do not apply under variations in light spectrum (even if images are grayscale). Hence prior results are applicable under very controlled conditions.

These results bring the theory much closer to the point where it applies to uncontrolled, real-world scenes.

However, a major drawback of my approach is that it relies on 3D models of the scene. But with more and more 3D data being available through both computer vision methods [34] and range scanning, it should become feasible for more scenes in the future. Once we have accurate 3D models, my approach provides a way to derive consistent looking texture maps for the entire scene. In fact, instead of a single texture map, the approach yields an entire space of texture maps corresponding to different possible appearances of the scene. This appearance space is currently parametrized by

the coefficients of basis appearances. Given photos that are annotated by the angle of the sun, weather conditions, etc., this abstract parametrization could potentially be mapped onto meaningful physical parameters in future providing controls for navigating the appearance space, e.g., one can move a slider to control the direction of the sun and see how the appearance of the scene changes.

Chapter 6

CONCLUSION

The goal of my thesis is to extend and generalize image mosaics to unstructured photo collections and use them to create visualizations of scenes that can be navigated in an intuitive fashion. The specific contributions of my work are:

- In Chapter 3, I develop a mosaicing method that allows for more general camera motion than possible with past approaches. In particular my technique is applicable to a general class of image collections which I call locally stitchable. A key novelty of my approach is that instead of generating a single static stitched image, I implement a dynamic viewer that composites a mosaic on the fly as the user navigates based on the user's viewpoint. Further, the composited mosaic is piecewise-perspective view of the scene that preserves straight lines in the scene, and provides intuitive drag and zoom navigation controls of a traditional mosaic.

- Stitching images seamlessly in the presence of exposure differences, scene motion, parallax and misalignment is a well studied problem in the image mosaicing literature. While past solutions work towards the goal of stitching a single static mosaic, my novel dynamic approach, proposed in Chapter 3, preserves the dynamic aspects of the scene while avoiding stitching artifacts. In particular, the dynamic viewer adjusts the brightness depending on which part of the scene is currently visible to the user and uses dynamic seams to recreate scene motion as the user navigates. Further, my implementation works at frame rate in an interactive viewer.

- In contrast to photos captured by a single user with the intention of creating a

mosaic, Internet photo collections are unstructured. I introduce an algorithm to detect stitchable photo sets in these large collections that allow intuitive panning and orbit controls (Chapter 4). Augmenting the PhotoTourism [94] system with these auto discovered controls makes scene exploration easier and more effective.

- Wide variation in appearance in Internet photos prevents seamless transitions between them in the PhotoTourism interface. In Chapter 5, I prove that photos of most man made scenes can be expressed as a linear combination of a small set of basis photos. While past results hold for simple scenes (Lambertian or single material) and controlled illumination, I develop a theory that proves that linear models are a good approximation in the case of Internet photo collections as well and can be used to model the wide variation in appearance that exists in such collections. The theory introduces a new factorization framework for analyzing linear dimensionality of a set of images that leads to new upper bounds on dimensionality allowing for variable illumination direction and spectra, viewpoint, BRDFs, and convolution effects.

- I introduce an algorithm for calculating basis photos of a scene from Internet photo collections. There are three major challenges that I address here. First, these collections are large and require scalable algorithms. Second, an image only captures a part of the scene, i.e., each image provides an incomplete sample of a specific appearance – the basis photos or appearances for the entire scene are learned from these incomplete samples. Lastly, Internet images have a lot of outliers, e.g., occluders in a scene, and the approach needs to be robust to such outliers. Given the basis photos of a scene, I demonstrate novel applications like expanding the field of view of a photo and removing occluders, e.g., people, from a photo.

6.1 Future Work

While I focused on mosaics in my thesis, the larger goal is to develop methods to generate visualizations of scenes that are both easy to create and navigate. I conclude this thesis with a discussion of future research directions that expand upon my work.

6.1.1 Generalized Image Mosaics in Internet Photo Collections

In my work in Chapter 4, I propose an algorithm to discover two kinds of *mosaics* in Internet photo collections – rotational panoramas and circular orbits. However, there can be more general mosaics embedded in Internet Photo Collections. One can generalize circular orbits to ellipses, lines, or more general paths. One could also detect *locally stitchable* image sets which can then be visualized via the dynamic mosaicing technique described in Chapter 3.

However, a significant challenge in the case of Internet photo collections is that they are not taken with the intention of creating mosaics. E.g., we saw in Chapter 4 that photos that are taken from almost the same viewpoint cannot be stitched into a single mosaic as there is still significant parallax between images that prevents seamless stitching. Similarly, while searching for locally stitchable sets in Internet Photo Collections, a homography may not be able to align two images well enough to stitch them seamlessly. Further, extreme changes in appearance make the problem even harder.

Along these lines, Google Photo Tours [39] are *tours* of famous tourist places computed automatically from photos available on the Internet. A *tour* is a sequence of a few (10 to 20) images that covers the most interesting parts of the scenes. However, the tour is linear, i.e., the user can only move to the previous or the next photo from the current photo, severely limiting the freedom in navigating the scene. Further, the user cannot see a panoramic view of the entire scene by zooming out as individual photos are not stitched into a composite.

6.1.2 Dynamic Mosaicing for Entire Scenes

The dynamic mosaicing technique is applicable to locally stitchable image collections. While they cover several common scenarios, a single locally stitchable image set cannot capture a large complex scene such as an entire house. However, individual rooms and perhaps even connections between them can be captured and visualized via dynamic mosaics. What is a good way to browse a collection of dynamic mosaics connecting them together in an intuitive way that conveys how the different mosaics are arranged relative to each other? Such an approach will truly allow capturing arbitrary scenes with a combination of image mosaics. E.g., when browsing an entire house, individual rooms and connections between them can be captured in separate mosaics. The user should be able to zoom out and visualize how these individual mosaics are arranged relative to each other and transition between them seamlessly.

6.1.3 Spatial, Temporal, and Appearance Space Navigation

A lot of work has been done on spatial navigation of scenes, i.e., viewing a scene from different viewpoints. But there are two other orthogonal dimensions along which a scene can be navigated, namely appearance and time.

Navigating the appearance space of a scene implies being able to visualize different possible appearances of the scene. E.g., one should be able to visualize how the scene appears at twilight, move the position of the sun, say using a slider, and see shadows move across the ground. My work makes initial steps towards this ambitious goal. My appearance modeling approach parametrizes the appearance space of a scene – the coefficients of the basis appearances are the parameters of this space. If one could annotate the photos using the position of the sun, weather conditions, etc. one could map this abstract parametrization on to these physically meaningful factors. These physical parameters can then act as controls to allow the user to navigate through the appearance space of a scene.

Temporal navigation involves experiencing the non-static elements of the scene. Imagine a *play* button on each rendering, that can recreate the ambiance (non static elements like motion, audio, etc.) that existed in the scene. In this vein, Microsoft's Digital Heritage project [86] combines visual and audio experience to create a more immersive experience of the scene but requires a lot of manual work in data capture and processing. Similarly, dynamic mosaics seek to recreate some of the dynamic aspects of the scene, e.g., scene motion and varying brightness, moving away from the static scene assumption of many IBR approaches.

Ultimately, I envision a system that allows the user to move about the scene freely using intuitive controls, visualize different possible appearances of the scene and experience dynamic aspects like scene motion. Such a system will be able to create an experience of *being there,* i.e., a virtual experience that is close to being physically present in the scene.

BIBLIOGRAPHY

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. In *Proc. SIGGRAPH*, pages 853–861, 2006.

[2] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *Proc. SIGGRAPH*, pages 294–302, 2004.

[3] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, 2005.

[4] P. Anandan. A computational framework and an algorithm for the measurement of visual. Technical report, Amherst, MA, USA, 1987.

[5] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43:32–38, 2010.

[6] Microsoft Photosynth App. http://photosynth.net/capture.aspx.

[7] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *PAMI*, 25(2):218–233, 2003.

[8] P. N. Belhumeur and D. J. Kriegman. What is the set of images of an object under all possible lighting conditions. *IJCV*, 28:270–277, 1998.

[9] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. ECCV*, pages 237–252, 1992.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222 –1239, nov 2001.

[11] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74:59–73, August 2007.

[12] M. Brown and D.G. Lowe. Recognising panoramas. In *Proc. ICCV*, pages 1218 –1225, 2003.

[13] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. SIGGRAPH*, pages 425–432, 2001.

[14] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, 2:217–236, October 1983.

[15] R. Carroll, M. Agrawal, and A. Agarwala. Optimizing content-preserving projections for wide-angle images. *Proc. SIGGRAPH*, pages 43:1–43:9, 2009.

[16] Y. Caspi, A. Axelrod, Y. Matsushita, and A. Gamliel. Dynamic stills and clip trailers. *Vis. Comput.*, 22(9):642–652, 2006.

[17] C. Y. Chen and R. Klette. Image stitching - comparisons and new techniques. In *Proceedings of the 8th International Conference on Computer Analysis of Images and Patterns*, pages 615–622, 1999.

[18] S. E. Chen. Quicktime VR: an image-based approach to virtual environment navigation. pages 29–38, 1995.

[19] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH*, pages 279–288, 1993.

[20] C. D. Correa and K. L. Ma. Dynamic video narratives. *Proc. SIGGRAPH*, 29(3), 2010.

[21] K.J. Dana, B. Van-Ginneken, S.K. Nayar, and J.J. Koenderink. Reflectance and Texture of Real World Surfaces. *ACM Trans. on Graphics*, 18(1):1–34, 1999.

[22] J. Davis. Mosaics of scenes with moving objects. In *Proc. CVPR*, pages 354–1361, 1998.

[23] P. Debevec, Y. Yu, and G. Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical report, Berkeley, CA, USA, 1998.

[24] P. E. Debevec. *Modeling and Rendering Architecture from Photographs.* PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

[25] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proc. SIGGRAPH*, pages 369–378, 1997.

[26] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. Proc. SIGGRAPH, pages 257–266, 2002.

[27] A. Eden, M. Uyttendaele, and R. Szeliski. Seamless image stitching of scenes with large motions and exposure differences. In *Proc. CVPR*, 2006.

[28] B. Epshtien, E. Ofek, Y. Wexler, and P. Zhang. Hierarchical photo organization using geometric relevance. In *ACM Int. Symp. on Advances in Geographic Information Systems*, 2007.

[29] R. Epstein, P.W. Hallinan, and A.L. Yuille. 5+/-2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. *IEEE Workshop on Physics-Based Modeling in Computer Vision*, 1995.

[30] http://www.facebook.com.

[31] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. In *Proc. SIGGRAPH*, pages 249–256, 2002.

[32] G.D. Finlayson, M.S. Drew, and B.V. Funt. Diagonal transforms suffice for color constancy. In *Proc. ICCV*, pages 164–171, 1993.

[33] http://www.flickr.com.

[34] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proc. CVPR*, pages 1434 –1441, june 2010.

[35] R. Garg, Hao Du, S.M. Seitz, and N. Snavely. The dimensionality of scene appearance. In *Proc. ICCV*, pages 1917 –1924, 2009.

[36] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proc. SIGGRAPH*, pages 209–216, 1997.

[37] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz. Multi-view stereo for community photo collections. *Proc. ICCV*, pages 1–8, 2007.

[38] J. Good. http://1000memories.com/blog/94-number-of-photos-ever-taken-digital-and-analog-in-shoebox.

[39] http://maps.google.com/phototours.

[40] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. SIGGRAPH*, pages 43–54, 1996.

[41] http://www.gtplanet.net.

[42] G.D. Hager and K. Toyama. X vision: Combining image warping and geometric constraints for fast visual tracking. *Proc. ECCV*, pages 507–517, 1996.

[43] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[44] A. Hertzmann and S. M. Seitz. Shape and materials by example: a photometric stereo approach. *Proc. CVPR*, pages 533–540, 2003.

[45] David Hockney. http://hockneypictures.com/.

[46] B. K. Horn. *Robot Vision*. McGraw-Hill Higher Education, 1986.

[47] W. M. Kaula and P. A. Baxa. The physical librations of the moon, including higher harmonic effects. *Earth, Moon and Planets*, 8(3):287–307, 1973.

[48] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *PAMI*, 12(1):103–108, 1990.

[49] J. J. Koenderink and S. C. Pont. Texture at the terminator. *3D Data Processing Visualization and Transmission, Int. Symp. on*, pages 406–415, 2002.

[50] J. Kopf, B. Chen, R. Szeliski, and M. Cohen. Street slide: browsing street level imagery. *ACM Trans. Graph.*, 29:96:1–96:8, July 2010.

[51] J. Kopf, D. Lischinski, O. Deussen, D. Cohen-Or, and M. Cohen. Locally adapted projections to reduce panorama distortions. *Computer Graphics Forum (Proceedings of EGSR 2009)*, 28(4), 2009.

[52] J. Kopf, M. Uyttendaele, O. Deussen, and M. F. Cohen. Capturing and viewing gigapixel images. *ACM Trans. on Graphics*, 26, 2007.

[53] V. Kwatra, A. Schrodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Proc. SIGGRAPH*, 22:277–286, July 2003.

[54] A. Levin, A. Zomet, S. Peleg, and Y. Weiss. Seamless image stitching in the gradient domain. In *Proc. ECCV*, 2006.

[55] M. Levoy and P. Hanrahan. Light field rendering. Proc. SIGGRAPH, pages 31–42, 1996.

[56] W. Y. Lin, S. Liu, Y. Matsushita, T. T. Ng, and L. F. Cheong. Smoothly varying affine stitching. In *Proc. CVPR*, pages 345–352, 2011.

[57] A. Lippman. Movie-maps: An application of the optical videodisc to computer graphics. *Proc. SIGGRAPH*, 14:32–42, July 1980.

[58] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. *ACM Trans. Graph.*, 25(3):646–653, 2006.

[59] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, November 2004.

[60] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, 1981.

[61] S. Mann and R.W. Picard. *On being undigital with digital cameras: extending dynamic range by combining differently exposed pictures.* M.I.T. Media Laboratory Perceptual Computing Section technical report. Perceptual Computing Section, Media Laboratory, Massachusetts Institute of Technology, 1995.

[62] Bing Maps. http://www.bing.com/maps/.

[63] W. Matusik, H. Pfister, M. Br, and L. Mcmillan. A data-driven reflectance model. *ACM Trans. on Graphics*, 22:759–769, 2003.

[64] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *Proc. SIGGRAPH*, pages 39–46, 1995.

[65] T. Mitsunaga and S.K. Nayar. Radiometric self calibration. In *Proc. CVPR*, 1999.

[66] H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *IJCV*, 14(1):5–24, 1995.

[67] Y. Nomura, L. Zhang, and S.K. Nayar. Scene Collages and Flexible Camera Arrays. In *Proc. of Euro. Symp. on Rendering*, Jun 2007.

[68] N.M. Oliver, B. Rosario, and A.P. Pentland. A Bayesian computer vision system for modeling human interactions. *PAMI*, 22(8):831–843, 2000.

[69] P. Otto. Between the virtual and the actual: Robert barker's panorama of london and the multiplication of the real in late eighteenth-century london, 2007.

[70] http://microsites.lomography.com/spinner-360/history.

[71] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet. Mosaicing on adaptive manifolds. *PAMI*, 22:1144–1154, 2000.

[72] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. *Proc. CVPR*, pages 84–91, 1994.

[73] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *Proc. SIGGRAPH*, pages 313–318, New York, NY, USA, 2003. ACM.

[74] http://www.picasaweb.google.com.

[75] Thomas Porter and Tom Duff. Compositing digital images. *Proc. SIGGRAPH*, 18(3):253–259, 1984.

[76] L. H. Quam. Readings in computer vision: issues, problems, principles, and paradigms. chapter Hierarchical warp stereo, pages 80–86. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[77] Paul Rademacher and Gary Bishop. Multiple-center-of-projection images. Proc. SIGGRAPH, pages 199–206, 1998.

[78] R. Ramamoorthi. Analytic PCA construction for theoretical analysis of lighting variability in images of a Lambertian object. *PAMI*, 24(10):1322–1333, 2002.

[79] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. *Proc. SIGGRAPH*, pages 117–128, 2001.

[80] R. Ramamoorthi and P. Hanrahan. Frequency space environment map rendering. *Proc. SIGGRAPH*, pages 517–526, 2002.

[81] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg. Dynamosaics: video mosaics with non-chronological time. In *Proc. CVPR*, 2005.

[82] E. Reinhard. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann series in computer graphics and geometric modeling. Elsevier, 2006.

[83] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, 2002.

[84] Dynamic Mosaicing Results. http://youtu.be/05IJonUTWXU.

[85] S. Roweis. EM algorithms for PCA and SPCA. *Proc. NIPS*, 10:626–632, 1998.

[86] A. Sankar, A. Prasad, J. Joy, N. Datha, and A. Manchepalli. Digital heritage. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 3503–3504, 2009.

[87] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, Proc. SIGGRAPH, pages 489–498, 2000.

[88] S. M. Seitz and C. R. Dyer. View morphing. In *Proc. SIGGRAPH*, pages 21–30, 1996.

[89] A. Shashua. Geometry and photometry in 3D visual recognition. Technical report, MIT AI Lab, 1992.

[90] H. Y. Shum and S. B. Kang. A survey of image-based rendering techniques. In *Videometrics, SPIE*, pages 2–16, 1999.

[91] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *Proc. ICCV*, pages 1–8, 2007.

[92] K. N. Snavely. *Scene reconstruction and visualization from internet photo collections*. PhD thesis, University of Washington, Seattle, WA, USA, 2008.

[93] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *Proc. SIGGRAPH*, pages 15:1–15:11, 2008.

[94] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *Proc. SIGGRAPH*, pages 835–846, 2006.

[95] N. Srebro and T. Jaakkola. Weighted low-rank approximations. *ICML*, pages 720–727, 2003.

[96] K. Sunkavalli, F. Romeiro, W. Matusik, T. Zickler, and H. Pfister. What do color changes reveal about an outdoor scene? *Proc. CVPR*, pages 1–8, 2008.

[97] R. Szeliski. Image mosaicing for tele-reality applications. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 44 –53, dec 1994.

[98] R. Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2:1–104, January 2006.

[99] R. Szeliski and H. Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Proc. SIGGRAPH*, pages 251–258, 1997.

[100] Techcrunch. http://techcrunch.com/2011/12/17/the-top-20-best-iphone-and-ipad-apps-of-2011/.

[101] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.

[102] M. Turk and A.P. Pentland. Face recognition using eigenfaces. *Proc. CVPR*, 591:586–591, 1991.

[103] M. Uyttendaele, A. Eden, and R. Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *Proc. CVPR*, pages 509–516, 2001.

[104] L.F. Wang, S.B. Kang, R. Szeliski, and H.Y. Shum. Optimal texture map reconstruction from multiple views. *Proc. CVPR*, pages 347–354, 2001.

[105] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cel animation. In *Proc. SIGGRAPH*, pages 243–250, August 1997.

[106] L. Zelnik-Manor and P. Perona. Automating joiners. In *Symp. on Non-Photorealistic Animation and Rendering (NPAR)*, pages 121–131, 2007.

[107] L. Zelnik-Manor, G. Peters, and P. Perona. Squaring the circle in panoramas. In *Proc. ICCV*, pages 1292 –1299 Vol. 2, 2005.

[108] D. Zorin and A. H. Barr. Correction of geometric perceptual distortions in pictures. In *Proc. SIGGRAPH*, pages 257–264, 1995.

Appendix A

VIEWPOINT SCORING IN INTERNET PHOTO COLLECTIONS

This scoring function was proposed by Noah Snavely and I refer the reader to his thesis for more details [92].

Assume that we are given a database of input images $\mathcal{I}$ whose camera parameters (intrinsics and extrinsics) have been computed. The term *camera* denotes the viewing parameters of an input image. The term *image* denotes an *input* photo I, with associated camera, from the database. The term *view* denotes an *output* photo v that we seek to render. A view is produced by *reprojecting* an input photo, through a rendering process, to the desired new viewpoint.

Define a *reprojection score* $S(I, v)$ that rates how well a database image I can be used to render a new view v. The best reprojection is obtained by maximizing $S(I, v)$ over the database, yielding a *viewpoint score* $S(v)$:

$$S(v) = max_I S(I, v) \tag{A.1}$$

Ideally, $S(I, v)$ would measure the difference between the synthesized view and a real photo of the same scene captured at v. Because we do not have access to the real photo, we instead use the following three criteria:

1. Angular deviation: the relative change in viewpoint between I and v should be small.
2. Field of view: the projected image should cover as much of the field of view as possible in v.

3. Resolution: I should be of sufficient resolution to avoid blur when projected into v.

For a given image and viewpoint, each of these criteria is scored on a scale from 0 to 1. To compute these scores, we require a geometric proxy plane for each image in order to reproject that image into a view v. Please refer [92] for details on how the proxy plane is selected.

The angular deviation score $S_{\text{ang}}(I, v)$ is proportional to the angle between rays from the current viewpoint through a set of points in the scene and rays from image I through the same points. This is akin to the *minimum angular deviation* measure used in Unstructured Lumigraph Rendering [13]. Rather than scoring individual rays, we score the entire image by averaging the angular deviation over a set of 3D points observed by I. These points, $\text{Pts}(I)$, are selected for each image in a pre-processing step. To ensure that the points are evenly distributed over the image, we take the set of 3D points observed by I, project them into I, and sort them into a 10×10 grid of bins defined on the image plane, then select one point from each non-empty bin. The average angular deviation is computed as:

$$S_{\text{ang}}'(I, v) = \frac{1}{n} \sum_{p \in \text{Pts}(I)} \text{angle}(p - \mathbf{p}(I), p - \mathbf{p}(v)). \tag{A.2}$$

where $\mathbf{p}(I)$ is the 3D position of camera I, $\mathbf{p}(v)$ is the 3D position of v, and $\text{angle}(a, b)$ gives the angle between a and b. The average deviation is clamped to a maximum value of $\alpha_{\max}$ (we set $\alpha_{\max} = 12°$), and mapped to the interval $[0, 1]$:

$$S_{\text{ang}}(I, v) = 1 - \frac{\min(S_{\text{ang}}'(I, v), \alpha_{\max})}{\alpha_{\max}}. \tag{A.3}$$

The field-of-view score $S_{\text{fov}}(I, v)$ is computed by projecting I onto its proxy geometry plane, then into v, and computing the area of the view that is covered by the reprojected image. We use a weighted area, with higher weight in the center of the view, as we found that it is generally more important to cover the center of the view

than the boundaries. The weighted area is computed by dividing the view into a grid of cells, $\mathcal{G}$, and accumulating weighted contributions from each cell:

$$S_{\text{fov}}(I, v) = \sum_{G_i \in \mathcal{G}} w_i \frac{\text{Area}(\text{Project}(I, v) \cup G_i)}{\text{Area}(G_i)}, \tag{A.4}$$

where $\text{Project}(I, v)$ is the polygon resulting from reprojecting I into v (if any point of the projected image is behind the view, Project returns the empty set).

Finally, the resolution score $S_{\text{res}}(I, v)$ is computed by projecting I into v and finding the average number of pixels of I per screen pixel. This is computed as the ratio of the number of pixels in I to the area, in screen pixels, of the reprojected image $\text{Project}(I, v)$:

$$S_{\text{res}}^{'}(I, v) = \frac{\text{Area}(I)}{\text{Area}(\text{Project}(I, v))}. \tag{A.5}$$

If this ratio is greater than 1, then, on average, the resolution of I is sufficient to avoid blur when I is projected onto the screen (we use mip-mapping to avoid aliasing). We then transform S_{res} to map the interval $[\text{ratio}_{\text{min}}, \text{ratio}_{\text{max}}]$ to $[0, 1]$:

$$S_{\text{res}}(I, v) = \text{clamp}\left(\frac{S_{\text{res}}{}'(I, v) - \text{ratio}_{\text{min}}}{\text{ratio}_{\text{max}} - \text{ratio}_{\text{min}}}, \epsilon, 1\right), \tag{A.6}$$

where $\text{clamp}(x, a, b)$ clamps x to the range $[a, b]$. We use values of 0.2 and 1.0 for $\text{ratio}_{\text{min}}$ and $\text{ratio}_{\text{max}}$, and enforce a non-zero minimum resolution score ϵ because we favor viewing a low-resolution image rather than no image at all.

The three scores are multiplied to give the view score S:

$$S(I, v) = S_{\text{ang}}(I, v) \cdot S_{\text{fov}}(I, v) \cdot S_{\text{res}}(I, v). \tag{A.7}$$

VITA

Rahul Garg grew up in Delhi, India and received his B. Tech. in Computer Science and Engineering from Indian Institute of Technology (IIT), Delhi in 2007. He then joined the Ph.D. program of the Computer Science and Engineering department at the University of Washington, where he worked with Steven M. Seitz. In 2012, he received the Doctor of Philosophy degree.