ABSTRACT

Title of dissertation: FUSING MULTIMEDIA DATA INTO DYNAMIC VIRTUAL ENVIRONMENTS

Ruofei Du
Doctor of Philosophy, 2018

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

In spite of the dramatic growth of virtual and augmented reality (VR and AR) technology, content creation for immersive and dynamic virtual environments remains a significant challenge. In this dissertation, we present our research in fusing multimedia data, including text, photos, panoramas, and multi-view videos, to create rich and compelling virtual environments.

First, we present *Social Street View*, which renders geo-tagged social media in its natural geo-spatial context provided by 360° panoramas. Our system takes into account visual saliency and uses maximal Poisson-disc placement with spatiotemporal filters to render social multimedia in an immersive setting. We also present a novel GPU-driven pipeline for saliency computation in 360° panoramas using spherical harmonics (SH). Our spherical residual model can be applied to virtual cinematography in 360° videos. We further present *Geollery*, a mixed-reality platform to render an interactive mirrored world in real time with three-dimensional (3D) buildings, user-generated content, and geo-tagged social media. Our user study has identified several use cases for these systems, including immersive social storytelling, experiencing the culture, and crowd-sourced tourism.

We next present *Video Fields*, a web-based interactive system to create, calibrate, and render dynamic videos overlaid on 3D scenes. Our system renders dynamic entities from multiple videos, using early and deferred texture sampling. *Video Fields* can be used for immersive surveillance in virtual environments. Furthermore, we present *VRSurus* and *ARCrypt* projects to explore the applications of gestures recognition, haptic feedback, and visual cryptography for virtual and augmented reality.

Finally, we present our work on *Montage4D*, a real-time system for seamlessly fusing multi-view video textures with dynamic meshes. We use geodesics on meshes with view-dependent rendering to mitigate spatial occlusion seams while maintaining temporal consistency. Our experiments show significant enhancement in rendering quality, especially for salient regions such as faces. We believe that *Social Street View*, *Geollery*, *Video Fields*, and *Montage4D* will greatly facilitate several applications such as virtual tourism, immersive telepresence, and remote education.

FUSING MULTIMEDIA DATA INTO DYNAMIC VIRTUAL ENVIRONMENTS

by

Ruofei Du

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Dr. Amitabh Varshney, Chair/Advisor
Dr. Matthias Zwicker
Dr. Furong Huang
Dr. Joseph F. JaJa
Dr. Ming Chuang

Acknowledgments

With my deepest sincerity and gratitude, thank you everyone I met in the incredible five years I have had at *University of Maryland, College Park* and *Microsoft Research*.

First and foremost, I have to say a special thank you to my advisor, *Professor Amitabh Varshney* for advising me, supporting me, and having faith in me on the creative and fascinating projects over the past years. I am not the most conventional of researchers, and there is often disruption along the path to achieving successful projects, but *Professor Varshney* was always in the background supporting me. Thank you for helping me to grow into the researcher I am today. With your encouragement, I have learned a great deal of knowledge about interactive graphics and visualization, virtual and augmented reality, parallel computing, and, most importantly, how to conduct research and behave myself.

I would like to deeply thank my committee members, *Dr. Zwicker*, *Dr. Huang*, *Dr. JaJa*, and *Dr. Chuang*, for offering me invaluable advice and direction. I am also grateful to my advisors in Human Computer Interaction, *Dr. Froehlich* and *Dr. Findlater*, for guiding me through *ProjectSideWalk* and *ProjectHandSight*, teaching me how to organize a team project, how to take human factors into account, and how to think out of the box. I owe my gratitude to all the teachers and classmates I have learned from and because of whom my graduate experience has been one that I will cherish forever.

I owe my thanks to all my colleagues, collaborators, and mentors from *Mi-*

crosoft Research, Redmond: *Hugues Hoppe*, *Wayne Chang*, *Sameh Khamis*, *Shahram Izadi*, *Mingsong Dou*, *Yury Degtyarev*, *Philip Davidson*, *Sean Fanello*, *Adarsh Kowdle*, *Sergio Orts Escolano*, *Christoph Rhemann*, *David Kim*, *Jonathan Taylor*, *Pushmeet Kohli*, *Vladimir Tankovich*, *Marek Kolwalski*, *Qiyu Chen*, *Spencer Fowers*, *Jeff Foster*, *Norm Whittaker*, and *Ben Cutler*. They live and breathe the spirit of research that I know and love: to work hard, to embrace science and engineering, to mix theory and practice, to think and work as a team rather than individuals, and to always focus on big things.

I am grateful to all my dear friends, colleagues, collaborators, and lab-mates at *University of Maryland, College Park*: *Dr. Sujal Bista*, *Dr. Hsueh-Chien Cheng*, *Dr. Changqing Zou*, *Dr. Eric Krokos*, *Dr. Kotaro Hara*, *Dr. Lee Stearns*, *Dr. Uran Oh*, *Dr. Hui Miao*, *Dr. Hao Li*, *Dr. Fan Du*, *Dr. Awalin Sopan*, *Dr. Xintong Han*, *Dr. Zebao Gao*, *Dr. Jin Sun*, *Yu Jin*, *Xiaoxu Meng*, *Shangfu Peng*, *Hong Wei*, *Liang He*, *Tiffany Chao*, *Kent Wills*, *Max Potasznik*, *Zheng Xu*, *Xuetong Sun*, *Hao Zhou*, *Xiyang Dai*, *Weiwei Yang*, *Shuo Li*, *Sida Li*, *Eric Lee*, *Somay Jain*, *Mukul Agarwal*, *Patrick Owen*, *Tara Larrue*, and *David Li*. I am lucky to collaborate with you and learn from you. Your passion and dedication will always inspire me.

Finally, there are three people missing up to now who I should thank the most: my parents and my wife, *Sai Yuan*. They have supported my continual dedication to study and research during the late nights, the weekends, and the travel.

Thank you all!

Dedication

2D justifies existence

3D validates identification

4D convinces living

To my advisors, teachers, friends, and families,

who taught me theorems, algorithms, data structures, and the meaning of life,

and to those who taught me to relish the moment in the true reality.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
4D	Four-Dimensional
AR	Augmented Reality
CUDA	Compute Unified Device Architecture
EXIF	Exchangeable Image File Format
FPS	Frames Per Second
GMM	Gaussian Mixture Models
GPU	Graphics Processing Unit
GPGPU	General-Purpose computing on Graphics Processing Units
GSV	Google Street View
GUI	Graphical User Interface
HCI	Human Computer Interaction
HMD	Head-Mounted Display
IMU	Inertial Measurement Unit
MR	Mixed Reality
MRF	Markov Random Fields
OpenCV	Open source Computer Vision library
OpenGL	Open Graphics Library
OpenMP	Open Multi-Processing
RMSE	Root Mean Square Error
SH	Spherical Harmonics
SSR	Spherical Spectral Residual
SSV	Social Street View
SQL	Structured Query Language
VR	Virtual Reality
VC	Visual Cryptography
WebGL	Web Graphics Library

Chapter 1: Introduction

With recent advances in the commoditization of virtual and augmented reality (VR and AR) displays, there is an increasing demand for three-dimensional (3D) content. But where will the massive amount of 3D data come from? Manually crafting dynamic 3D scenes will not be the ultimate solution, since it usually requires tremendous efforts from professional artists and designers. In this dissertation, we present our research in automatically fusing multimedia data, including text, images, 360° panoramas, and multiview videos, into dynamic virtual environments. We devise novel system architectures, visualization strategies, and rendering techniques to address the challenges for each type of data with real-time constraints. Our research may catalyze several VR and AR applications: mixed-reality social platforms, immersive surveillance, and real-time telepresence.

1.1 Social Street View: Blending Immersive Maps with Geotagged Social Media

In Chapter 2, we present *Social Street View* [1], the first interactive system for rendering geotagged social media in its natural geospatial context provided by immersive maps. An overview is shown in Figure 1.1.

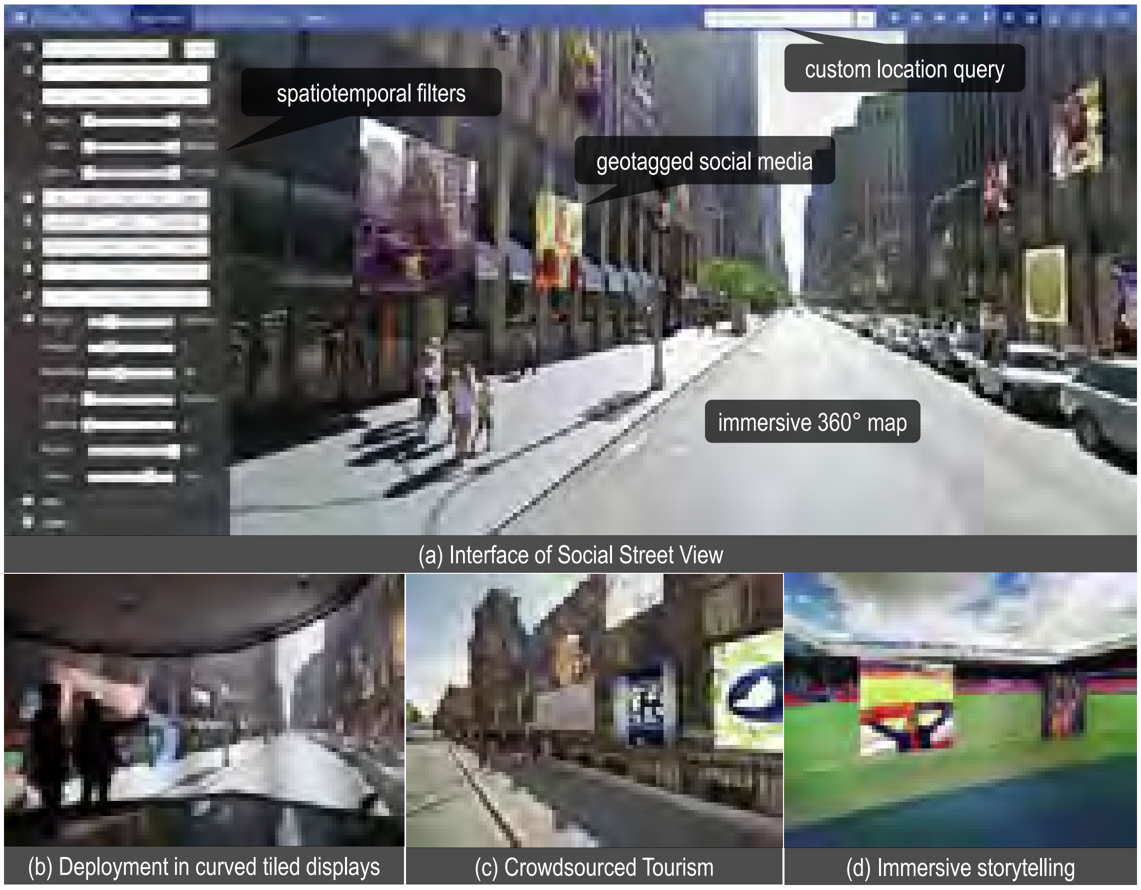


Figure 1.1: Interface, deployment, and use cases of *Social Street View*. (a) shows the graphic user interface of *Social Street View*. Given a custom location query with spatiotemporal filters, our system automatically fetches geotagged social media from *Twitter* and *Instagram* and immersive maps from *Google Street View*. The social media images are aligned with building geometry and laid out aesthetically with the immersive map. (b) shows the deployment in an immersive curved screen environment with 15 projectors. Users can interactively explore hundreds of social media messages near a New York City street with a game controller at a resolution of 6K $\times$ 3K pixels. (c) shows a use case for crowdsourced tourism, where users can look through the museum in Paris for geotagged artworks inside as well as the dishes in nearby restaurants. (d) shows an example for immersive storytelling, where geotagged photographs of a football game are embedded into the 360° stadium where the event occurred. Please refer to the supplementary materials at `http://socialstreetview.com` for more examples.
.

Social Street View consists of a street view scraper, a social media scraper, distributed SQL databases, a web-server powered by Apache and PHP, and optional modules such as a temporal filter, a geolocation filter, and a machine-learning-based face filter. Given a requested location, our system first creates a 3D world using the tiles of panoramic data, depths maps, normal maps, and road orientations from *Google Street View*[1]. Taking the number and the total area of the surrounding building surfaces into account, *Social Street View* classifies the immersive maps into urbanscapes or landscapes. For urbanscapes, our system takes advantage of the depth and normal maps, visual saliency, and maximal Poisson-disc placement sampling to render social multimedia in an aesthetically pleasing manner with geospatial renderings. For landscapes, our system uses the road orientations to place social media alongside the street. Furthermore, we present a system architecture that is able to stream geotagged social media and render it across a range of display platforms spanning tablets, desktops, head-mounted displays, and large-area room-sized curved tiled displays. We explore several potential use cases including immersive social storytelling, learning about the culture, and crowdsourced tourism.

1.1.1 TopicFields: Spatiotemporal Visualization of Geotagged Social Media With Hybrid Topic Models and Scalar Fields

While *Social Street View* offers an immersive street-level interface that interleaves visual navigation of our surroundings with social media content, how can we

[1]Google Street View: `http://www.google.com/maps/streetview`

acquire an overview of the geotagged social media? What are the dominant topics over time and how are they distributed on the map? To address these questions, in Section 2.8, we present *TopicFields*, an interactive system to explore, aggregate, and visualize geotagged social media using hybrid topic models, scalar fields, and stream graphs. An overview of *TopicFields* is shown in Figure 1.2.

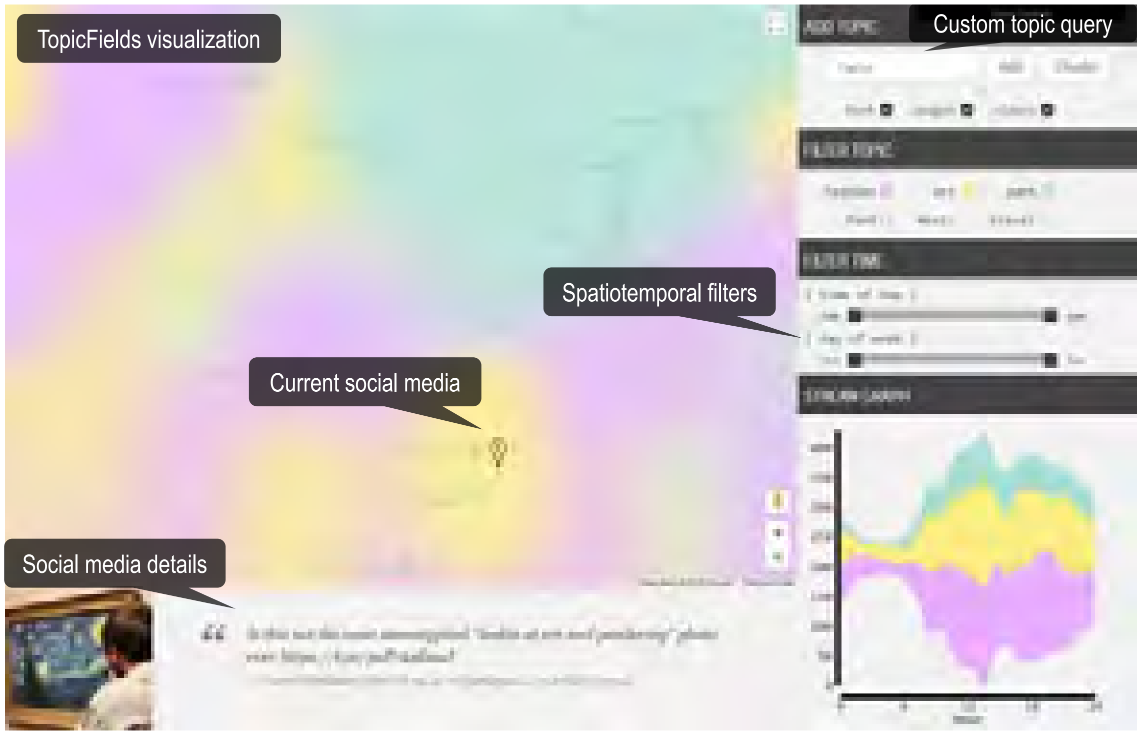


Figure 1.2: Interface and visualization of social media in Manhattan district from *TopicFields* The top-left map view is overlaid with an interactive scalar map, with each color indicating a different topic: fashion, art, or park. We show the ground truth labels such as *Central Park* and *The Museum of Modern Art* from *Google Maps* for reference. The detail view at the bottom shows the corresponding social media text, image, or video, as the user explores and clicks on the map. The control panel on the right allows the user to select, add, and modify the topics to explore. The stream graph shows the volume of social media of different topics over the queried time.
.

In the data processing stage, we apply two machine learning models, *Word2Vec* [2] and *Inception-v3* [3], to the data and address the relationships among the extracted topics by rearranging them via spectral ordering. In the visualization stage, we allow users to interactively select the preferred topics and alter the transfer function for visualizing the social media on a map. Our system, *TopicFields*, can efficiently estimate the kernel density distribution and visualize the scalar fields of the requested topics on a map on the GPU. In addition, we use temporal filters and stream graphs to enhance the comprehensibility of the data over time. We further demonstrate the effectiveness of *TopicFields* with potential use cases such as trip planning and event discovery.

1.1.2 Geollery: Designing an Interactive Mirrored World with Geotagged Social Media

In *Social Street View*, the user interaction is limited to street-level panoramas. What if we could create an interactive 3D mirrored world which has the same availability of the 2D map? Motivated by this challenge, we have developed *Geollery*, an interactive mixed-reality platform for creating, sharing, and exploring geotagged social media (Figure 1.3).

Geollery introduces a real-time pipeline to progressively render an interactive mirrored world with 3D buildings, user-generated content, and external geotagged social media. This mirrored world allows users to see, chat, and collaborate with remote participants with the same spatial context in an immersive environment. We conduct a user study with semi-structured interviews to qualitatively compare *Geollery* with *Social Street View*. From a Welch's paired t-test, we found a signifi-

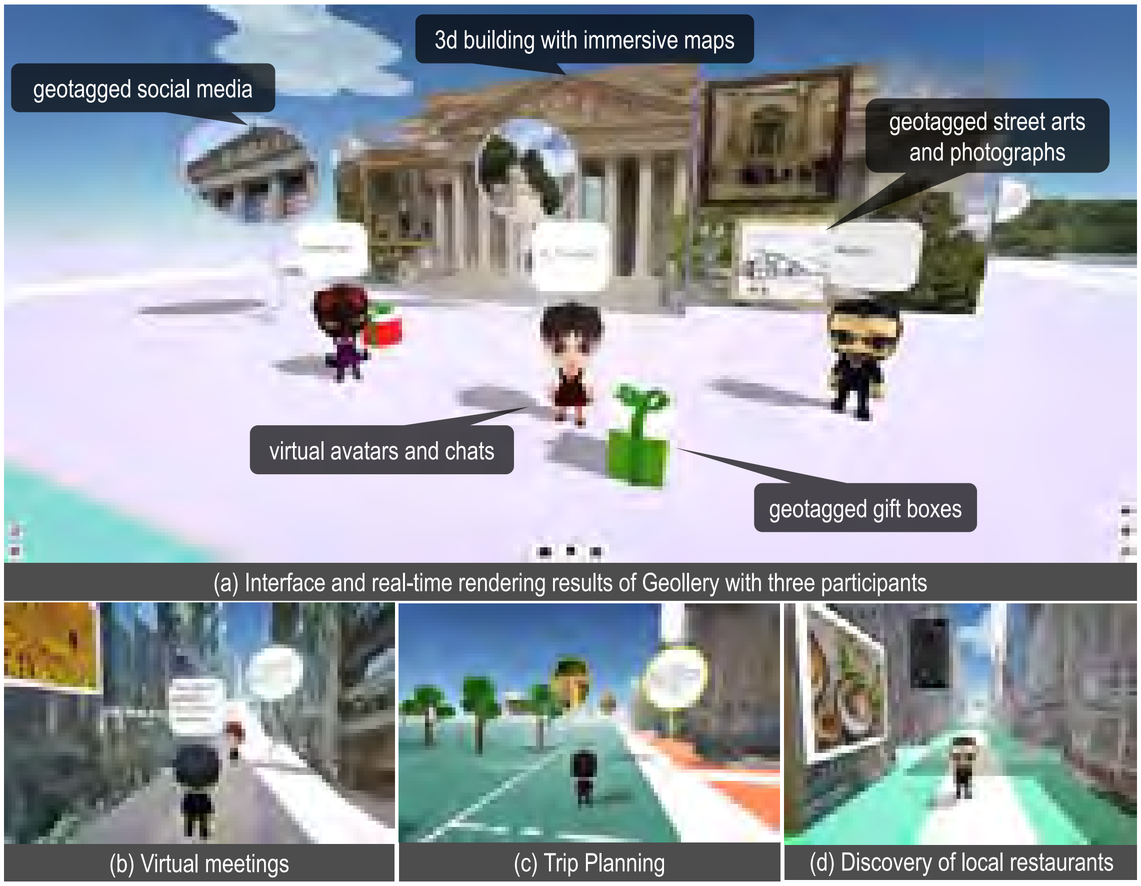

3d building with immersive maps
geotagged social media
geotagged street arts and photographs
virtual avatars and chats
geotagged gift boxes
(a) Interface and real-time rendering results of Geollery with three participants
(b) Virtual meetings
(c) Trip Planning
(d) Discovery of local restaurants

Figure 1.3: Interface, rendering results, and use cases of *Geollery* (a) shows the interface and real-time rendering results of *Geollery* with three participants. *Geollery* progressively fuses 3D buildings, virtual avatars, and geotagged social media into a mirrored world. (b), (c), and (d) show three potential applications: virtual meeting, trip planning, and discovery of local business and events.

cant effect for interactivity ($t(20) = 3.04, p < 0.01, \text{Cohen's } d = 0.83$) and creativity ($t(20) = 2.10, p < 0.05, \text{Cohen's } d = 0.66$) with *Geollery* outperforming *Social Street View*. We point out that data sources, interactivity, immersive textures, and customization play significant roles in designing an interactive mirrored world with geotagged social media. User feedback from our study reveals several use cases for *Geollery* including travel planning, virtual meetings, and discovery of local business. Please refer to `https://www.geollery.com` for a live demonstration.

1.2 Spherical Harmonics for Saliency Computation and Virtual Cinematography in 360° Videos

When applying the saliency metrics to the immersive maps in *Social Street View*, we found that classic saliency may not work directly on 360° images. The results are inconsistent with horizontal translation and spherical rotations for the same 360° image. In Chapter 3, we present a novel GPU-driven pipeline for saliency computation and virtual cinematography in 360° videos using spherical harmonics (SH).

First, we present the preprocessing for computing the SH coefficients for representing the 360° videos. Our pipeline pre-computes a set of the Legendre polynomials and SH functions and stores them in GPU memory. We adopt the highly-parallel prefix sum algorithm to integrate feature maps of the downsampled 360° frames as 15 bands of spherical harmonics coefficients on the GPU.

Next, by analyzing the spherical harmonics spectrum of the 360° video, we extract the spectral residual by accumulating the SH coefficients between a low band and a high band. As shown in Figure 1.4, our spherical spectral residual (SSR) model reveals the multi-scale saliency maps in the spherical spectral domain and reduces the computational cost by discarding the low bands of SH coefficients. From the experimental results, our SSR model outperforms the classic Itti *et al.*'s model by 5× to 13× in timing and runs at over 60 FPS for $4K$ videos.

Finally, our interactive computation of spherical saliency can be used for

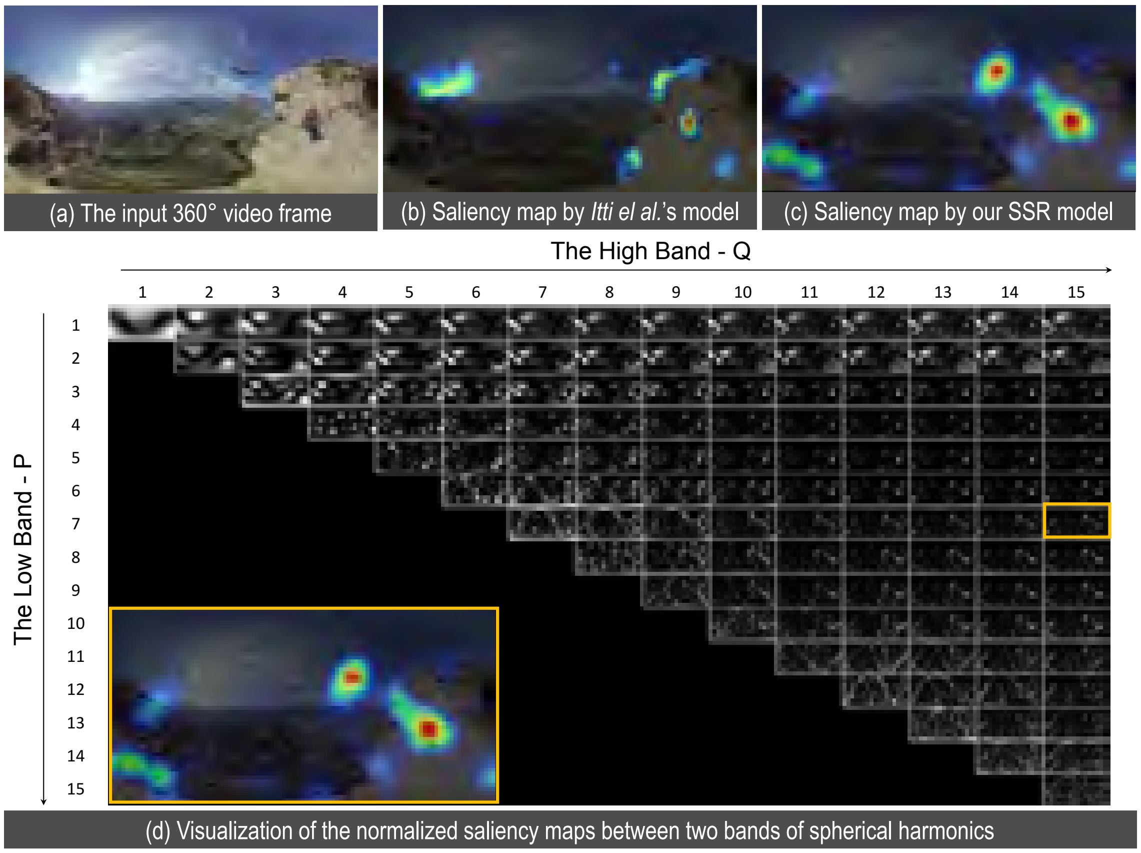

(a) The input 360° video frame
(b) Saliency map by *Itti el al.*'s model
(c) Saliency map by our SSR model
The High Band - Q
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
The Low Band - P
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
(d) Visualization of the normalized saliency maps between two bands of spherical harmonics

Figure 1.4: (a) shows an input frame, (b) shows the saliency map by Itti *et al.*'s model, (c) shows the saliency map by our mode, and (d) shows the spectral residual maps between a low band and a high band of spherical harmonics. The number along the horizontal axis indicates the high band Q, while the vertical axis indicates the low band P. Note that the saliency maps within or close to the orange bounding box successfully detect the two people in the frame.

saliency-guided virtual cinematography in 360° videos. We formulate a spatiotemporal model to ensure large saliency coverage and smooth camera movement. Since the frame rate of the video may be lower than the display rate of the displays, we further employ a spline interpolation amongst the rotational quaternions of the virtual camera, as shown in Figure 1.5(a). Compared with the baseline model which only maximizes the saliency coverage, our model significantly reduces the camera movement jitter, as shown in as shown in Figure 1.5(b).

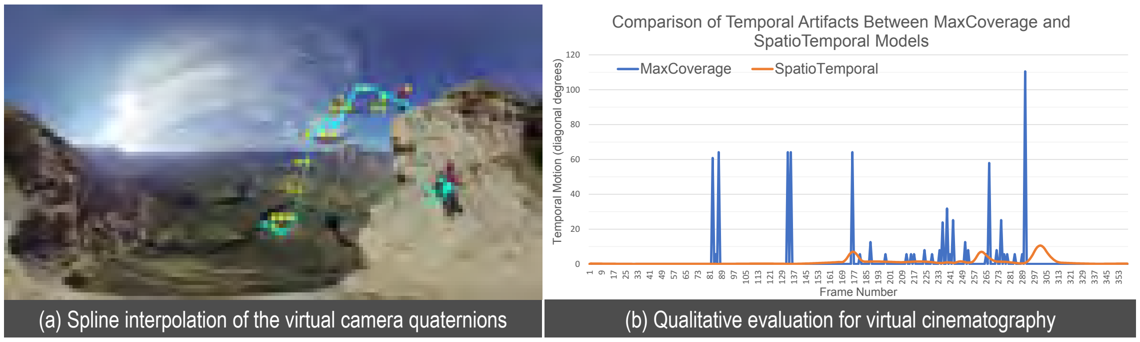


Figure 1.5: (a) shows the smooth spline interpolated path of the virtual camera quaternions. (b) shows the qualitative comparison between the MaxCoverage model and the SpatioTemporal model for virtual cinematography.

1.3 Video Fields: Fusing Multiple Surveillance Videos into a Dynamic Virtual Environment

In Chapter 4, we present *Video Fields*, a novel web-based interactive system to create, calibrate, and render dynamic video-based virtual reality scenes in head-mounted displays, as well as high-resolution wide-field-of-view tiled display walls.

Video Fields system consists of a camera world calibration interface, a back-end server to process and stream the videos, and a web-based rendering system. As shown in Figure 1.6(a), *Video Fields* requires input of customized 3D models and multiple surveillance videos. In the calibration interface, we provide a four-stage workflow for the user: importing and synchronizing the surveillance videos, defining the ground projection plane, draw initial building geometries, and adjusting the camera positions and rotating quaternions.

To estimate robust background images, we take advantage of Gaussian Mixture Models (GMM) for background modeling. Compared with the mean filter or the

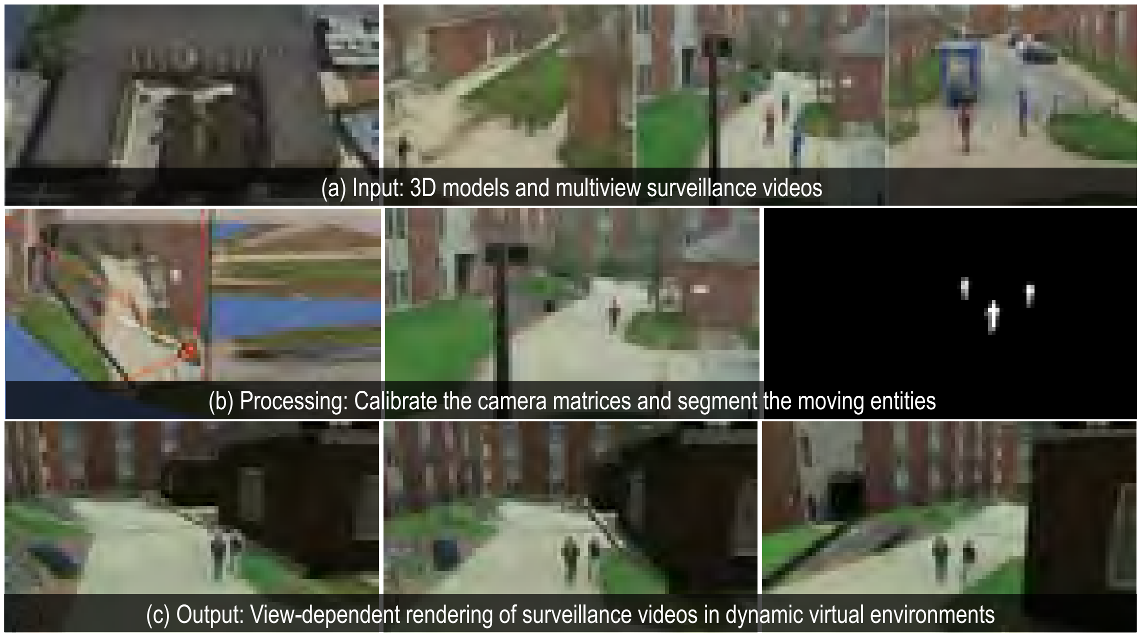

(a) Input: 3D models and multiview surveillance videos
(b) Processing: Calibrate the camera matrices and segment the moving entities
(c) Output: View-dependent rendering of surveillance videos in dynamic virtual environments

Figure 1.6: *Video Fields* system fuses multiple videos, camera-world matrices from a calibration interface, static 3D models, as well as satellite imagery into a novel dynamic virtual environment. *Video Fields* integrates automatic segmentation of moving entities during the rendering pass and achieves view-dependent rendering in two ways: early pruning and deferred pruning. *Video Fields* takes advantage of the WebGL and WebVR technology to achieve cross-platform compatibility across smart phones, tablets, desktops, high-resolution tiled curved displays, as well as virtual reality head-mounted displays. See the supplementary video at `http://videofields.com`

Kalman filter, GMM is more adaptive with different lighting conditions, repetitive motions of scene elements, as well as moving entities. After learning the background model for each video, we achieve real-time interactive segmentation of moving entities during the fragment-shader rendering pass by taking advantage of the many-core computing on the GPU. An example is shown in Figure 1.6(b).

Finally, we present two methods to render video fields: early pruning and deferred pruning. In the *Early Pruning* approach, we discard the pixels that do not belong to the foreground as soon as the foreground is identified. In the *Deferred Pruning* approach, we dynamically project videos on moving billboards for better anti-aliasing, bi-linear sampling, and faster visibility testing. As shown in Figure 1.6(c), we use view-dependent rendering techniques to render the moving entities in different perspectives. The user can also adjust the opacity of every object, thus allowing themselves to "see through the buildings".

We recorded three 10-minute video clips with the resolution of 1280×720 pixels and tested both the early- and deferred-pruning algorithms in three settings. The experimental results indicate that the early pruning approach is more efficient than the deferred pruning. Nevertheless, the deferred pruning achieves better results through anti-aliasing and bi-linear interpolation. We envision the use of the system and algorithms introduced in *Video Fields* for immersive surveillance monitoring in virtual environments.

1.4 Integrating Gesture Recognition, Tactile Feedback, and Visual Cryptography into Virtual Environments

Dynamic virtual environments can further be enhanced with gestures, haptic feedback, and visual cryptography. In Chapter 5, we present our research in gesture recognition, tactile feedback *(VRSurus)*, and visual cryptography *(ARCrypt)* for VR and AR technologies.

1.4.1 VRSurus: Enhancing Interactivity and Tangibility of Puppets in Virtual Reality

VRSurus is a smart device designed to recognize the puppeteer's gestures and render tactile feedback to enhance the interactivity of physical puppets in virtual reality. As shown in Figure 1.7, *VRSurus* is wireless, self-contained, and small enough to be mounted upon any puppets.

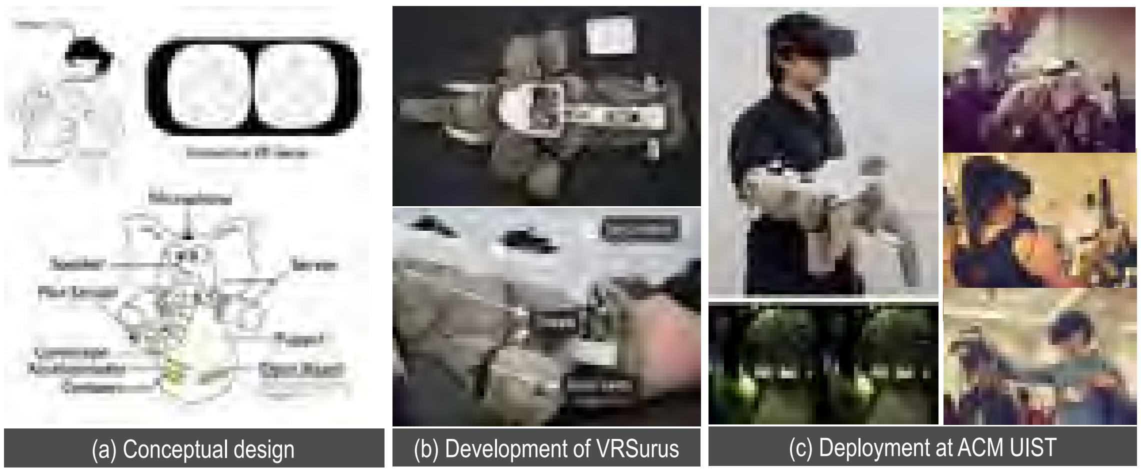


Figure 1.7: (a) shows the conceptual design of *VRSurus*, where a puppeteer could control an elephant puppet in both physical and virtual reality and receive haptic feedback. (b) shows the research prototype of *VRSurus*: a custom 3D-printed cap that encapsulates the Arduino microcontroller, an inertial measurement unit, and other electronic modules. (c) shows the deployment of *VRSurus* at *ACM UIST 2015 Student Innovation Contest*, where 63 participants including two K12 specialists tried out our device. See the supplementary video, open sourced software and hardware at `http://videofields.com`

For gesture recognition, we train the classifier using the decision tree algorithm in Weka [4]. In total, we facilitate the following sixteen features: the sum of mean values on all axes, the differences between each pair of axes, the power of each axis, the range of each axis, the cross product between every two axes and the standard deviation of values along each axis. To recognize the four gestures: idle, swiping, shaking and thrusting, we collected 240 sets of raw accelerometer values for each gesture from 4 lab members (60 sets per gesture per person). For haptic feedback, we install various actuators, (*e.g.*, solenoids, servos, and vibration motors) to provide tactile feedback on the puppeteer's forearm and animate the physical puppetry.

Finally, we deployed VRSurus at *ACM UIST 2015 Student Innovation Contest* for 2.5 hours. In total, 63 participants from the UIST community including two invited K12 specialists tried out our device. There were more than a hundred people who watched our device as well as the gaming procedure. Our 3D models, circuitry and the source code are publicly available at `www.VRSurus.com`.

1.4.2 ARCrypt: Visual Cryptography with Misalignment Tolerance using Augmented Reality Head-Mounted Displays

ARCrypt uses augmented reality and visual cryptography with misalignment tolerance to safeguard the information depicted on ordinary displays. As shown in Figure 1.8, *ARCrypt* partitions the confidential data into two shares. It sends one to a Microsoft HoloLens and the other one to a regular display. Even if one share of the data is stolen by hackers, the original, complete, data cannot be restored from it. Only the recipient with both shares can visually decrypt the data by fusing the two shares of images through HoloLens.

However, head jittering is identified as one of the greatest challenges for users to fuse the two images in head-mounted displays [5]. In *ARCrypt*, we model the

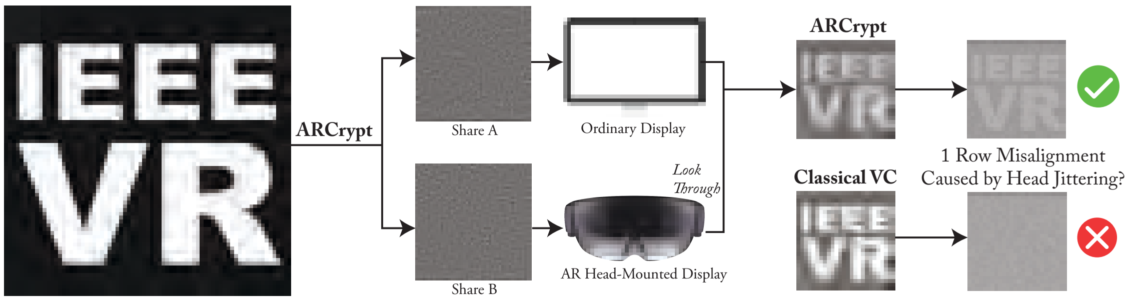


Figure 1.8: Results and overview of our system, *ARCrypt*. *ARCrypt* is able to split a confidential message into two shares of images, which guarantees that the original information could not be revealed with either share of the image alone. The *ARCrypt* system then transmits the two shares to an ordinary display and an augmented reality head-mounted display, respectively. When the user looks through the two aligned images, the secret message is revealed directly to the user's human visual system. Nevertheless, head jittering may cause the two images to misalign with each other. The proposed *ARCrypt* algorithm outperforms the original visual cryptography algorithm in the presence of one or two rows of misalignment.

probability of misalignment caused by head jittering as a Gaussian distribution. Each foreground pixel has a probability to be misaligned with one of its surrounding pixels; in this way, when the two shares match perfectly, the background is unchanged, but the foreground is darker. Therefore, the new algorithm enables the visual cryptography to be tolerant of misalignment when using AR head-mounted displays.

1.5 Montage4D: Real-time Seamless Fusion and Stylization of Multiview Video Textures

In Chapter 6, we present *Montage4D* as the successor of *Video Fields*. *Montage4D* is a practical solution towards real-time seamless texture montage for dynamic multiview reconstruction. We illustrate its work flow in Figure 1.9.

We build on the ideas of dilated depth discontinuities and majority voting from

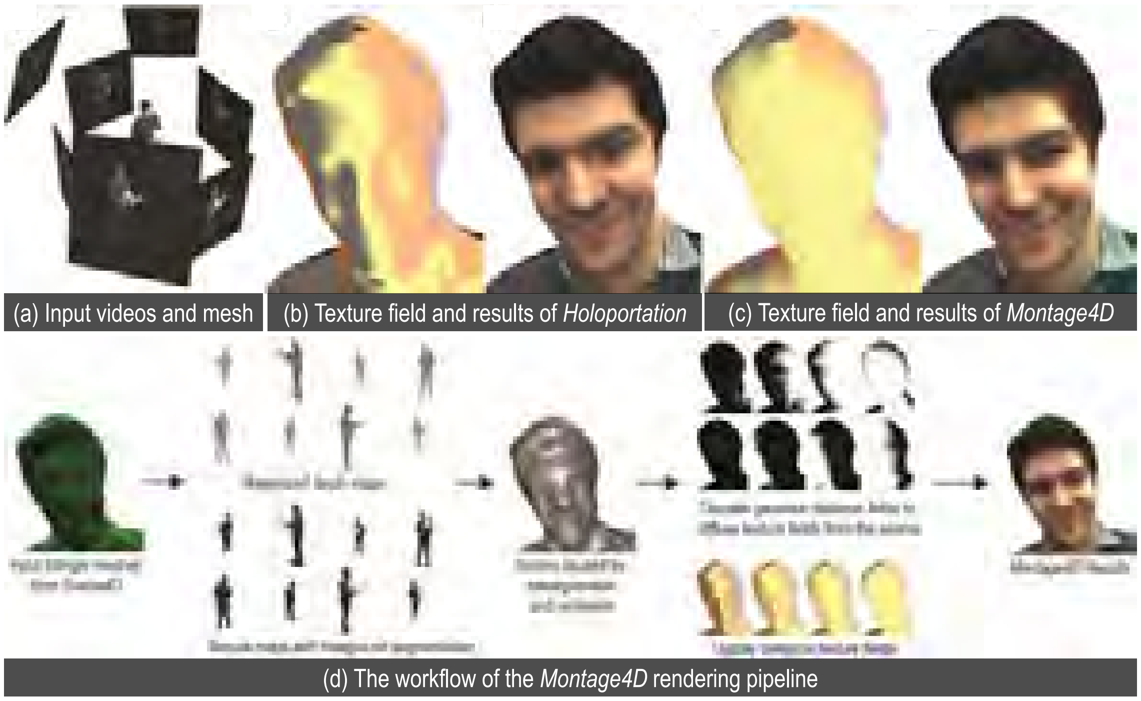

(a) Input videos and mesh
(b) Texture field and results of *Holoportation*
(c) Texture field and results of *Montage4D*
(d) The workflow of the *Montage4D* rendering pipeline

Figure 1.9: Results and work flow of the *Montage4D* system. (a) shows the input multiview videos and mesh, (b) shows the texture field and results of *Holoportation*, (c) shows the texture field and results of *Montage4D*, where we dramatically mitigate seams and blurring issues, and (d) shows the work flow of the *Montage4D* rendering pipeline.

Holoportation to reduce ghosting effects when blending textures. In contrast to their approach, we determine the appropriate blend of textures per vertex using view-dependent rendering techniques, so as to avert fuzziness caused by the ubiquitous normal-weighted blending.

We further identify the potential causes of visible seams: self-occlusion, mis-projected colors, and field-of-view boundaries. For the datasets acquired for real-time telepresence applications, we have observed the fraction of seam triangles to be less than 1%. This observation has guided us to process the triangles adjacent to the seams, using a propagation procedure by calculating the geodesics directly on the GPU. We follow a variant of the highly efficient approximation algorithm described in [6] to efficiently diffuse the texture fields using the geodesic distance fields. To prevent the texture weights from changing too fast during view transitions, we leverage temporal texture fields to mitigate spatial occlusion seams while preserving temporal consistency.

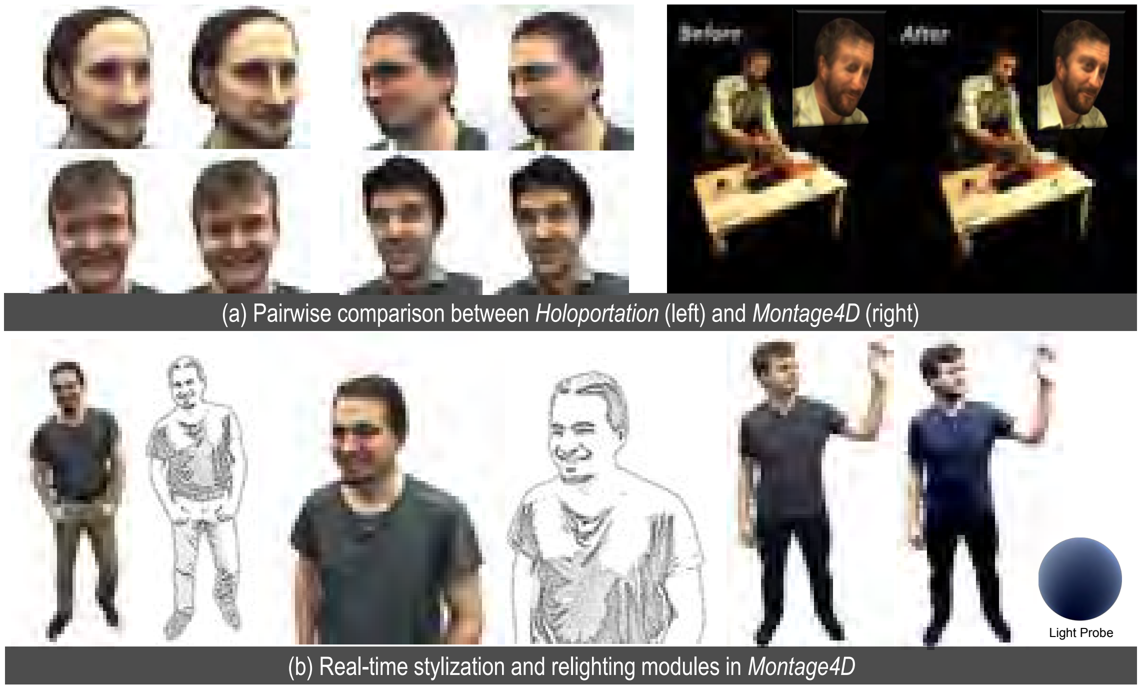

Before
After
(a) Pairwise comparison between *Holoportation* (left) and *Montage4D* (right)
Light Probe
(b) Real-time stylization and relighting modules in *Montage4D*

Figure 1.10: (a) shows the pairwise comparative results between *Holoportation* (left) and *Montage4D* (right). (b) shows examples of the real-time stylization and relighting components in the *Montage4D* system.

As shown in Figure 1.10(a) and Table 1.1, the experimental results demonstrate significant enhancement in rendering quality, especially in detailed regions such as faces. Furthermore, in Figure 1.10(b), we present our research towards real-time stylization and relighting to empower the Holoportation users to interactively stylize live 3D content. We envision that *Montage4D* will greatly facilitate a wide range of applications, including immersive business meetings, family gathering, and remote education. Please refer to `www.montage4d.com` for the supplementary materials.

Table 1.1: Comparison between *Holoportation* and *Montage4D* in cross-validation experiments. Note that *Montage4D* outperforms the Holoportation approach in visual quality while maintaining an average frame rate of over 100 Hz.

Dataset	Frames	#tri	*Holoporation*			*Montage4D*		
			SSIM	PSNR	FPS	SSIM	PSNR	FPS
Timo	837	251K	0.9805	38.60dB	227.2	0.9905	40.23dB	135.0
Yury	803	312K	0.9695	39.20dB	222.8	0.9826	40.52dB	130.5
Sergio	837	404K	0.9704	29.84dB	186.8	0.9813	30.09dB	114.3
Girl	1192	367K	0.9691	36.28dB	212.56	0.9864	36.73dB	119.4
Julien	526	339K	0.9511	33.94dB	215.18	0.9697	35.05dB	120.6

Chapter 2: Social Street View: Blending Immersive Street Views with Geotagged Social Media

2.1 Introduction

Social media plays a vital role in our lives because of its interactivity, versatility, popularity, and social relevance. Every day, billions of users create, share, and exchange information from their lives among their social circles [7]. Social media spans several modalities that include text, photos, audio, videos, and even 3D models. In addition to the content visible on the social networks, social media also consists of metadata that is useful for understanding the relationship amongst users, sentiment mining, and propagation of influence. Specifically, digital photographs often embed metadata such as the time of creation, the location of creation (through GPS coordinates), and camera parameters, that are included in the EXIF data fields. In spite of the availability of such rich spatiotemporal metadata, the current-generation social media content is most often visualized as a linear narrative, rarely in a 2D layout, and almost never in a natural immersive space-time setting. For small screens of mobile devices, perhaps a linear narrative ordered by time or relevance, makes the most sense given the limitations of interaction modalities and display real-estate. However, in immersive virtual environments such as those afforded by virtual and augmented reality headsets, a spatiotemporal view of the social media in a mixed reality setting may be the most natural.

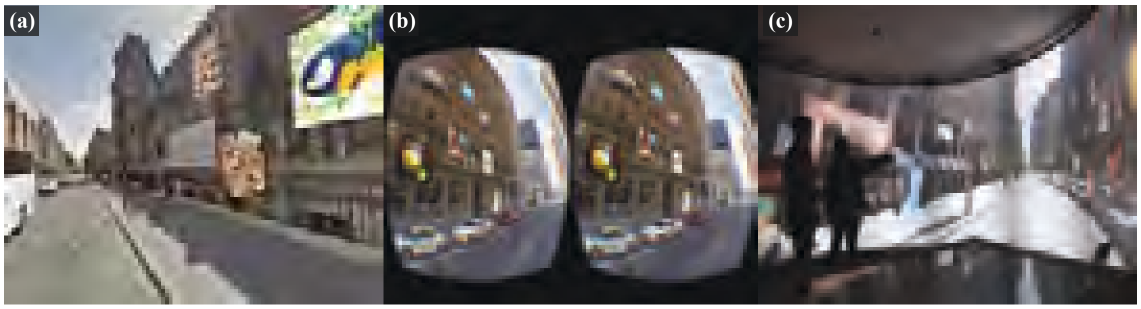


Figure 2.1: Results from our system, *Social Street View*. (a) shows the rendering results in a regular display. Users can look through the museum in Paris for ° artworks inside as well as the dishes in nearby restaurants. (b) shows the stereo rendering results in a VR headset. Geotagged images are automatically aligned with building geometry and laid out aesthetically. (c) shows the deployment in an immersive curved screen environment with 15 projectors. Users can explore hundreds of social media messages near a New York city street at a resolution of 6K × 3K pixels. Please refer to the supplementary video at `http://socialstreetview.com` .

Immersive interfaces that interleave visual navigation of our surroundings with social media content have not yet been designed. *NewsStand* [8], *Flickr* [9], and *Panoramio*[1] have taken important first steps towards this goal by using a user's geolocation information on 2D maps to display content. Still, we have not come across any system that (a) enables user exploration of social media in an immersive 3D spatial context in real time, and (b) allows temporal filtering of social messages in their spatial setting. Such a system will facilitate new genres of social interactions in spatial contexts mediated through virtual and augmented reality. These could be widely adopted in immersive social storytelling, learning about the culture, and

[1]Panoramio: `https://en.wikipedia.org/wiki/Panoramio`

crowd-sourced tourism.

As a proof-of-concept, we have developed a prototype system called the *Social Street View* (SSV) (Figure 3.1), the first immersive social media navigation system for exploring social media in large-scale urbanscapes and landscapes. Given a requested location, *SSV* builds a 3D world using tiles of panorama data from *Google Street View*[2] and *Bing Maps Streetside*[3], depths and normal maps, and road orientations. It then downloads geotagged data near the requested location from public-domain social media sites such as *Instagram* and *Twitter*. After building the 3D world, *SSV* renders the social media onto buildings or as virtual billboards along the road. The user can see photos of food uploaded by social media users next to the relevant restaurants, visual memories with friends in specific locations, identify accessibility issues on roads, as well as preview the coming attractions on scenic drives and nature hikes. The main contributions of our work are:

- conception, architecting, and implementation of *Social Street View*, a mixed reality system that can depict geotagged social media in an immersive 3D environment,
- blending multiple modalities of panoramic view metadata, including depth maps, normal maps, and road orientation, with social media metadata including GPS coordinates and the time of creation,
- enhancing visual augmentation by using maximal Poisson-disk sampling and image saliency metrics,

[2]Google Street View: `http://www.google.com/maps/streetview`
[3]Bing Maps Streetside: `http://www.bing.com/mapspreview`

- using WebGL and WebVR to achieve cross-platform compatibility across a range of clients including smartphones, tablets, desktop, high-resolution large-area wide-field-of-view tiled display walls, as well as head-mounted displays.

2.2 Background and Related Work

Our work builds upon rich literature of prior art on the creation of immersive maps as well as related work in visual management of geotagged information, analysis of geotagged social media and mixed reality in immersive maps.

2.2.1 Immersive Maps

In this chapter, we refer the term **immersive maps** to refer to online services that provide panoramic 360° bubbles at multiple way-points. Since Google Street View (GSV) debuted in 2007, several immersive maps have covered over 43 countries throughout the world. Most street views are captured using a car equipped with a spherical array of cameras. For places inaccessible to ordinary cars, volunteers and trekkers on foot, tricycle, trolley, camel, snowmobile, and even underwater apparatus capture immersive panoramas [10]. Therefore, the latest immersive maps include not only outdoor urbanscapes, but also indoor areas, rural areas, forests, deserts, and even under-water seascapes. Images from a spherical array of cameras can recover depth and reconstruct 3D point clouds using structure-from-motion algorithms [11]. Recently, laser scanners are being coupled with the cameras to directly acquire depth with omnidirectional panoramas.

2.2.2 Visual Management of Geotagged Information

As a geographic information system, *Social Street View* is most closely related to *Panoramio*, *Newsstand* [8], and *PhotoStand* [12] (Figure 2.2). These systems accomplish the visual management of geotagged information on 2D maps. *Panoramio* is one of the first systems that collects user-submitted scenery photographs and overlays them on a 2D map. *Newsstand* [8] is a pioneering system that allows users to interactively explore photos directly from news articles depending on the query location and the zoom-level on a 2D map. Its successor, *TwitterStand* [13] is able to visualize tweets on a 2D map by using geotagged information as well as inferring geospatial relevance from the content of the tweets. Recently, *PhotoStand* [12] has shown how to visualize geotagged images from real-time scraped news on a 2D map. A primary distinction between *Social Street View* and the above systems is the use of immersive maps instead of 2D maps. In virtual environments, immersive maps require us to address challenges such as visual clutter, designing the content layout in 3D, and registration of pictorial information.

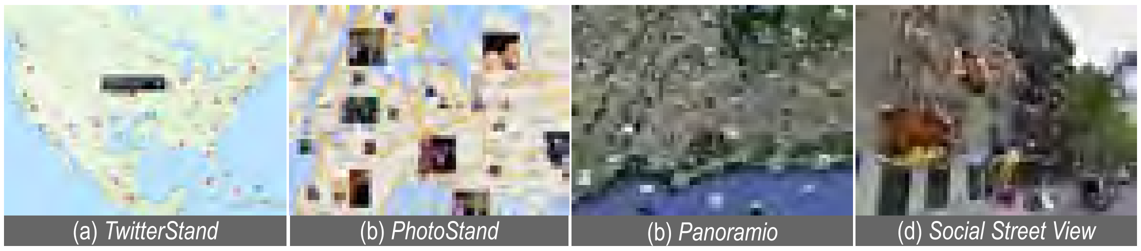


Figure 2.2: Comparison with prior art in visualizing geo-tagged news, social media or photographs. (a) TwitterStand [13] (b) PhotoStand [12] (c) Panoramio (d) Social Street View. All snapshots were taken from their public websites in March 2015.

In 3D environments, the well-known view-management system by Bell *et al.*

[14] registers user-annotated text and images to a particular point in 3D space. Their algorithm reduces visual discontinuities in dynamically labeled and annotated environments. Our system does not involve manual steps to visualize social media in immersive maps. *SSV* also reduces the visual clutter by maximal Poisson-disk sampling or road orientations. Recent efforts in novel social media visualization interfaces include *Social Snapshot* [15], *Photo Tourism* [16] and *3D Wikipedia* [17]. Instead of using immersive maps, *Photo Tourism* uses image-based modeling and rendering for navigating thousands of photographs at a single location. However, since *Photo Tourism* performs 3D scene reconstruction from unstructured photos, it is slow, taking a few hours to process a few hundred photos. The system also discards noisy, dark, cluttered photos due to registration failure. Similarly, *3D Wikipedia* automatically transforms text and photos to an immersive 3D visualization but suffers from a relatively slow bundle adjustment and multi-view reconstruction. In contrast, *Social Street View* takes advantage of the large-scale 2.5D data and can visualize multiple geo-relevant photos in immersive environments at interactive rendering rates.

Creating an immersive visualization of geotagged social media is a challenging task due to the lack of 3D data. For example, reconstructing a 3D mirrored world from images typically requires intensive computation for a few hours or days.Early seminal work [17, 18, 19, 20, 21] focuses on offline, image-based modeling approaches to generate virtual 3D cities. In these systems, 3D models are generated from a large collection of unstructured photos via different structure-from-motion pipelines. Since the debut of *Social Street View* [1], recent research offers more practical solutions to integrate geotagged social media with pre-reconstructed cities in several minutes [22], and virtual city models [23, 24]. However, these approaches are not quite applicable to real-time applications. On the one hand, the texturing [22] of 3D buildings suffers from artifacts on complex geometries. On the other hand, the pre-crafted digital cities used in [22, 24, 25, 25, 26, 27] are usually unavailable in rural areas and require enormous amounts of collaborative work from crowdsourced workers, artists, researchers, and city planners [28, 29, 30, 31]. Moreover, without a partitioning algorithm, the digital cities (over 100 MB as mentioned in [24]) may be a bottleneck for practical online deployment.

In contrast to the prior art, we further develop *Geollery* and circumvent the offline reconstruction or manufacture of a digital city by progressively streaming open 2D maps. With 2D polygons and labels, *Geollery* extrudes and textures geometries on demand in *real-time* using nearby street view data, enabling visualization of geo-relevant social media with their spatial context, and allowing user interaction in a mirrored world. As for human factors in 3D social media platforms, Kukka *et al.*[23] conduct a pioneering qualitative *anticipated* user experience study with 14 participants to explore the design space of geospatial visualization of social media in mirror worlds. Nevertheless, the human factors have not yet been fully discussed for *experiencing* a *real-time* mixed-reality social platform such as *Geollery* or *Social Street View* [1]. We conduct a comparative study with 20 participants and derive key insights from the semi-structured interviews. Our qualitative evaluation further reveals the strengths and weaknesses of *Geollery* and *Social Street View.*

2.2.3 Analysis of Geotagged Social Media

An important question to consider for us is the accuracy of the geotagged media corresponding to the real geographic location. Zielstra and Hochmair [32] experimented to investigate the positional accuracy of 211 image footprints for 6 cities in Europe by comparing the geotagged position of photos to the most likely place that they were taken at, based on image content. In this study, they found Panoramio has a median error distance of 24.5 meters, and Flickr images have a median error distance of 58.5 meters. With the growing popularity of global positioning system (GPS) and Wi-Fi positioning systems on mobile devices, the geospatial metadata information is increasingly reliable and accurate. We have observed that occasionally there may be some irrelevant social media based on location query, but this is not common. Nevertheless, we have implemented a mechanism for users to report the abuse of geotagged social media in our system.

Previous research also explores various ways to analyze and visualize geotagged information on 2D maps. For example, MacEachren *et al.*[33] present a seminal system for visualizing the heat maps of health reports on a map. Their further work, *SensePlace2* [34], presents a geospatial visualization of Twitter messages with user-defined queries, time filters, spatial filters, and heap maps of tweet frequencies. Chae *et al.*[35] present a social media analysis system with message plots on a map, topic filtering, and abnormality estimation charts. Recent research also focuses on gridded heat maps [36], multivariate kernel methods [37], movement patterns [38], Reeb graphs [37], sentiment modeling [39, 40, 41], and flow visualizations of spatio-temporal patterns [42]. Using domain-specific knowledge, previous research has analyzed geotagged social media to improve emergency responses [43, 44], assist disease control [45], understand the dynamics of neighborhoods [46] and cities [47, 48], and travel route planning [49].

The key differentiator of our work is the ability to explore geotagged information in immersive 3D environments. *Geollery*, the successor of *Social Street View*, is able to create a digital city progressively in real time and fuse geotagged social media into the mirrored world. For 2D spatiotemporal analysis, we present *TopicFields* to offer social-media-topic extraction, spectral ordering of related topics, and exploratory visualization of the scalar fields of multiple topics.

2.2.4 Mixed Reality in Immersive Maps

Past work on mixed reality in immersive maps generally required users to manually augment content for immersive maps. Devaux and Paparoditis [50] added laser-scanned depth data to some street views and enabled users to manually add images or videos at their desired 3D position. Their system also had additional interactive features including human labeling, crowdsourcing mode to blur faces, and localizing and measuring objects. In contrast, by automatically extracting proximal social media content, *SSV* dynamically enhances the user experience in immersive maps and allows users to focus on social media interactions. Korah and Tsai [51] convert large collections of LIDAR scans and street-view panoramas into a presentation that extracts semantically meaningful components of the scene via point-cloud segmentation. In addition, they propose an innovative compression algorithm and also show how to augment the scene with physics simulation, environmental lighting, and shadows. Past research on the analysis of immersive maps also addresses the important problems of segmentation, human recognition, and accessibility identification. For instance, Xiao and Quan [52] propose a multi-view semantic segmentation framework using pair-wise Markov Random Fields (MRF) to differentiate ground, buildings, and people in street views. [53] have proposed a novel algorithm to replace any unknown pedestrian with another one which is extracted from a controlled dataset. [54] have designed a visual interface that used a machine learning algorithm to facilitate crowd-sourcing-driven identification of accessibility issues in immersive maps, such as missing curb ramps, for people in wheel chairs.

2.3 System Architecture

In this section, we discuss the overview of the system architecture of *Social Street View* presented in Figure 2.3. *Social Street View* consists of a street view scraper, a social media scraper, distributed SQL databases, a web-server powered by Apache and PHP, and optional modules such as a temporal filter, a geolocation filter, and a computer-vision-based face filter.

2.3.1 Street View Scraper

Our street view scraper is a custom web-scraper tool written in Javascript and PHP that downloads GIS-related panoramic images and metadata at any geoloca-

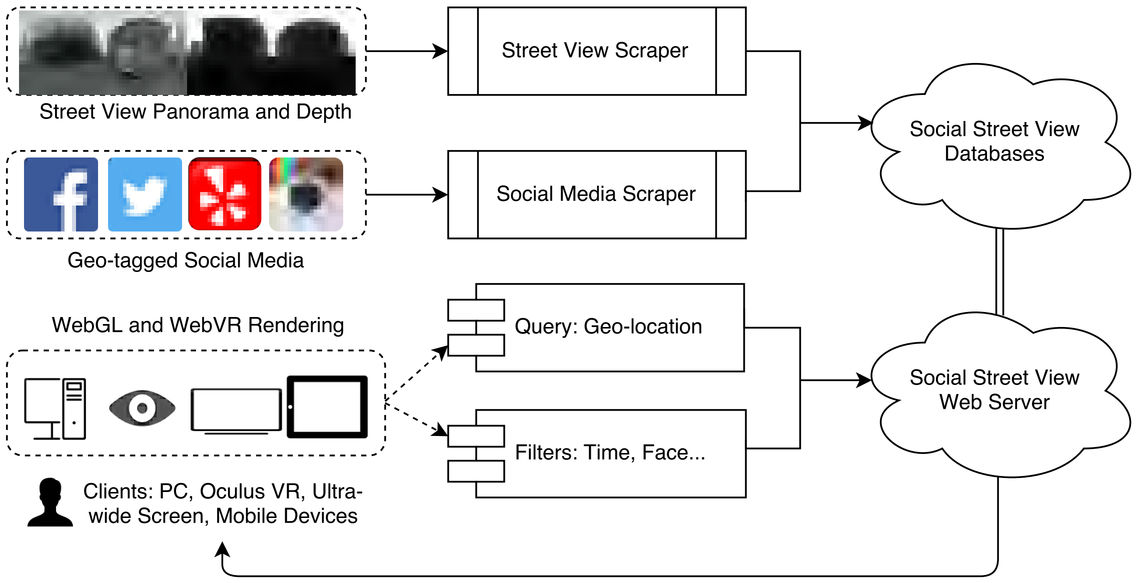

Street View Panorama and Depth
Street View Scraper
Geo-tagged Social Media
Social Media Scraper
Social Street View Databases
WebGL and WebVR Rendering
Query: Geo-location
Filters: Time, Face...
Social Street View Web Server
Clients: PC, Oculus VR, Ultra-wide Screen, Mobile Devices

Figure 2.3: The work flow of *Social Street View*. Our system streams data from two scrapers based on users' geolocation requests and renders social media in WebGL. Users can access the system via any WebGL-supported browser on a desktop, a tablet, a head-mounted display, or an immersive room-sized tiled display (see Figure 2.1).

tion where GSV data is available. Our tool is inspired by `GSVPano.js`[4], but we additionally scrape normal maps, infer building surfaces, and make use of the road orientations. Currently, we request all street view data from Google Maps API. Each location query is analyzed by regular expressions, and thus it can be either a mailing address (*e.g.*, 129 St., New York) or a pair of latitude and longitude coordinates (*e.g.*, 40.2384, -70.2394). For each location query, the scraper finds the closest panorama and downloads the following types of data:

1. **Tiles of panoramic images** with five types of resolution: from highest 13312×6656 pixels to the lowest 832×416 pixels. A stitched panorama is shown in Figure 2.4(a).

[4]GSVPano.js: `https://github.com/heganoo/GSVPano`

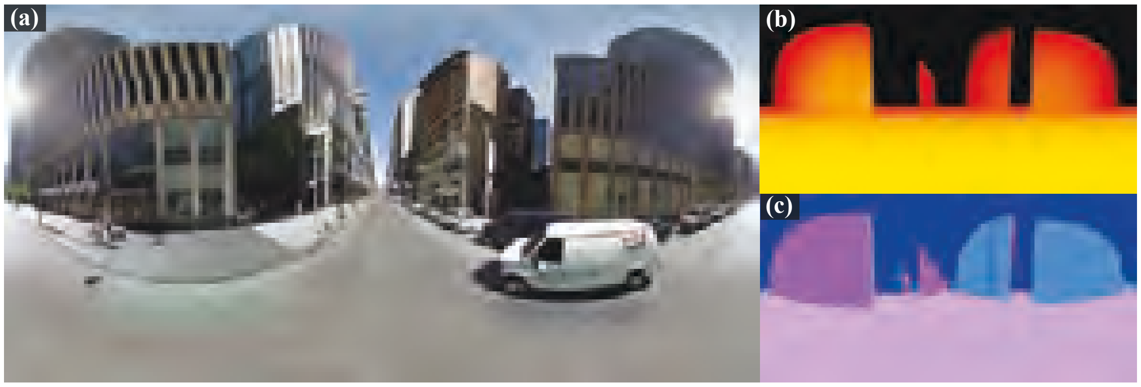


Figure 2.4: The illustration of (a) stitched panoramic image tiles, (b) depth map and (c) normal map from Google Street View. The depth map is visualized in a yellow-red-black color scheme, where black indicates 255 meters or more, red indicates 128 meters and yellow indicates 0 meters. The normal map contains a 3D normal vector for each pixel. We visualize the normal data by converting the normal vector to HSV color space with blue-purple hues.

2. **Depth map** from GSV meta data with a coarser resolution of 512×256 pixels.

   We normalize and up-sample the depth and normal maps in our GLSL shaders (Figure 2.4(b)).

3. **Normal map** of 512×256 pixels (Figure 2.4(c)).

4. **Road orientation and heading direction** indicates the travel direction of the GSV car or trekker.

5. **Geolocation and other information** including latitude, longitude, image age, and adjacent panoramas' hash IDs.

2.3.2 Mining Social Media

Our geotagged social media scraper is a back-end program written in PHP. Tuchinda *et al.* [55] have proposed to model the web services as information sources in a mediator-based architecture and have built an exemplary application, Mashup. Using similar architecture, *Social Street View* is able to integrate information from several web services. For now, we use Instagram as the major source of social media in our proof-of-concept system. Since Instagram only allows users to upload images or videos from mobile devices, it largely avoids incorrect geotagged data from desktop clients. Our social media scraper collects the following data:

1. **Geospatial and textual location** including latitude and longitude coordinates, street names and user-tagged location name.

2. **Media type** indicating whether it is an image or a video.

3. **Caption and tags** containing text information.

4. **Published Time** containing the exact time-stamp when published.

5. **User comments and likes** reflecting the popularity level.

6. **URL** provides a link to the images or videos on the web.

Instagram API supports social-media query based on both geolocation and time. We use two distance thresholds, $\alpha = 10m$ and $\beta = 5km$ for dense urban areas and rural areas, respectively. Given a street view panorama, the scraper first requests social media within a radius of α on Instagram. If nothing is found, the scraper increases the threshold to β and queries again. If either threshold returns social media content, we send out R requests to acquire data over the past R months (we typically use $R = 12$). This allows *Social Street View* to answer spatial queries with a temporal filter, such as "*What do people wear in winter at this location?*".

2.3.3 Servers and Relational Databases

At present, we use distributed MySQL databases to store information of visited immersive maps and social media to reduce response time for duplicated or similar queries. One of the important components of our system is to build spatial data structures to efficiently answer proximity queries relating geotagged social media with spatially-located immersive panoramas. To accomplish this effectively, we build a bipartite graph that establishes edges between social media message nodes on one side with the immersive panorama nodes on the other side. This allows us to quickly answer what social media messages are relevant to be shown in any panorama. Since this can result in an all-pairs quadratic relationship, we needed to do this in an efficient manner. Once the two scrapers complete their tasks, the back-end servers build the bipartite graph in a separate thread: $G = \langle V, S, E\rangle$, where $V = \{v_i\}$ are the visited street views and $S = \{s_i\}$ are scraped social media, and $E = \{\langle v_i, s_j, d_{ij}\rangle \mid v_i \in V, s_j \in S\}$ are the edges between V and S. The weights of edges d_{ij} are defined by the distance between s_i and v_i according to the Haversine formula [56]:

$$\alpha_{ij} = \sin^2\left(\frac{\varphi_i - \varphi_j}{2}\right) + \cos(\varphi_i)\cdot\cos(\varphi_j)\cdot\sin^2\left(\frac{\lambda_i - \lambda_j}{2}\right) \tag{2.1}$$

$$\beta_{ij} = 2\cdot atan2\left(\sqrt{\alpha_{ij}}, \sqrt{(1-\alpha_{ij})}\right) \tag{2.2}$$

$$d_{ij} = R \cdot \beta_{ij} \tag{2.3}$$

where φ_i, λ_i and φ_j, λ_j are the latitude and longitude of s_i and v_j respectively; R is the radius of the earth ($R = 6371km$); the result d_{ij} is the great-circle distance between s_i and v_j. Since both social media and street views are indexed by a $B+$ tree, the insertion and query without building the graphs take $O(\log|V| + L)$ time, where $L = 100$ is the maximum number of queried social media. There could be an additional cost of sorting based on users' query and filters. However, the maintenance of the bipartite graph may take $O(k \cdot |V|)$ for k newly scraped social media S_k. To solve this issue, given a street view v_i, *SSV* calculates $S_i = \{s_j \mid \forall s_j \in S \wedge \langle v_i, s_j, d_{ij} \rangle \in E\}$ in $O(\log|V| + L)$. If $S_i = \emptyset$, *Social Street View* returns S_k and builds the bipartite graph at the back-end for the next query; otherwise, we return $S_k \cup S_i$ for more results. Thus, for each panorama, the streaming time is $O(\log|V| + L\log(L))$.

2.4 Social Street View Interface

Using WebGL and WebVR, we have designed and implemented an open-source and cross-platform system which is easy to access via most modern browsers and is shown in Figure 2.5. Users can query any location in the input field on the top panel. The left optional panel has filters to specify query words, temporal (month and hour) ranges, distance ranges, number of faces and rendering controls. The bottom panel is an optional 2D visualization. We use `Three.js`[5], a cross-browser and GPU-accelerated Javascript library. We use `Bootstrap`[6] and `jQuery`[7] for 2D elements of the user interface.

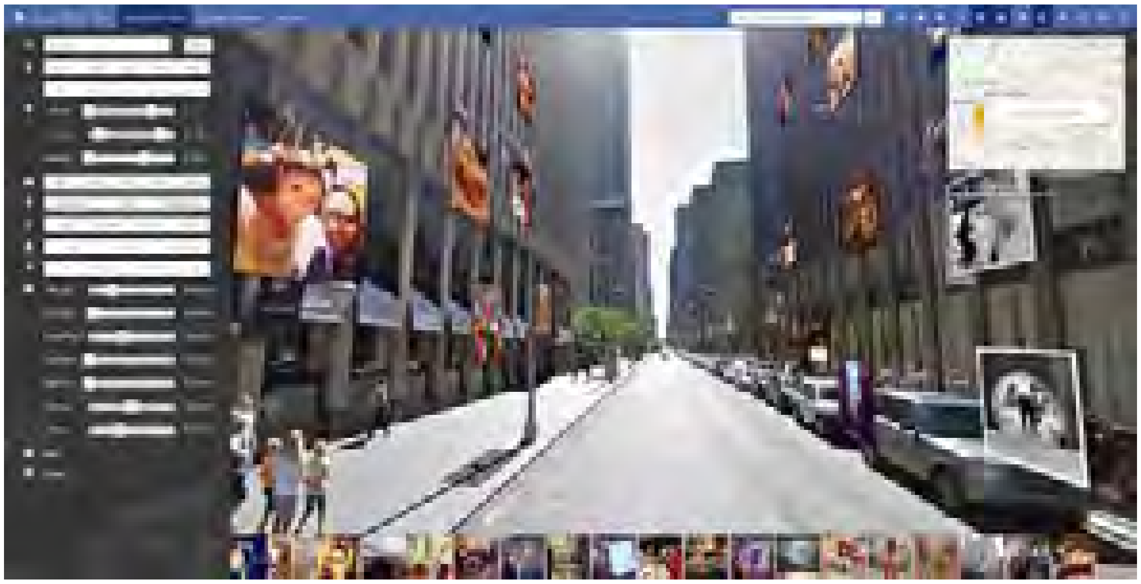


Figure 2.5: The *Social Street View* interface powered by WebGL and WebVR. The top navigation bar allows the user to switch among desktop and VR mode, input addresses, and enabling different postprocessing modes. The left control panel allows the user to search for certain keywords, filter based on the month of the year, the day of the week, and the distance to the current location. The top left legend shows a mini-map indicating where the user is located. The bottom panel is a baseline 2D approach for displaying nearby social media.

2.5 Social Media Layout Algorithm

In this section, we present our approach to blending the visualization of social media with street view panoramas in an immersive mixed reality setting.

[5]Three.js: `http://www.threejs.org`
[6]Bootstrap: `http://www.getbootstrap.com`
[7]jQuery: `https://www.jquery.com`

2.5.1 Baseline: 2D Visualization

Since there is limited previous work on visualizing multiple social media images in immersive maps, we first devised a basic 2D solution for users to have a quick glimpse of social media near their location. This is shown in the bottom panel of Figure 2.5. The users can click on any image to see a higher-resolution (640×640) image or video, text caption message, geolocation, and timestamp data. In addition, the user can use the image to link to Instagram to *comment*, *like*, or *forward* the social media. By clicking the geolocation of the social media, the user can navigate to the closest street view panorama to the image using *Social Street View*. Compared with *Stweet* that places a single Tweet message on a top layer above Google Street View with limited interactivity, this basic 2D visualization provides multiple queries near the center of panorama with richer information. In addition, the users can filter out the social media based on a desired time range and distance to the center of the panorama.

2.5.2 Uniform Random Sampling

From *Social Street View*'s server, the client acquires a subset of images or videos $\hat{S} = \{s_i \mid i = 1 \ldots N\}, \hat{S} \subseteq S$. Our goal is to place them naturally in low-saliency areas in an immersive map so that the social media rendering will minimize the visual clutter in a user's view. To do this, we first stitch the tiles of panoramic images into a rectangle T. Next, we apply a projection $\mathscr{P} : T \to \mathit{\Omega}$ from T to the sphere $\mathit{\Omega}$, thus building an immersive panoramic map with an inside camera looking outwards. For each point $\mathbf{p}_i = (u_i, v_i) \in T$, the corresponding point $\mathbf{q}_i = (x_i, y_i, z_i) \in \mathit{\Omega}$ is projected uniformly on a sphere. The easiest way to place social media is to randomly sample N points $\tilde{P} \subseteq T$ from the panorama. For each $\tilde{p} = (\tilde{u}, \tilde{v}) \in \tilde{P}$, calculate the corresponding 3D position $\tilde{q} = (\tilde{x}, \tilde{y}, \tilde{z}) \in \mathit{\Omega}$ as the center of the social media. As shown in Figure 2.6(a), while we find that we can blend many interesting social media in this interactive mixed-reality world such as photography, food, and people there are several drawbacks in the way they are laid out. To address this, we propose the following desiderata:

1. Ground-level context is important for way-finding for pedestrians and drivers. Therefore a system blending social media with immersive maps should minimize the rendering of the social media at or near the ground level.

2. Since billboards and other structures in the real world are often aligned with physical landmarks, it is desirable to align social media with the proximal geometric structures.

3. Free-floating imagery in mixed reality worlds is likely to result in cognitive dissonance and a psychophysical unease in users.

4. Social media imagery should be reasonably spaced apart to avoid visual clutter and overlaps, if at all possible.

We next discuss how we accomplished the above goals in our work.

2.5.3 Depth and Normal-map-driven Placement of Social Media

A depth map $D = \{d_i\}$ in which each depth value d_i corresponds to a pixel p_i in T, could be used to filter out points that are too close (e.g., the ground) or too far away (*e.g.*, the sky). Additionally, we scale the size of images based on depth to give a perspective effect. The result is shown in Figure 2.6(b). This minimizes the images that are projected on to the ground or the sky.

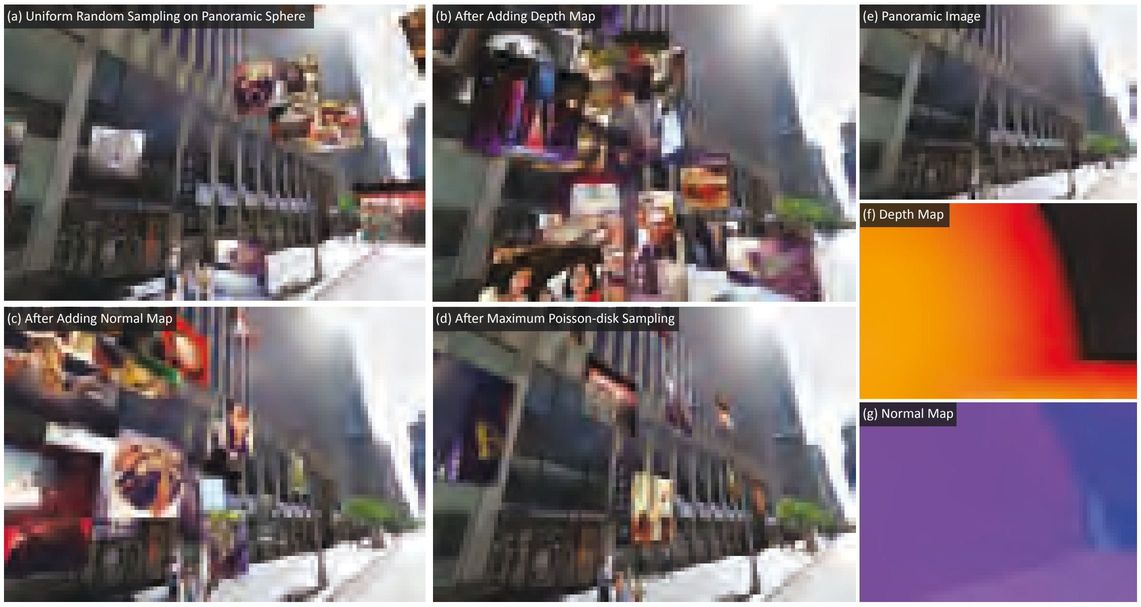

(a) Uniform Random Sampling on Panoramic Sphere
(b) After Adding Depth Map
(e) Panoramic Image
(f) Depth Map
(c) After Adding Normal Map
(d) After Maximum Poisson-disk Sampling
(g) Normal Map

Figure 2.6: Results before and after applying the depth map, the normal map, and maximal Poisson-disk sampling. (a) shows the random layouts generated uniformly on the sphere, (b) shows results after using the depth map, (c) shows results after applying the normal map, (d) show results after using maximal Poisson-disk distribution for laying out photographs, (e) shows the original street view image, (f) and (g) visualize the depth map and the normal map respectively, for reference

We use the normal map $\mathbf{N} = \{\mathbf{n}_i\}$ to project the images onto the surfaces of buildings. Denoting the normal vector of the ground as $\mathbf{n}_g = (0, 1, 0)$, we define the ground level Ω_g as follows:

$$\Omega_g = \{q_i \mid \forall q_i \in \Omega \wedge \|\mathbf{n}_i - \mathbf{n}_g\| < \delta\} \tag{2.4}$$

where $\|\mathbf{n}_i - \mathbf{n}_g\|$ is the Euclidean distance between two vectors $\mathbf{n}_i$ and $\mathbf{n}_g$, $\delta = 0.5$ is a user-defined threshold. Next, for each sampled point $\tilde{\mathbf{p}_i}$, we use the corresponding normal vector $\mathbf{n}_i$ and rotate the social media to the correct orientation. The results are illustrated in Figure 2.6(c). As one can see, the images are now well-aligned with the geometry of the buildings. However, the rendering still suffers from visual clutter and overlaps.

2.5.4 Maximal Poisson-disk Sampling

We use maximal Poisson-disk sampling to solve the problem of visual clutter. Poisson-disk distribution has been widely used in the field of computer graphics for global illumination [57], object placement [58] and stochastic ray tracing [59]. In our work, we follow the approach of the PixelPie algorithm devised by Ip *et al.* [60], which uses vertex and fragment shaders and GPU-based depth-testing features to efficiently implement the dart-throwing algorithm for maximal Poisson-disk sampling.[8]

After sampling points from the building surfaces, we sort them according to their depth. We preferentially place more *popular* social media closer, where popularity is defined in section 4.7. We outline this approach in Algorithm 1.

[8]Code of PixelPie: `http://sourceforge.net/projects/pixelpie/`

ALGORITHM 1: Social Media Layout using Poisson-disk Samples

```
Input: $N$ sorted social media images $\hat{S} = \{s_i \mid i = 1 \dots N\}$, acquired from
        $SSV$ servers.
Output: A set of image planes to display social media: $I = I_1 \dots I_M, M \leq N$.
Generate the set of candidate sample points $\tilde{\mathbf{P}}$ by the PixelPie algorithm;
Sort points in $\tilde{\mathbf{P}}$ in descending order according to their corresponding values
  in the depth map $D$ so that the closest sample point is laid out first;
Set $I \leftarrow \emptyset$;
for $i \leftarrow 1 \dots \min(N, |\tilde{P}|)$ do
    Place $I_i$ with texture from $s_i \in \hat{S}$ at the projected position $\tilde{\mathbf{q}}_\mathbf{i} \leftarrow \mathscr{P}(\tilde{\mathbf{p}}_\mathbf{i})$;
    Rescale $I_i$ according to the corresponding depth value: $\tau_i \leftarrow \tau / d_i$ for
      perspective visual effects;
    Rotate $I_i$ so that it is perpendicular to the normal vector $\mathbf{n_i} \leftarrow \mathbf{N}(u_i, v_i)$;
    Add $I_i$ to the result set: $I \leftarrow I \cup I_i$;
end
```

This provides us with an aesthetic layout to display social media blended in with the immersive panorama. A screenshot of the resulting placement after this algorithm appears in Figure 2.6(d).

2.5.5 Placement of Social Media in Scenic Landscapes

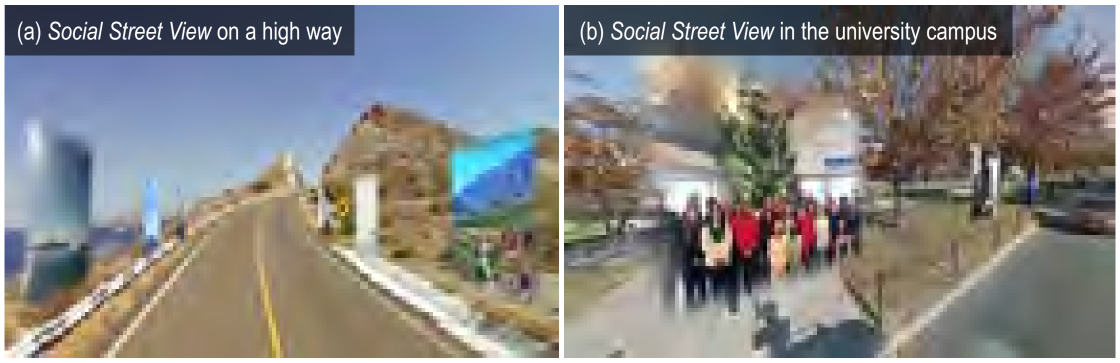

(a) Social Street View on a high way
(b) Social Street View in the university campus

Figure 2.7: Results of laying out social media in scenic landscapes. (a) shows the rendering results on a highway and (b) shows the results in the university campus. Note that both locations lack the information of depth maps, but the system is able to lay out social media along the road.

In open and scenic areas, there are a limited number of surfaces on which to place social media. However, with the high-level knowledge of where the Google Street View camera is traveling from and traveling to, it is possible to place social media along the way without depth or normal maps. To generalize our system from urbanscapes with buildings to more general scenic landscapes, we propose Algorithm 2 and the results are shown in Figure 2.7.

```
ALGORITHM 2: Social Media Layout using Road Orientations
Input: $|O|$ road orientations with $o_i \in [0, 2\pi]$. $K$ social media to be placed for
       each orientation. Typically, $|O| = 2$ for a road with two orientations.
Output: A set of image planes to display social media:
        $I = I_1 \dots I_M, M \geq K \cdot |O|$.
Set $I \leftarrow \emptyset$;
for $i \leftarrow 1 \dots |O|$ do
    Set the position $\mathbf{q_i} \leftarrow (KR\cos o_i,\ h,\ KR\sin o_i)$ at height $h$ and radius $R$;
    (Optional based on user's preference) Add a frontal image plane to $I$ at
     $\mathbf{q_i}$;
    Set the translation $\mathbf{t} \leftarrow (T\cos(o_i + \frac{\pi}{2}), 0, T\sin(o_i + \frac{\pi}{2}))$ with constant $T$;
    for $k \leftarrow 1 \dots K$ do
        Set $\tilde{\mathbf{q}} \leftarrow (kR\cos o_i,\ h,\ kR\sin o_i)$;
        Add a left side image plane to $I$ at position $\mathbf{q}' \leftarrow \tilde{\mathbf{q}} + \mathbf{t}$;
        Add a right side image plane to $I$ at position $\mathbf{q}' \leftarrow \tilde{\mathbf{q}} - \mathbf{t}$;
    end
end
```

This algorithm is enabled whenever the depth map is missing or the number of building surfaces is smaller than 2. With the normal map, we run a flood fill algorithm to cluster the adjacent pixels whose Euclidean differences are less than 0.05 with the neighbors. The flood fill is implemented by breadth-first search and union sets. A building surface is defined as a cluster whose length is greater than 1500 pixels.

2.5.6 Post-processing, Rendering and Interaction

To enhance the visual effects of the social media in an immersive setting, our system allows the users to add shadows, glowing shader effects, and alpha blending to the virtual billboards that depict the social media message. We can also model the difference between daytime and nights by using a blooming shader and an additive layer based on depth and normal. To experience the static street view in different seasons, we have implemented particle systems to render snow, falling leaves, or cherry petals in the scene. Additionally, we implemented a simple ray tracer that enables users to click on social media to read associated text.

2.5.7 Filtering of Social Media

In crowded areas such as the New York City, it is almost impossible to visualize every message in *Social Street View*. One solution is to give preference to the most popular social media. However, quantifying popularity itself is subjective since popularity spans features such as comments, replies, creation time, likes, and the number of times forwarded. We have adopted the following criteria as a substitute for popularity:

$$\frac{\alpha C_i + L_i}{\Delta T_i} \tag{2.5}$$

where, for a given social media s_i, C_i is the number of comments, L_i is the number of likes, and ΔT_i is the age of the social media message. Since comments generally have a higher impact than likes, we scale comments by a user-defined scaling factor α.

To protect potential privacy concerns or to find celebrity figures from the news, we also incorporate the face filter using Face++ API [9]. Users can also filter social media based on time and distance.

2.6 Experiments and Evaluation

We have carried out a number of experiments to evaluate the *Social Street View* system. Here we report some of our results for a variety of Google Street View resolutions and social media.

2.6.1 Dataset Acquisition and Hardware Setup

We scraped 100 Google Street View panoramas from Manhattan in New York City as our main dataset. We found over $84,055$ social media images on Instagram within a query distance of 20 meters in these panoramas. For each query, our system returns 100 closest social media images according to the distance to the panorama by searching in $B+$ trees. The experiments were conducted in Google Chrome (Version 40.0.2214.115 m) with Nvidia Quadro K6000 and Intel Xeon CPU E5-2667 2.90GHz. The rendering resolution we used is 2650×1440 pixels. To reduce the effects of different network latency, we store all the panoramas and social media on the local disk of the workstation. We present the file sizes in Table 2.1 for a variety

[9]Face++: `http://www.faceplusplus.com`

of panoramas.

Table 2.1: Resolutions, tile counts, and file sizes of the GSV panoramic data

Pixels	Resolution	Number of tiles	File size
88.6M	13312×6656	26×13	$\sim 5M$
22.2M	6656×3328	13×7	$\sim 2M$
5.5M	3328×1664	7×4	$\sim 800K$
1.4M	1664×832	4×2	$\sim 300K$
0.3M	832×416	2×1	$\sim 90K$

2.6.2 Evaluation of Initialization and Rendering Time

Interactivity and latency are of great importance for user navigation in the *Social Street View*. Figure 2.8(a) shows the initialization time based on five different resolutions in Table 2.1. The initialization mainly includes the time spent in querying for panorama and social media, as well as loading texture to memory and WebGL initialization. We notice that when the size of the panoramic texture is reduced to 6656×3328, the overall time cost is reduced from approximate two seconds to one. Choosing an appropriate resolution based on a user's display, downloading speed, and GPU power can therefore make a meaningful difference. We noticed that at least 900ms was being spent on initializing panoramas and social media. To reduce the time when switching between adjacent street views, we pre-fetch the data in memory for faster initialization. Google Street View uses progressive refinement to address latency by initially loading low-resolution images that are refined to higher-resolution ones over time. We can also rely on such an approach when the local disk is unable to support pre-fetching. In Figure 2.8(b), we show how querying time is reduced by pre-fetching. Further, maximal Poisson-disk samples to place 100 social media can be generated at interactive rates. However, the texture loading can still take hundreds of milliseconds and we hope that it could be improved with further advances in the WebGL technology. In Figure 2.8(c), we report the rendering time with and without social media. From the chart, it can be seen that the system runs at around 60 frames per second (fps) for all resolutions. Thus, the rendering of the social media does not affect the experience of navigating *Social Street View*.

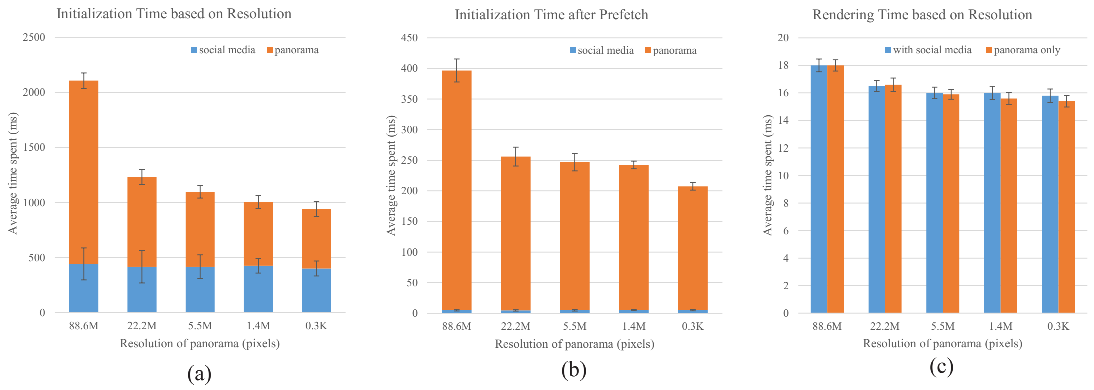


Figure 2.8: Evaluation of processing time in different resolutions using 100 panoramas in Manhattan, (a) shows the initialization time decreases as the resolution goes down, (b) shows that by pre-fetching, initialization time is reduced by over 3 times; (c) shows after initialization, the rendering time costs about 16ms (about 60 FPS) in WebGL while the rendering of social media does not affect the rendering performance much.

2.6.3 Evaluation of Saliency Coverage

Saliency maps can represent regions where a user is likely to allocate visual attention in a fixed-time free-viewing scenario [61, 62, 63, 64]. We compute image saliency using the Matlab tool by Hou *et al.* [65] to evaluate the social media coverage of saliency maps. An example of such a saliency map is shown in Figure

2.9 (a).

The average social media coverage of saliency maps over all the 100 immersive Google Street View panoramas is illustrated in Figure 2.9(b). Initially, the saliency map is covered by uniform random sampling algorithm at about 35%; after incorporating the depth map, most sky areas are filtered out but social media are highly likely to cover the vanishing point where saliency is high; after incorporating depth and normal maps, most social media are aligned with the building structures where saliency is low in most cases; with maximal Poisson-disk sampling, the social media distributes evenly and aesthetically, thus reducing the likelihood that several images could overlap in a high-saliency area. This is the reason why the maximal Poisson-disk sampling has a significantly lower standard deviation in saliency coverage in Figure 2.9 (b). In contrast, the uniform and random sampling, as well as approaches that rely only on depth-map-based placement, result in a larger coverage of higher saliency regions.

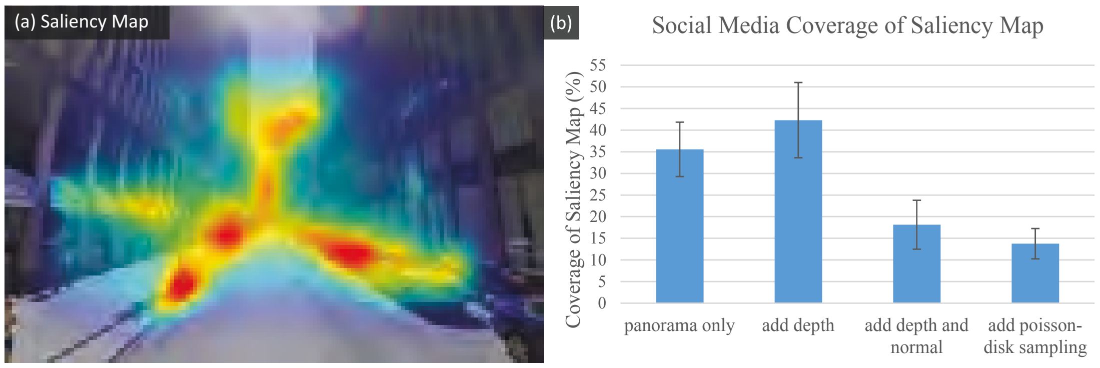


Figure 2.9: (a) Street view with saliency map overlay. The visualization has a red-yellow-green-transparent scheme, where red indicates high saliency and transparent indicates low-saliency. (b) Evaluation results of saliency coverage from 100 immersive GSV panoramas.

2.7 Use Cases and Discussion

While exploring *Social Street View* in a variety of scenarios using publicly available social media, we discovered a number of potential use cases that are promising in enhancing storytelling, business advertising, learning cultures and languages, and in visual analytics of social media in a spatiotemporal context.

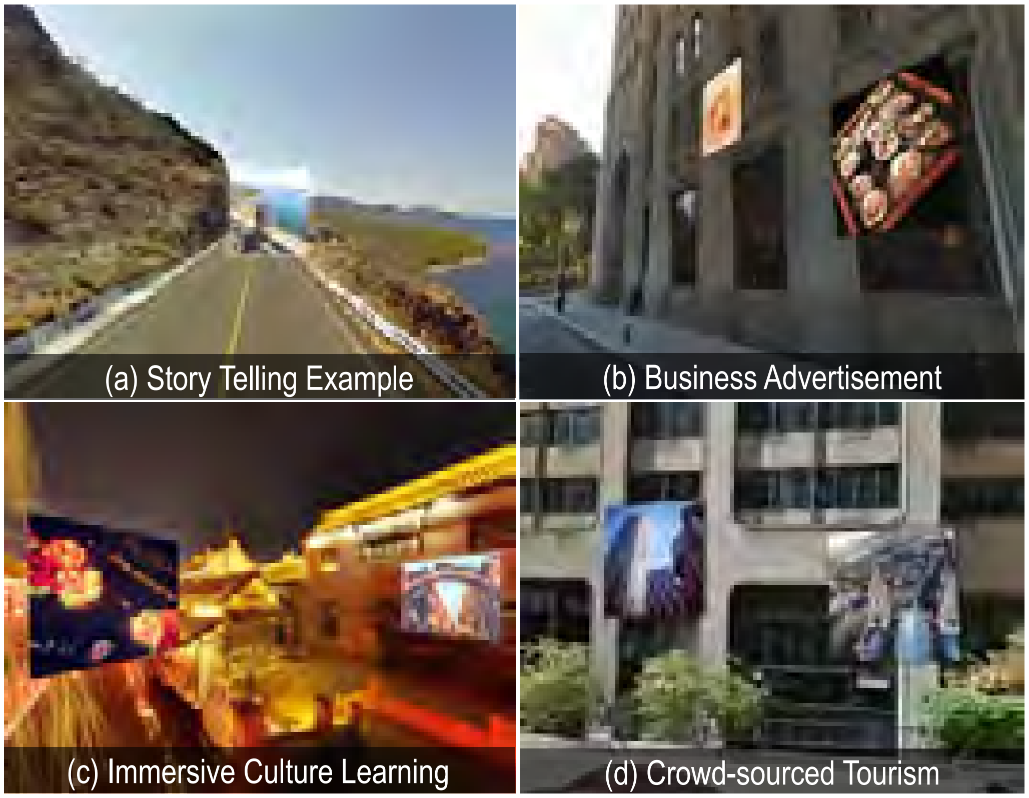

(a) Story Telling Example
(b) Business Advertisement
(c) Immersive Culture Learning
(d) Crowd-sourced Tourism

Figure 2.10: Potential applications of *Social Street View*: (a) Users can link to *Social Street View* to tell immersive stories (b) Business owners can use *Social Street View* for impressive advertising (c) Children can learn culture from local social media (d) Tourists can preview the trip from crowd-sourced photographs embedded in the immersive maps.

2.7.1 Storytelling

Social Street View could greatly enhance the storytelling experience. For example, users could see photos from recent trips of their friends while allowing them to explore the 360° context in an immersive setting.

In Figure 2.10(a), we present how social media stories can be more convincing using *Social Street View*. This panorama is along a road in Baja California Sur within Mexico[10]. Since this open road does not have vertical proximal structures, we use the scenic landscape layout mode here. Further, because it is along a long road we did not expect anyone to take photos and upload them to the social media. Nevertheless, our system found 3 images within a radius of 20 meters. In one of the images, Instagram user Daniela wrote on *July* 12, 2014:

> *Stuck in traffic on our way to Cabo with this awesome view #roadtrip #cabo #view #mexico*

When we pan and walk around this location, we are also impressed by *this awesome view*. We like to think of this as our system facilitating a democratized, crowd-sourced version of the *Kodak Picture Spot*.

2.7.2 Business Advertising

Social Street View can also be used for business advertising. For example, restaurant managers could showcase the social media photographs of their dishes shared by their customers in the context of the interior ambiance of their restaurants.

[10]geolocation: North 25.855319593, West 111.333931591

Similarly, real-estate customers could view the neighborhood street view augmented by the dynamism of the social media of that community to get a better feel for their prospective neighbors.

Figure 2.10(b) uses a panorama in *6 E 24th St, New York, United States.* In the rightmost image of dishes, an Instagram user *frankiextah* commented:

> *... dinner started off with amazing oysters paired with my favorite Ruinart blanc de blancs champagne*

With mixed-reality rendering, *Social Street View* enhances future consumers' visual memories and makes it easier for them to seek *"amazing oysters"* around this place.

2.7.3 Learning Culture and Crowd-sourced Tourism

Immersive virtual environments have been used to protect the world's cultural heritage and serve as a useful medium for cultural education [66]. However, it is usually challenging to generate relevant captions and up-to-date photographs for each scene of a virtual environment. By blending crowd-sourced social media content with panoramic imagery, *Social Street View* can (with age-appropriate filters and curation) serve as an educational tool for children and researchers to learn cultures and languages in different cities and countries. As shown in Figure 2.10(c), users can experience the holiday atmosphere of the Spring Festivals in Taierzhuang Ancient Town of China, where the oldest "living ancient canal" was built in the Ming Dynasty and the Qing Dynasty. Here again, because of a lack of sufficient vertical structures, one can enable the scenic-landscape mode and visualize recent photographs of the architecture taken by tourists in the daytime and at night.

Figure 2.10 (d) presents an example of crowd-sourced tourism in urban areas. Using face and popularity filters, users can get rid of most pictures with human faces and blend some high-quality photographs with a New York street. These photographs provide novel views for the user's exploring experience.

2.8 TopicFields: Spatiotemporal Visualization of Geotagged Social Media with Hybrid Topic Models and Scalar Fields

Social Street View presents an immersive visualization of geotagged social media at the street level. How can we achieve an overview of the geotagged social media for the entire city? What topics are dominant in different times of day? With recent advances in the deep learning models applied to natural language processing and computer vision, machines are now able to extract hundreds of topics or categories out of the social media data. Nevertheless, there are several problems with directly presenting the topics to the end user:

1. **Topic duplication**. Some, but not all, of the extracted topics could be closely related to each other. In addition, neural networks trained for text and images usually use different classes or labels in the results. From visualization's perspective, we would like to aggregate similar topics from hybrid models. For example, *artists*, *painting*, *art* all refer to *art*. Worse still, machine learning algorithms may fail to classify some features to a specific topic, which requires additional input from the users.

2. **Information overload**. Low-level features such as the unigrams or the image classification labels usually result in hundreds of labels. The variety of results are usually too overwhelming for the user. To deal with this problem, we use spectral clustering and provide the cluster with top frequencies for the user to reduce the scope of their visual search.

3. **Diversity**. Previous visual analytic approaches have investigated heat map visualization of a single topic on a map [37, 67], or heat maps of positive or negative emotions on a map [39]. The challenge of effectively visualizing a large number of topics that are implicit in a diverse spatiotemporal social media has thus far not been adequately addressed.

In this section, we present *Topic Fields* (Figure 2.11), an interactive visualization system of geotagged social media to address these three critical problems. Our goal is to understand and correlate the social topics that occur in the real world at various geographical locations over time.

The main contributions of this work are:

- a novel web-based framework for analyzing, aggregating, and visualizing multiple topics from large-scale geotagged social media data,
- clustering hybrid machine learning classification results with spectral ordering algorithm, such as an interactive matrix diagram,
- an efficient and interactive GPU-driven visualization algorithm for visualizing multi-variate scalar data with kernel density estimation and non-linear normalization methods.

2.8.1 System Overview

As shown in Figure 2.12, our system consists of a social-media query engine, distributed SQL databases, a machine-learning server that runs hybrid modules, and a client-side visualization system.

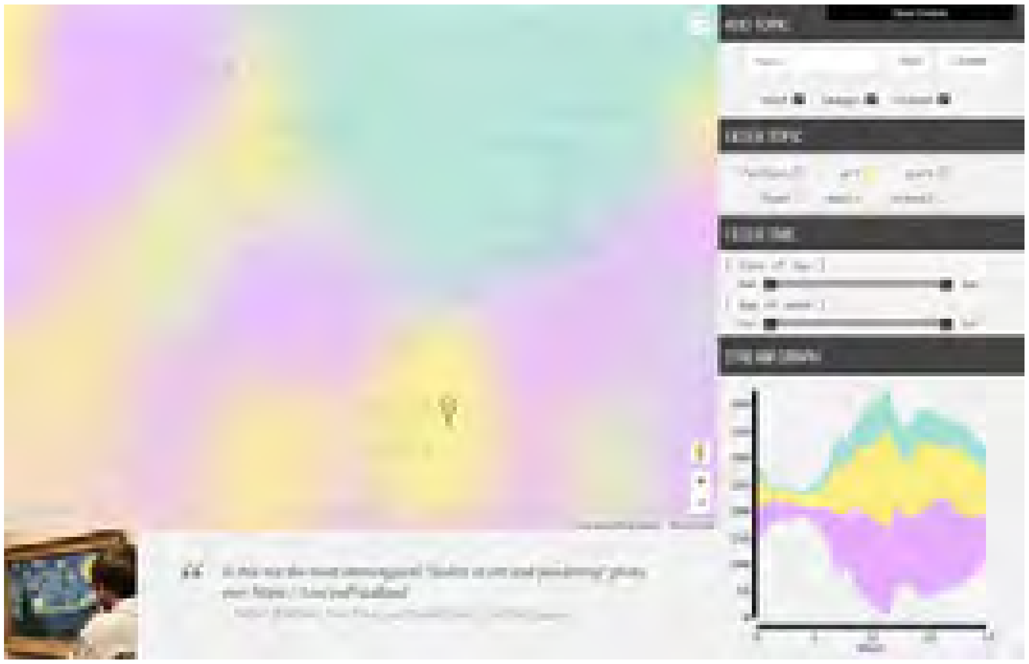


Figure 2.11: A screenshot of the TopicFields system for visualizing approximately one million geotagged social media with hybrid topic models and scalar fields (this figure is best visualized on a computer screen and does not reproduce as well on a printout). The top-left map view is overlaid with an interactive scalar map, with each color indicating a different cluster of topics: fashion, art, and park. We show the ground truth labels such as *Central Park* and *The Museum of Modern Art* from *Google Maps* for reference. The detail view at the bottom shows the corresponding social media text, image, or video, as the user explores and clicks on the map. The control panel on the right allows the user to select, add, and modify the topics to explore. The user could use the "Cluster" button to open the topic matrix diagram (Figure 2.14) and adjust the clustering results. The stream graph shows the volume of social media of different topics over the queried time. The corresponding topic matrix diagram is shown in Figure 2.15.

First, the user is provided with an interactive map. The user can pan around, zoom in or out to select the region of interest. The boundary of the map is then sent to the social media query engine. In the current prototype, the query engine is able to scrape a few hundred geotagged social media from external sources such as *Twitter* and *Instagram* in around a second, but will mostly query from the offline databases. The social media are stored and organized in distributed SQL databases. Next,

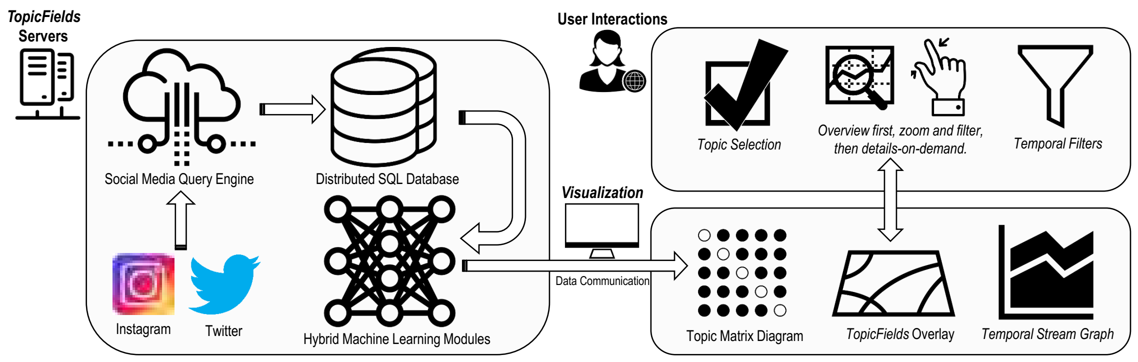


Figure 2.12: The architecture and workflow of the *TopicFields* system. The servers consist of social media query engine, distributed SQL databases, and hybrid machine learning modules for topic classification and clustering from text and images. After the aggregation of topics, the server transfers the data to the client visualization system to present the topic matrix diagram, topic fields overlay, and temporal stream graphs. The users can select the desired topics for visualization, have an overview of the distribution, zoom in and filter by keywords and time, and then visualize the details on demand.

the machine learning module extracts topic features from the text and images, as described in Section 2.8.2.2. It uses the spectral ordering algorithm on the groups of topics and sends the matrix to the user for filtering. On the client side, the majority groups of the topics after spectral ordering are visualized as a matrix diagram. The user can select, add, and remove topic features from the groups and interactively visualize the topic fields on the map.

For exploratory visualization, we have followed the Shneiderman design principle [68] *overview first, zoom and filter, then details-on-demand.* First, we present the map view with topic fields visualization to offer the user an overview of the social media according to the topics. The stream graph offers the user an overview of the volume of the topics over time. Finally, the user is able to zoom in and click on the map view to query proximate social media. We have linked our system with Google Street View to provide an immersive experience to explore the map.

2.8.2 Data Processing

In this section, we present the process of data mining, feature extraction, and spectral clustering.

2.8.2.1 Data Mining

Our geotagged social media scraper is a back-end program written in PHP. Tuchinda *et al.*[55] proposed to model web services as information sources in a mediator-based architecture and have built an exemplary application, *Mashup*. Using a similar architecture, our system is able to integrate information from several web services. In this prototype, we use Twitter and Instagram as the major sources of social media in our proof-of-concept system. We collected the following four types of data:

1. **Geo-spatial and textual location** including latitude and longitude coordinates, street names, and user-tagged location name.
2. **Caption and tags** containing text information of the Tweets or Instagram messages.
3. **Publication time-stamp** containing the exact date and time when published.
4. **User comments and likes** reflecting the popularity level.

We have investigated two major districts on the eastern coast of the United States: *the Manhattan District of New York*, and *the District of Columbia* (Washington D.C.). Over three months from *December* 2017 to *March* in 2018, we have collected 946,856 *Twitter* and *Instagram* messages with specific geographical labels and publication time-stamps from the public domain, with 589,902 in the *Manhattan District* and 356,954 in the *District of Columbia*. The data was scraped using a flood-fill algorithm inspired by Shen *et al.*[69].

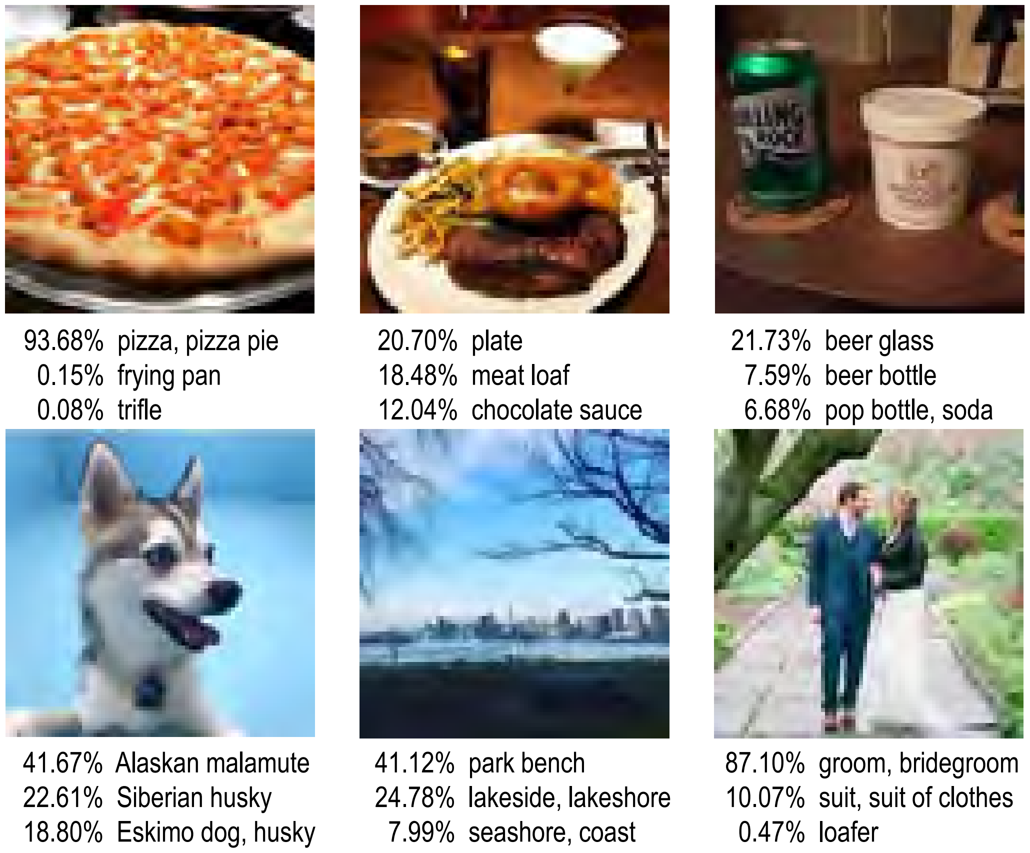


Figure 2.13: The top three classification results by applying the *Inception-v3* deep neural network to our dataset. Our topic feature vector of images consists of the entire hierarchical tree of the top three labels from the *WordNet*. For example, pizza belongs to dish, nutriment, and food.

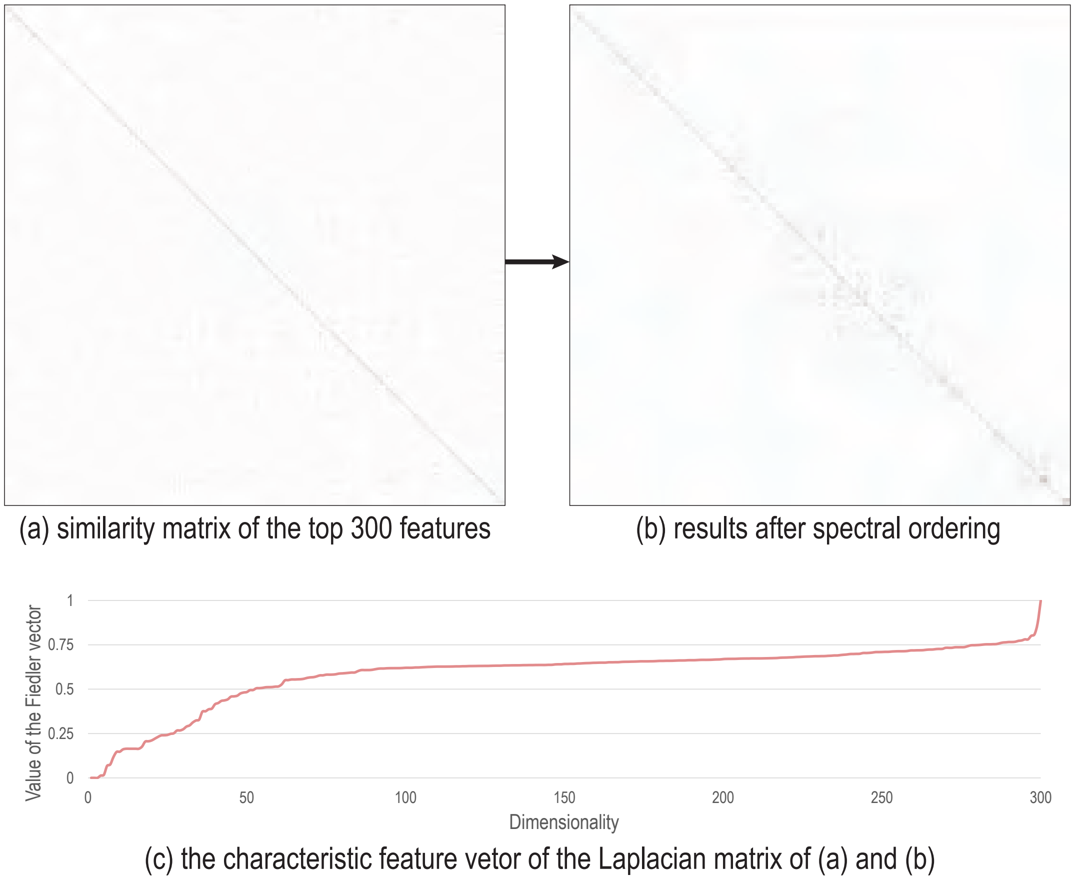


Figure 2.14: Spectral ordering of the topic features; (a) shows the similarity matrix of the top 300 features according to the frequencies, while (b) shows the results after spectral ordering (best viewed on a computer screen). Note that spectral ordering eliminates the randomness of the data and clustering similar groups together. (c) shows the Fiedler vector of the normalized Laplacian matrix of (a) and (b).

2.8.2.2 Feature Extraction

Our data consists of two types of social media: text and images. Sometimes, we obtain videos via the social media query engine. Instead of using video data, we use the first frame or the user-defined thumbnail for feature extraction.

As for text messages, we first experimented with the well-known topic model, Latent Dirichlet Allocation (LDA) [70], to extract accurate clusters of topics from our data. However, the state-of-the-art topic models cannot guarantee clean results.

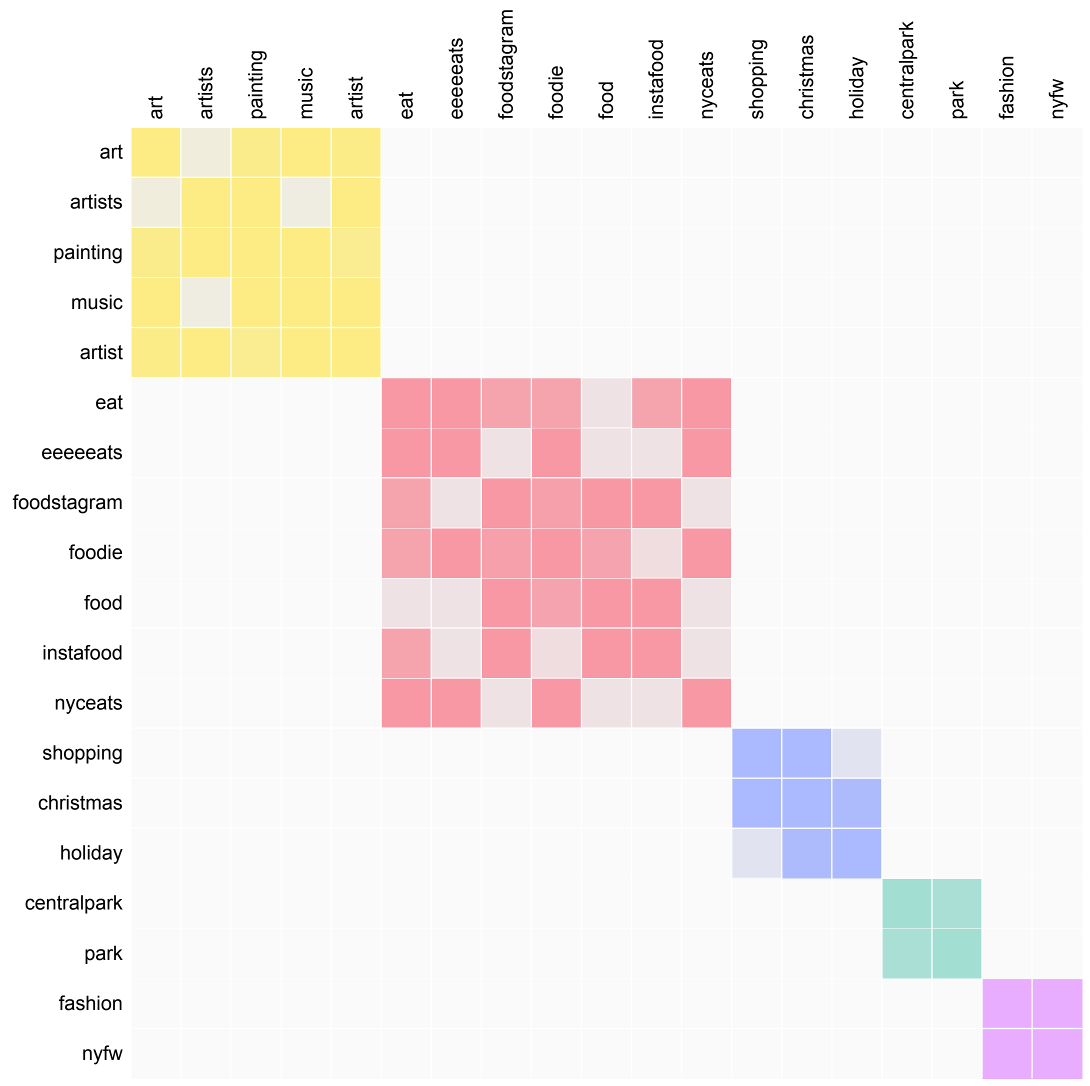


Figure 2.15: The resulting matrix diagram after spectral clustering with the following queries: *art*, *food*, *shopping*, *park* and *fashion*.

For instance, the top three topics we have extracted using LDA [70] are:

enhance, ishootfilm, bend, contemporary, dance, woo, retrospective, ...

sadly, indore, holidayshopping, foodbaby, prk, ...

minidachshund, ana, bulking, busstopdinernyc, friday's, ...

These topics can hardly be used directly for visualization. This led us to the question: can we use machine learning to cluster the most frequent keywords, provide visualization results, and allow the visual analyst to select the desired topics she wants?

Towards this direction, we use the Natural Language Toolkit (NLTK) to extract to top 300 words from the entire social media datasets and apply the $Word2Vec$ neural network to the topics to compute the feature vector for each text data.

Still, 300 features are too much for a visual analyst to interactively explore, so we decided to compute the similarity between each pair of features and use spectral clustering (Section 2.8.2.3) with image labels to help further aggregate the topics into topic classes.

For images, we use the *Inception-v3* model on the image dataset to compute the top three classification results. The results above the 80th quantile are used for extracting the feature vectors. We concatenate all labels from the hierarchical tree to find the topics for the image social media. For example, as shown in the first image of Figure 2.13, the feature vectors are “pizza, pizza pie / dish / nutriment, nourishment, nutrition, sustenance, aliment, alimentation, victuals / food, nutrient”.

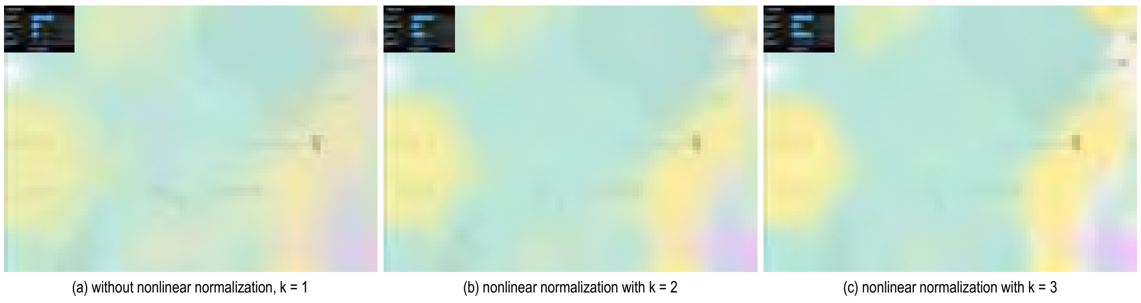

(a) without nonlinear normalization, k = 1
(b) nonlinear normalization with k = 2
(c) nonlinear normalization with k = 3

Figure 2.16: Visualization of the topic fields with different gain factors. (a) shows the baseline visualization using the Gaussian PDF without nonlinear normalization, (b) shows the nonlinear normalization result with a gain factor of 2, and (c) shows the result with a gain factor of 3. The nonlinear normalization significantly increases the contrast between different clusters of topics.

2.8.2.3 Spectral Clustering

Our topic features consist of unigrams and the image classification labels, which can be 300-dimensional vectors. Some, but not all, of the extracted features could be closely related to each other. When features are in arbitrary order along the x-axis of the design widget (Figure 2.14(a)), assigning a meaningful characteristic feature vector may require numerous control points, explicitly defining the value for each dimension. Although the reordering of features does not add to the possible visualizations that can be generated using the machine-learning-assisted approach, the usability issue must be addressed to benefit from the power of high-dimensional representations. We address the relationships among features by rearranging them using spectral ordering, which sorts the features by the eigenvector of the second smallest eigenvalue of a graph Laplacian. First, the normalized Laplacian matrix is generated based on feature-to-feature similarity; then, the eigenvector associated with the second smallest non-negative eigenvalue (the Fiedler vector) is calculated, as shown in Figure 2.14(c); finally, the features are sorted based on their values in the Fiedler vector. The result is shown in Figure 2.14(b) is an ordering of features where neighboring features are similar. The Fiedler vector and other eigenvectors associated with small eigenvalues also form the basis of spectral clustering. Many pairs of the 300 features are indeed highly correlated as can be seen by several dark pixels. Nevertheless, an arbitrary order of features does not take advantage of such correlations, resulting in a disorganized similarity matrix in Figure 2.14(a).

After rearranging the topic features, we cluster the adjacent features using disjoint-set data structures and partition refinement algorithms [71]. The pairs of features whose similarity is greater than $\sigma = 0.7$ and a distance of the spectral ordering is smaller than $\delta = 0.5$ are clustered into one disjoint set. We allow the users to change σ and δ in the control panel for the matrix diagram.

The spectrally-ordered similarity matrix places similar features closer together, resulting in large colorful blocks of various sizes along the diagonal. Thus an accessible feature order allows user-directed selection of similar topics using fewer operations in the control panel.

2.8.3 Topic Fields Visualization

Our algorithm visualizes the scalar field of user-filtered topics of geotagged social media over a map.

Given N geotagged social media over the map, with locations $g_i, i = 1, 2, \cdots, N; g \in G$, suppose each social media is assigned to a set of M topics $T : \{t_1, t_2, \cdots, t_M\}$. Each topic consists of multiple unigrams. We classify a social media to belong to a topic if and only if the unigram appears in the caption, tag, or the hierarchical tree of the image classification results. We limit $M \leq 6$ in our system, since the capacity of the short-term memory for processing information is usually seven, plus or minus two [72], so is the number of colors distinguishable in visualization schemes [73].

First, we generate a grid mesh with $W \times H$ vertices and assign a scalar vector f to each vertex. For vertex centered at g_v, we apply the kernel density estimation within its circle of radius R:

$$f_t(g_v) = \frac{1}{NR}\sum_{i=1}^{N}\mathbf{K}\left(\frac{D(g_v, g_i)}{R}\right) \tag{2.6}$$

, where the kernel function K could be any non-negative function that integrates to one. However, we prefer the kernel functions that smoothly model the falloff of the spatial distribution, such as *Gaussian*, *Quatic*, *Epanechnikov*, or *Triweight* functions. Here we use the *Gaussian Probability Density Function* (PDF) with a bandwidth of R:

$$\mathbf{K}(R) = \frac{1}{\sqrt{2\pi}}e^{-R^2/2} \tag{2.7}$$

Suppose we have a transfer function to colorize each topic t with the color c_t. For each vertex, we can blend the topic fields over the grids by:

$$c = \sum_{t\in T} c_t \cdot \mathscr{N}(f_t) \tag{2.8}$$

, where $\mathscr{N}(\cdot)$ is a nonlinear normalization method to the scalar fields to emphasize centralized topics:

$$\mathscr{N}(f_t) = \frac{g(f_t)}{\sum_{t\in T}\mathscr{N}(g(f_t, k))} \tag{2.9}$$

This nonlinear normalization operator partitions the map into different clusters consisting of different topics. In particular, we apply the gain function $g(x, k)$

employed in modern ray tracing frameworks such as *Pixar Renderman* [74]:

$$g(x,k) = \begin{cases} \frac{1}{2} \cdot (2x)^k, & x < 0.5, \\ 1 - \frac{1}{2} \cdot (2-2x)^k, & x \geq 0.5 \end{cases} \tag{2.10}$$

, where we call k as the gain factor to adjust the contrast of the scalar field. We plot the function in Figure 2.17.

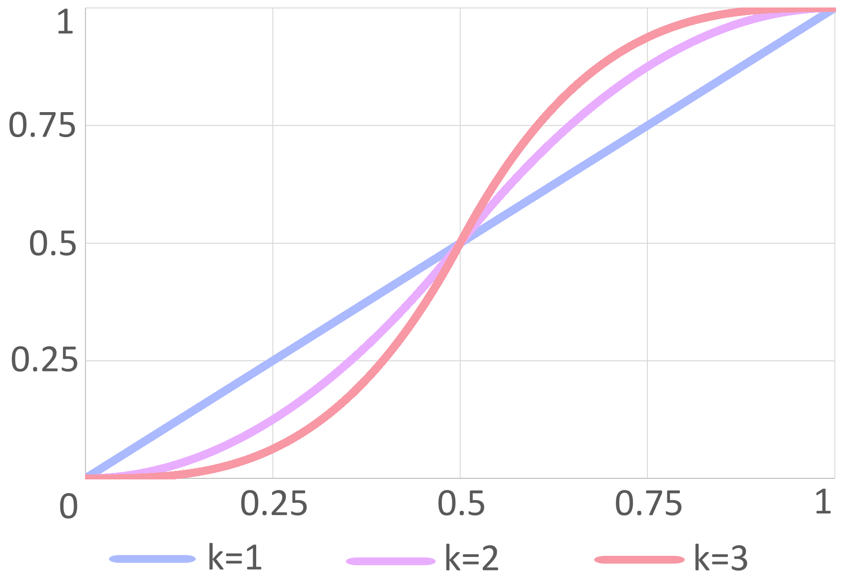


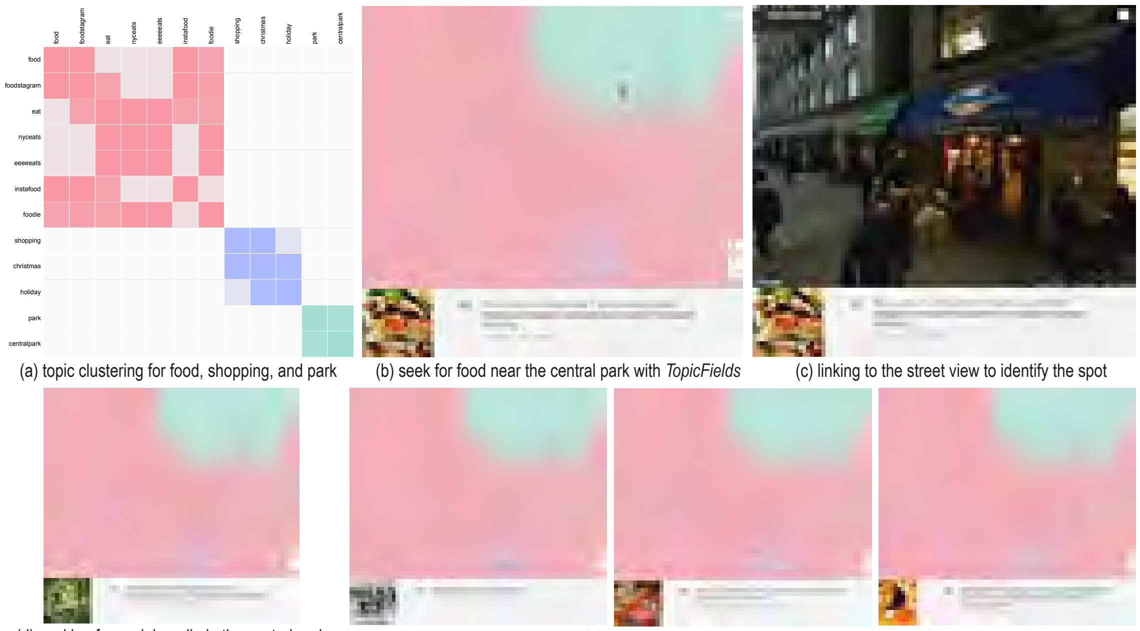
Figure 2.17: The gain function remaps the unit interval into the unit interval. It maps 0.5 to 0.5 while expanding expanding the sides and compressing the center.

By default, we take $k = 2.0$. However, we allow the user to change the gain factor using a control panel powered by *dat.gui* for the WebGL rendering. In this way, we assign a scalar vector to every vertex on the planar mesh. Typically, we use 32×32 vertices for the current boundary on the map.

In the fragment shader, we interpolate the normalized scalar field using Lagrange Bicubic sampling [75] and colorize the scalar field using the user-defined colors. Finally, we efficiently render the Topic Fields using WebGL in a modern browser in real time. We show the visualization results with different gain factors in Figure 2.16.

2.8.4 Use Cases

With *TopicFields* system, we demonstrate two potential use cases for trip planning and searching with temporal filters.



(a) topic clustering for food, shopping, and park (b) seek for food near the central park with *TopicFields* (c) linking to the street view to identify the spot

(d) seeking for social media in the central park (e) finally, looking for places for shopping nearby

Figure 2.18: These figures show the procedure for planning a trip via *TopicFields* near the central park.

2.8.4.1 Trip Planning

First, we demonstrate how *TopicFields* could help a user to plan a short trip near the central park region. Suppose that the user has decided to explore the central park, but has no idea where to go for food and shopping. So the user inputs *park*, *food*, and *shopping* into the topic query box. With the spectral ordering algorithm, the user quickly aggregates three clusters including 12 features into the query engine. With the *TopicFields* visualization, the user could quickly identify geotagged social media that relates to parks, as shown in Figure 2.18(d):

> *#centralparksouth #centralpark #nature #flowers #flowersofinstagram #ferns #flowersofcentralpark #spring #nyc*

From the topic fields, food is distributed everywhere in the map. The user could simply select food, and query close to the central park. One of the results is shown in Figure 2.18(b):

> *This is what I am talking about! #food #newyork #follow #Instagram #foodporn #sandwich #love #Angelas #centralpark #Saturday*

The user could further drag the street view Pacman on the right-bottom of the map to identify the environment near the spot: it seems like a real sandwich shop.

However, the places for shopping seems to be a little far away from the park, given the small purple distribution on the map. Without the *TopicFields* visualization, one may click anywhere on the map to seek for social media that mentions *shopping*, while the *TopicFields* visualization quickly identifies the *5th Ave* as the concentrated places for shoppers:

> *essentials? #louisvuitton #gucci #guccigang #dolcegabbana #guccicommunity #essentials #shopping #manhattan #nyc #5thavenue #streetstyle*

2.8.4.2 Searching with Temporal Filters

We briefly demonstrate how *TopicFields* could help a user to seek for geo-tagged social media with temporal filters. Using the stream graph, we can see that the amount of social media has peak values around 10 am and 5 pm. To see the differences regarding the lake in the central park, we adjust the temporal filter to *before* 5 pm and *after* 5 pm. The results are shown in Figure 2.19.

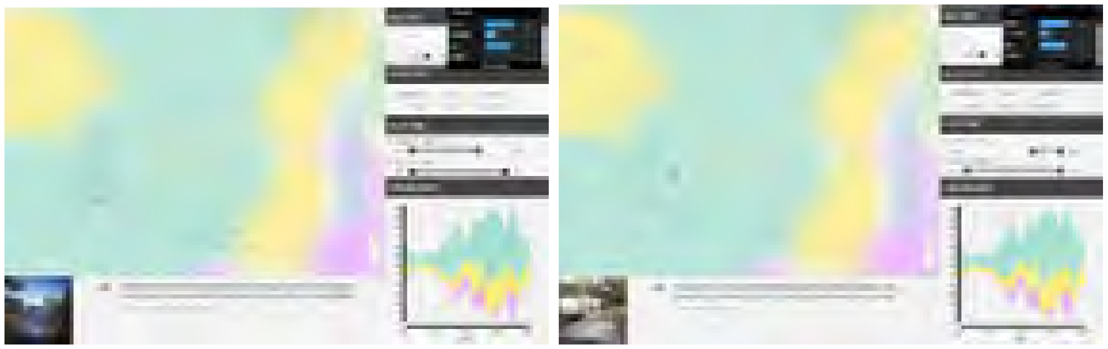


Figure 2.19: The left figure shows the geospatio query with the "park" topic before 5pm, while the right figure shows the geospatio query with the "park" topic after 5pm.

Before 5pm, the pond looks clear and beautiful:

Cooler days as summer turns to fall (and wishing I lived in a four seasons kind of place). #summertofall #centralpark #newyork #nyc #happyfriday

#wagnerscove #wcp #westcentralpark #spring #nature #ponds #nyc beautiful day! #faeries live here ? #nycmydna #timeoutnewyorkcity

After 5pm, the pond has a different atmosphere:

#wagnerscove #wcp #westcentralpark #spring #nature #ponds #nyc beautiful day! #faeries live here ? #nycmydna #timeoutnewyorkcity

2.8.5 Discussion

There are a few limitations and improvements that could be made to our algorithm. First, the variety and diversity of Twitter posts were very surprising. The Tweets were written in various manners, with many tweets containing few *real* words, or no real words at all. In addition, the amount and degree of sarcasm and double meaning in tweets also made calculating the actual topic of the tweets very difficult. For example, there are a few social media messages that used the word *park*, to refer to Park Avenue in New York City.

Second, the deep-learning model on images is not always reliable. Sometimes it provides completely wrong information. For example, in a picture of a dog, the neural network recognized it as a boxer.

Third, the spectral ordering algorithm heavily relies on the neural network that learns the similarity between pairs of word vectors. If the similarity score is not high enough, the algorithm may not place similar words into a cluster.

2.9 Geollery: Designing an Interactive Mirrored World with Geotagged Social Media

2.9.1 Introduction

Social media plays a significant role in many people's daily lives covering a wide range of topics such as reviews of restaurants, updates from friends, local news, and sporting events. Despite the huge innovation in virtual and augmented reality, existing social media platforms typically use a linear narrative or a grid layout. Recently, several technologies and designs [1, 22, 23] (Figure 2.21) have emerged for visualizing social media in *mirrored worlds*[11][77]. Nevertheless, designing an interactive social platform with immersive geographical environments remains a challenge due to the *real-time* constraints of rendering 3D buildings. In addition, the design space of visualizing and interacting with social media in mixed reality settings is not yet fully explored.

As introduced previously, *Social Street View* [1] contributes the initial efforts in blending immersive street views with geotagged social media. Nevertheless, user interaction is limited to street-level panoramas thereby limiting the system to areas covered by street views. Consequently, users could not virtually *walk* on the streets but only *teleport* among the panoramas. Bulbul and Dahyot [22] further reconstruct three cities with the street view data and visualize the popularity and sentiments with virtual spots lights. However, their system requires 113 - 457 minutes to

[11]A mirrored world is defined as "a representation of the real world in digital form [which] attempts to map real-world structures in a geographically accurate way" [76].

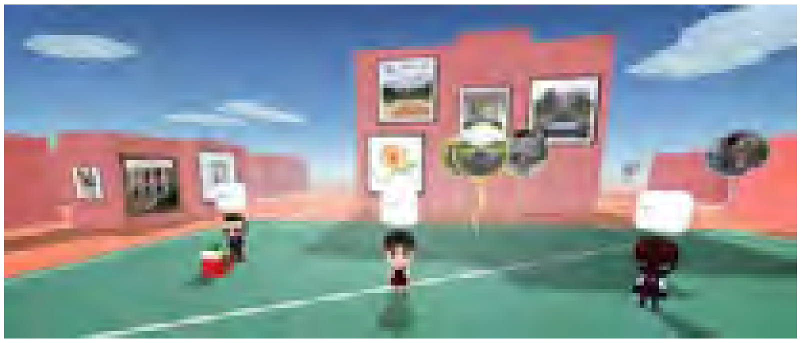


Figure 2.20: *Geollery* creates an virtual *mirrored world* where users are immersed with three-dimensional buildings and geo-tagged social media. The *mirrored world* corresponds to the real world structures and content. Social media messaged from Twitter, Yelp, Flickr, and our own system are visualized as balloons, billboards, framed photos, and gift boxes *in real-time*.

reconstruct each city and does not explore the visualization and interactivity of social media at the street level. Kukka *et al.*[23] have presented the conceptual design of visualizing street-level social media in a 3D virtual city, VirtualOulu [31]. However, such pre-designed 3D city models are not practical for deployment in larger areas.

In this section, we present *Geollery* (Figure 2.20 and 2.21 D), an interactive mixed-reality social media platform in 3D which uses a mirrored world rendered in *real-time*. We introduce a progressive pipeline that streams and renders a mirrored world with 3D buildings and geotagged social media. We extend the design space in several dimensions: progressive meshes and view-dependent textures, virtual representations of social media, aggregation approaches, and interactive capabilities.

To evaluate the system and envision the future of 3D social media platforms, we conduct a user study with semi-structured interviews with 20 people. The qualitative evaluation and individual responses reveal the strengths and weaknesses of both systems. We further summarize the challenges and limitations of the systems, as well as the types of decisions these could influence and their potential impact. Finally, we improve *Geollery* according to the user feedback and deploy it via Amazon Web Services (AWS) at `https://geollery.com`.

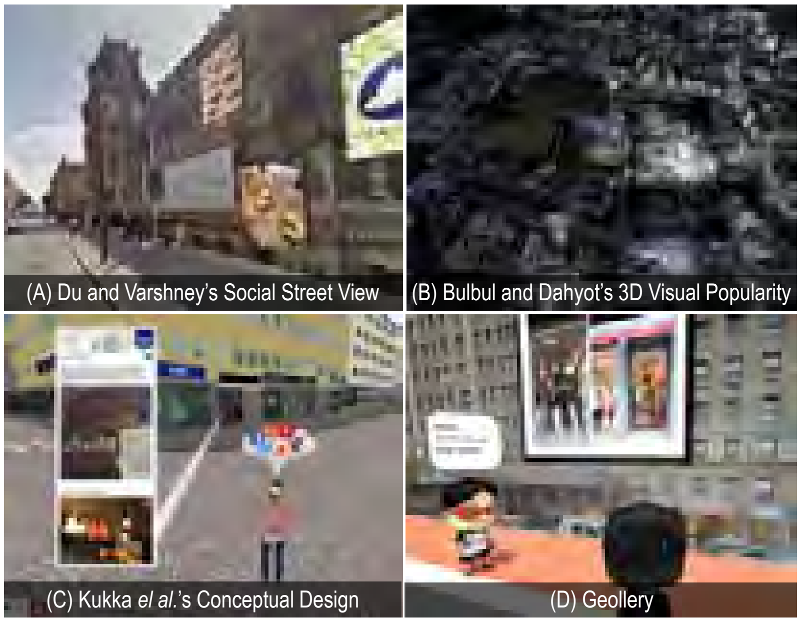

(A) Du and Varshney's Social Street View
(B) Bulbul and Dahyot's 3D Visual Popularity
(C) Kukka el al.'s Conceptual Design
(D) Geollery

Figure 2.21: Comparison amongst prior mixed reality systems or designs for visualizing geotagged social media. (A) shows *Social Street View*, a real-time system which depicts social media as billboards over building walls, (B) shows Bulbul and Dahyot's offline system [22] which leverages virtual lighting to visualize popularity and sentiments of social media, (C) shows the conceptual design by Kukka *et al.*[23], which explores presentation manner, visibility, organization, and privacy during co-design activities, and (D) shows our results in *Geollery* v2, which fuses 3D textured buildings, geotagged social media, and virtual avatars in *real-time*.

Our contribution is summarized as follows:

1. conception, design, and development of *Geollery*, an online system that can depict geotagged social media, 3D buildings, and panoramas in an immersive 3D environment,

2. extending the design space of 3D social media platforms to include aggregation approaches, virtual representations of social media, co-presence with virtual

avatars, and interactivity,

3. conducting a user study to qualitatively compare two 3D social media platforms (*Geollery* and *Social Street View*) by discussing their benefits, limitations, and potential impact on future 3D social media platforms.

2.9.2 System Overview

In this section, we present an overview of *Geollery*'s system architecture. *Geollery* consists of a data engine which streams 2D polygons and labels from *OpenStreetMap*[12] and social media data from our internal database or external sources such as *Twitter*[13], *Yelp*[14], *Instagram*[15], and *Flickr*[16]. Deployment of *Geollery* requires only an SQL database and a web server powered by Apache and PHP. We take advantage of the $B+$ tree to index the geotagged information for querying in real-time. We build the rendering system upon `Three.js`[17], a cross-browser, GPU-accelerated JavaScript library.

Geollery allows users to explore social media nearby or at a custom location. Users have a choice of either sharing their device's current location or entering a query into a search box powered by Google's Place Autocomplete API[18]. Unlike the prior art which aims at reconstructing the entire city, *Geollery* leverages a progressive approach to partially build the mirrored world. We present the workflow of *Geollery*

[12]OpenStreetMap: `https://openstreetmap.org`, an open world map.
[13]Twitter: `https://twitter.com`, social networking service.
[14]Yelp: `https://yelp.com`, local city guide.
[15]Instagram: `https://instagram.com`, a photo-sharing platform.
[16]Flickr: `https://flickr.com`, an image hosting service.
[17]Three.js: `http://www.threejs.org`.
[18]Place Autocomplete API: `https://goo.gl/4eTK5y`.

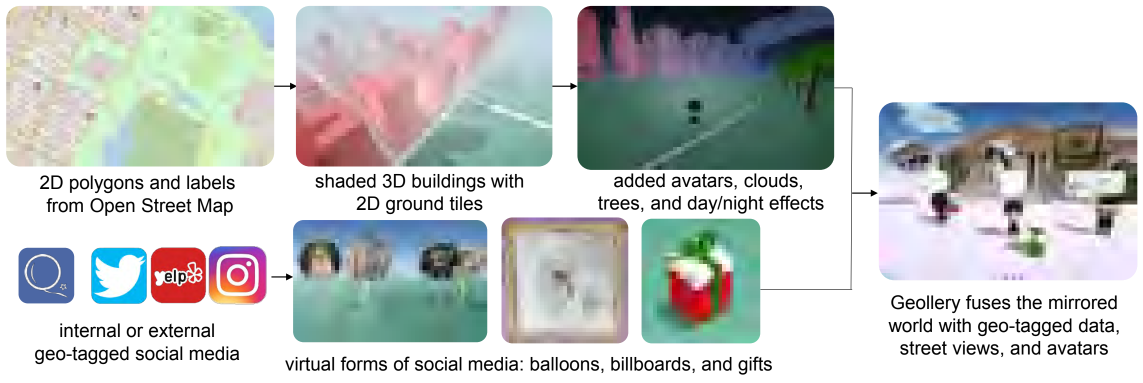

2D polygons and labels from Open Street Map
shaded 3D buildings with 2D ground tiles
added avatars, clouds, trees, and day/night effects
Geollery fuses the mirrored world with geo-tagged data, street views, and avatars
internal or external geo-tagged social media
virtual forms of social media: balloons, billboards, and gifts

Figure 2.22: The workflow of *Geollery*. Based on users' geo-location requests, *Geollery* loads the nearby 2D map tiles, extrudes 3D geometry, and renders social media in real-time. We take advantage of WebGL to enable users to access *Geollery* via modern browsers on a desktop, a mobile phone, or a head-mounted display.

in Figure 2.22.

First, given a pair of latitude and longitude coordinates, our system queries 2D map tiles and renders the ground plane within a radius of about 50 meters (depending on the user's specification). The ground plane visualizes roads, parks, waters, and buildings with a user-selected color scheme. As users virtually walk on the street, *Geollery* streams additional data into the rendering system. Next, *Geollery* queries 2D map data from *OpenStreetMap* to gather information about buildings and terrains. 3D geometry is extruded from 2D polygons and then shaded with the appropriate lighting and shadows to form buildings. Trees are randomly generated in forests.

In Section 2.9.4, the user study was conducted in *Geollery* v1 without building textures. *Geollery* v2 is able to project the closest panorama onto the 3D geometry, based on the availability of street view data (see Section 2.9.5 for technical details). Finally, the system renders a mirrored world within the user's field of view in real-time, which contains 3D buildings, virtual avatars, trees, clouds, and different forms of social media, such as balloons, billboards, framed photos, and virtual gifts.

For registered users, *Geollery* connects the client with the back-end server via a web socket, which allows real-time communication and collaboration with other participants nearby. We explain our design and implementation details in the next section.

2.9.3 Design Space

As listed in Table 2.2, we explore and compare several variables in the design space of 3D social media platforms between *Geollery* and *Social Street View* [1], including the choice of meshes and textures, availability, degrees of freedom (DoF) in motion, virtual avatars, and social media representations. We further discuss other dimensions of interest such as privacy concerns, real-world phenomena, and temporal filters.

2.9.3.1 Meshes and Textures

During the design process of selecting meshes and textures, we consider the tradeoff between the processing speed and visual appearance. While prior art [20, 21, 22] has presented various approaches to reconstruct textured 3D buildings in minutes or hours, we prefer a progressive approach to only reconstruct the nearby building geometry. This allows us to create buildings in real-time as needed. We circumvent preconstructed models to allow *Geollery* to be used at any location where 2D building data are available.

Variable	Geollery	Social Street View
Mesh	Ground, 3D Buildings, trees, and clouds	Sphere
Textures	Geollery v1: No texture Geollery v2: With 360° street views	Textured by 360° street views
Availability	Almost always available	Only available for the locations with 360° street view data
Motion	6 DoF	3 DoF + Teleport
Virtual Avatar	Available	Not applicable
Collaboration	Available	Not applicable
Social Media Location Accuracy	Almost the exact location in the world	Estimated by distance and orientation
Virtual Representation	Billboards / Balloons / Framed photos / Doodles / Gifts	Billboards (v2: added balloons and gifts)
Aggregation	Based on spatial relationship	Based on direction and distance

Table 2.2: Comparison between *Geollery* and *Social Street View* with different variables. Note that the original version of *Social Street View* only uses billboards as a virtual form of social media while the latest version also uses balloons and virtual gifts.

Social Street View is another approach for real-time rendering of immersive street-level environments with geotagged social media. Nevertheless, it reconstructs textured spheres with depth maps and normal maps rather than 3D building blocks. Since the building geometries are not fully recovered, its degree of freedom in motion is limited to pitch, roll, and yaw. Users have to *teleport* to the other locations by clicking on the streets or a 2D map.

To achieve six degrees of freedom movement, we progressively stream data from *OpenStreetMap* to build 3D meshes in real time. *Geollery* extrudes polygons of nearby buildings into 3D blocks according to metadata such as building heights (usually available in the metropolis) and building levels. Although this approach cannot reconstruct complex geometries such as the Effiel Tower or the London Eye, it provides the spatial context necessary for augmented reality scenarios (when the user holds a mobile device).

In the first version of *Geollery*, we explore different color schemes for visualizing the mirrored world. Considering the users' feedback from the user study, we add images from *Google Street View* to *Geollery*, so that the closest street views are rendered with the building geometries in real-time. We discuss the technical details in Section 2.9.5. Note that while Google Earth provides textured meshes of buildings for many cities, their data is not yet publicly available and the texturing quality is not as high as street views. We believe that in the future, streaming large-scale high-quality textured meshes is the vital key to developing an immersive social media platform in the mirrored world.

2.9.3.2 Interactive Capabilities

The real-time mirrored world enables new interactive capabilities in *Geollery*. Here, users can see nearby friends as virtual avatars, chat with friends, and paint street art collaboratively on the virtual building walls.

Avatars. First-time visitors to *Geollery* are asked to select a 3D avatar from a collection of 40 rigged models. These models are stored in glTF[19] format for efficient transmission and loading in the WebGL context. After selecting an avatar, users

[19]glTF: `https://github.com/KhronosGroup/glTF`, GL Transmission Format.

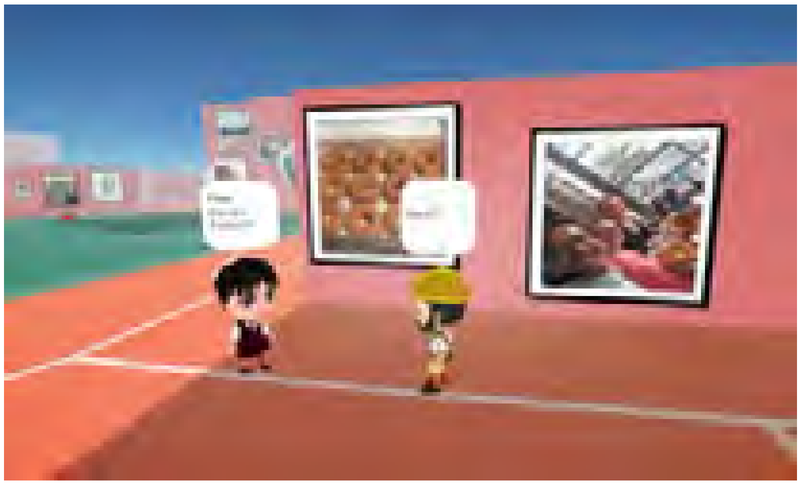


Figure 2.23: Real-time communication in *Geollery* v1 with geotagged social media and virtual buildings may help the users for better spatial context.

can use the keyboard or the panning gesture on a mobile device to virtually walk in the mirrored world.

Chat. As shown in Figure 2.23, when two participants virtually meet with each other, *Geollery* allows them to chat with each other in the form of text bubbles. Users can click on other avatars to send private chat messages or their own avatar to send public chat messages.

Collaborative Street Art. Inspired by street art, *Geollery* enables two or more users to share a single whiteboard, draw on it, and add pictures or text via WebSockets. The server updates drawings on nearby users' canvases after every edit allowing real-time collaboration.

2.9.3.3 Virtual Representations of Social Media

In classic 2D interfaces, social media are usually laid out linearly (*Twitter*, *Instagram*) or in a grid (*Pinterest*) within the screen space. Nevertheless, in a 3D space, the virtual forms of social media can have more diversity. Taking locations, real-world analog, and gamification into account, we have designed the following four virtual representations of social media:

(a) **Billboards**. Billboards, newsstands, and posters are widely used in the physical world for displaying information. As shown in Figure 2.24(a), billboards show thumbnails of geotagged images or text. We implement four levels of details for thumbnails: 64^2, 128^2, 256^2, and 512^2 pixels. The resolution of the thumbnail shown on each billboard is inversely proportional to the squared distance of the avatar to the billboard. Higher resolution thumbnails are loaded as the avatar approaches each billboard. When users hover over a billboard, the system reveals associated text captions (four lines at most). When users click on a billboard, it pops up a window with details including the complete text caption, the number of likes, and any user comments.

(b) **Balloons**. To attract users' attention and sustain their interest, we design floating balloons in Figure 2.24(b) to showcase nearby social media. The border colors of balloons categorize their social media based on the text of each social media post.

(c) **Framed photos or street art**. These two representations are respectively inspired by galleries and street art. *Geollery* allows the users to put on framed photos or a public whiteboard on building walls. If the creator of the whiteboard allows, nearby users can collaborate in drawing doodles or writing text on the whiteboard. When creating whiteboards, users have the option of selecting from multiple sizes and frame styles.

(d) **Virtual gifts**. To encourage users to engage with their friends, we design virtual gift boxes. Users can leave a gift box at any location in the world and their friends can open it and get rewards in the form of a message or a picture. Gifts can also be secured via questions and answers.

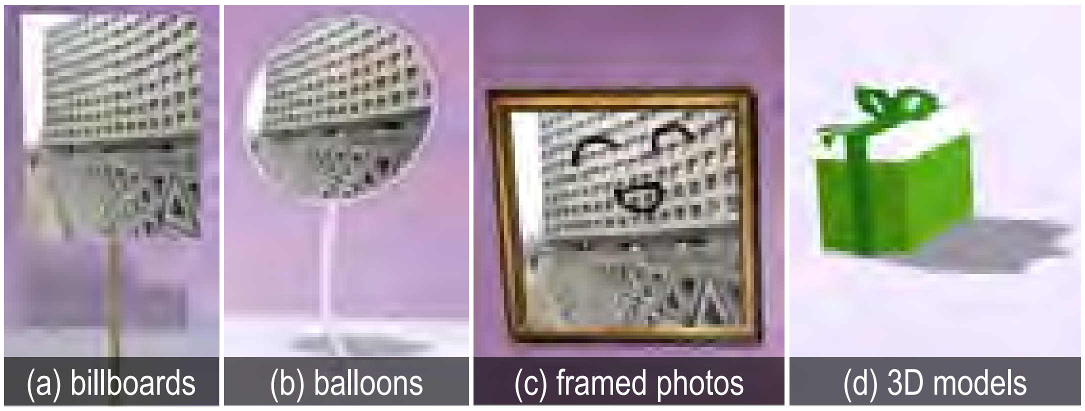


Figure 2.24: Four virtual representations of geotagged social media: (a) billboards, (b) balloons, (c) framed photography, and (d) 3D models such as gift boxes.

Geollery allows users to create billboards, balloons, or gift boxes at their avatar's location by uploading photos or text messages. To create a framed photo or whiteboard, users simply click on or touch an empty section of virtual wall with the drawing mode enabled. *Geollery* hangs the frame outside the building by aligning the normal vectors of the wall and the frame [1].

2.9.3.4 Aggregation Approaches

One of the challenges of visualizing a large amount of social media in 3D spaces is visual clutter. When multiple social media are placed at approximately the same location, their virtual representations may occlude with each other. We propose three modes as shown in Figure 2.25 to resolve this issue:

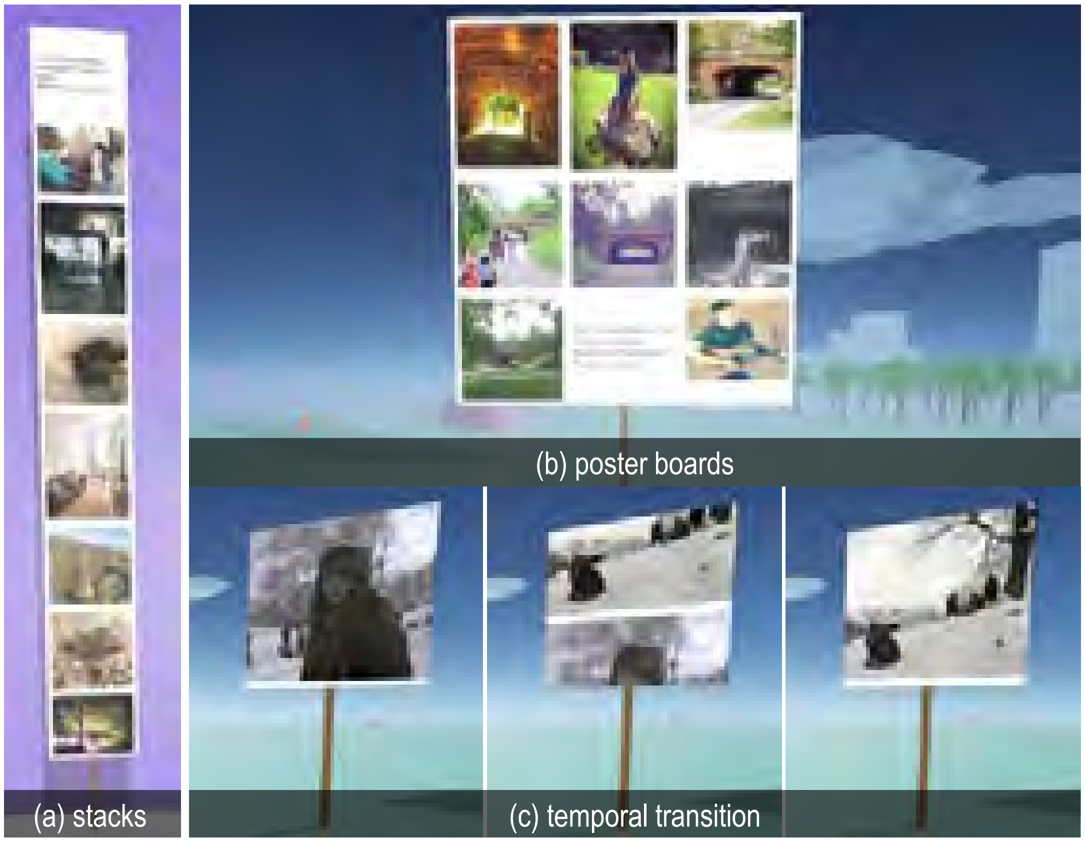


Figure 2.25: *Geollery* spatially aggregates social media into: (a) stacks, (b) poster boards, or (c) a single billboard or balloon with temporal transition.

(a) **Stacks**. This mode stacks older billboards upon the newer ones so that all co-located social media can be viewed at the same time.

(b) **Poster boards**. This mode is similar to the stacks, but lays out the social media in a grid on a large poster board. Compared to stacks, posts are not placed as high when more than three are aggregated together.

(c) **Temporal transition**. This mode clusters nearby social media within a radius of approximately 12 meters into a single standard size billboard or balloon. The content displayed dynamically changes between aggregated social media every 10 seconds. This method greatly reduces the visual clutter while displaying the latest information to the user.

The advantage of stacks or poster boards is that multiple information is displayed at a glance, while the advantage of temporal transition is reducing the visual clutter. We provide all options and discuss the users' preferences in Section 2.9.4

2.9.3.5 Privacy

When designing *Geollery*, we take privacy concerns into consideration at an initial stage since location data may reveal details of people's lives [78]. We tackle privacy in three aspects:

1. **Social Media**. When creating social media, users can select among multiple privacy options including: only available to themselves, only visible to friends, and visible to the public. Although we do not support tagging on photos now, we note that future systems with the tagging feature should mitigate the multiparty privacy conflicts [79].

2. **Avatar**. Users can set their avatar to be invisible to prevent exposing themselves to the public. Users can also customize their display name to remain

anonymous in the mirrored world.

3. **Street Art**. Users can restrict rights to editing the street art they create on the building walls or allow other people to collaborate on their street art.

2.9.3.6 Real-world Phenomena

As suggested in [1, 23], real-world phenomena such as day and night transitions and changing seasons make virtual worlds more alive and realistic. In *Geollery*, we have designed a day/night transition system which adjusts the lighting and sky based on the local time of the user.

2.9.3.7 Filtering of Social Media

Applying topic models [2, 80] and temporal filters [43, 45] to social media has been researched intensively in recent years. In *Geollery* v2, we allow the users to filter the social media within the day, month, or year, or by keywords.

2.9.4 User Study

To discover potential use cases and challenges in designing a 3D mixed-reality social media platform, we evaluated our prototype, *Geollery* v1, against another social media system, *Social Street View*, in a user study with semi-structured interviews. The key differences between the two systems are discussed in Section 2.9.3 and Table 2.2.

We recruited a total of 20 participants (ten females and ten males; average

age: 25.75 ± 3.02) via campus email lists and flyers. Each participant was paid 10 dollars as compensation for their time and effort. None of the participants had been involved with this project before. The individual semi-structured interviews took place in a quiet room using two side-by-side workstations with 27-inch displays and NVIDIA GTX 1080 graphics cards. Participants interacted with the systems using keyboards and mice alongside the interviewer. The session for each participant lasted between $45 - 60$ minutes and involved four stages: a background interview, an exploration of *Geollery* and *Social Street View*, a quantitative evaluation, and a discussion about the future of 3D social media platforms.

2.9.4.1 Background Interview

In the first stage (5 minutes), the interviewer introduced *Geollery* and asked the participant about their prior experiences of social media. All of our participants reported social media usage of at least several times per week. Furthermore, 16 out of 20 responded *several times per day.* However, only 5 out of 20 posted social media frequently (more than several times a week): *"I post news about sports and games every day."* (P7/M); *"I majorly use Instagram, I post from my own portfolio."* (P17/F). The rest of our participants primarily use social media for viewing friends' updates and photos.

2.9.4.2 Exploration of *Geollery* and *Social Street View*

In the second stage (30-40 minutes), the interviewer instructed the participant to virtually visit four places using each of the target technologies, *Geollery* and *Social Street View*. The places participants were asked to explore include the university campus where the study took place, the Manhattan District of New York, the National Gallery of Art in Washington D.C, and a custom location of the participant's choice. We counterbalanced the order of technology conditions (*Geollery* or *Social Street View*), as well as ordered the first three places using the Latin square design [81]. For the duration of the study, the interviewer observed the participants' behaviors and took notes about their comments and interaction.

First, the participant was asked to choose a nickname and an avatar. Meanwhile, the interviewer logged in to the same system on the other workstation so the participant could virtually interact with the interviewer.

Next, we asked if the participant was aware of their location in each virtual setting. In *Social Street View*, all participants quickly figured out their virtual locations. In *Geollery*, participants who noticed the minimap would immediately know where they were, but four out of 20 users became confused. For example, P5/F asked: *"Am I in a gallery?"*, and P16/M responded: *"I believe I am in a museum."*

After allowing the participants to freely explore each interface for 3 minutes, we interviewed them about their first impressions. In *Geollery*, many participants were amazed by walking in the mirrored world and the progressive loading of the geometries: *"I think it's a very good start, it's very good experience to walk around."* (P6/F); *"I like that the buildings are forming while I am walking."* (P16/M); *"I really like the fact that it's scaled, so I don't have to walk 15 minutes from one place to the other."* (P17/F).

In *Social Street View*, many participants appreciated the texturing of the 360 views: *"I think the texturing actually helps me."* (P17, F); *"It's like you don't have to be there."* (P11, F).

However, several participants found *Social Street View* frustrating in that they could not freely *walk* around, but only *teleport* by clicking the mouse: *"So how do I walk here?"* (P5/F) [The interviewer instructed her how to teleport.] *"Oh, I see, it zooms in when I scroll. It's like Google Street View."*

We further asked their preferences of different virtual representations, different aggregation methods, and whether they preferred the system to read out the social media contents on demand. In regards to billboards versus balloons, 14 out of 20 participants preferred balloons: *"Balloons are informal and billboards can have notices. Balloons may be better for social media."* (P13/F). The other 6 participants preferred the billboards: *"I like billboards. First thing, balloons keep moving, it's a little distracting. Billboards look like you are announcing something. It's neater"* (P17/F). In addition, 75% of participants preferred the temporal transition approach to aggregate nearby social media into one billboard or balloons.

In the end, we encouraged the participants to input any desired location and compare *Geollery* with *Social Street View*. Most participants chose their homes as a final destination while a few participants input locations where *only Geollery* is available. For example, P12/M typed the Statue of Liberty in New York City, where only *Geollery* was able to present the geotagged social media with their spatial context, Liberty Island.

2.9.4.3 Quantitative Evaluation

After exploring the two interfaces for 30 minutes, we asked the participant to comparatively and quantitatively rate the two systems along 9 attributes in an AttrakDiff-based 9- point (-4, -3, -2, -1, 0, 1, 2, 3, 4) antonym word pair questionnaire inspired by [23]. The average ratings are visualized as a radar chart in Figure 2.26. From a Welch's paired t-test, we found a significant effect for interactivity ($t(20) = 3.04, p < 0.01, \text{Cohen's } d = 0.83$) and creativity ($t(20) = 2.10, p < 0.05, \text{Cohen's } d = 0.66$) with *Geollery* outperforming *Social Street View.* In addition, 14 out of 20 found *Geollery* more or equally immersive compared to *Social Street View* and 16 out of 20 found *Geollery* more or equally entertaining compared to *Social Street View.*

We then asked the participant which system appealed more to them. Overall, more participants (13 out of 20) preferred *Geollery* to *Social Street View* due to its interactivity: *"I prefer Geollery in terms of moving around, and because you have the options to draw on walls and interact with people."* (P17/F) *"I like Geollery because I have free roaming there, and it's kind of cool that I can walk over the world."* (P11/F)

Several participants pointed out that *Geollery* is more like a massively mul-

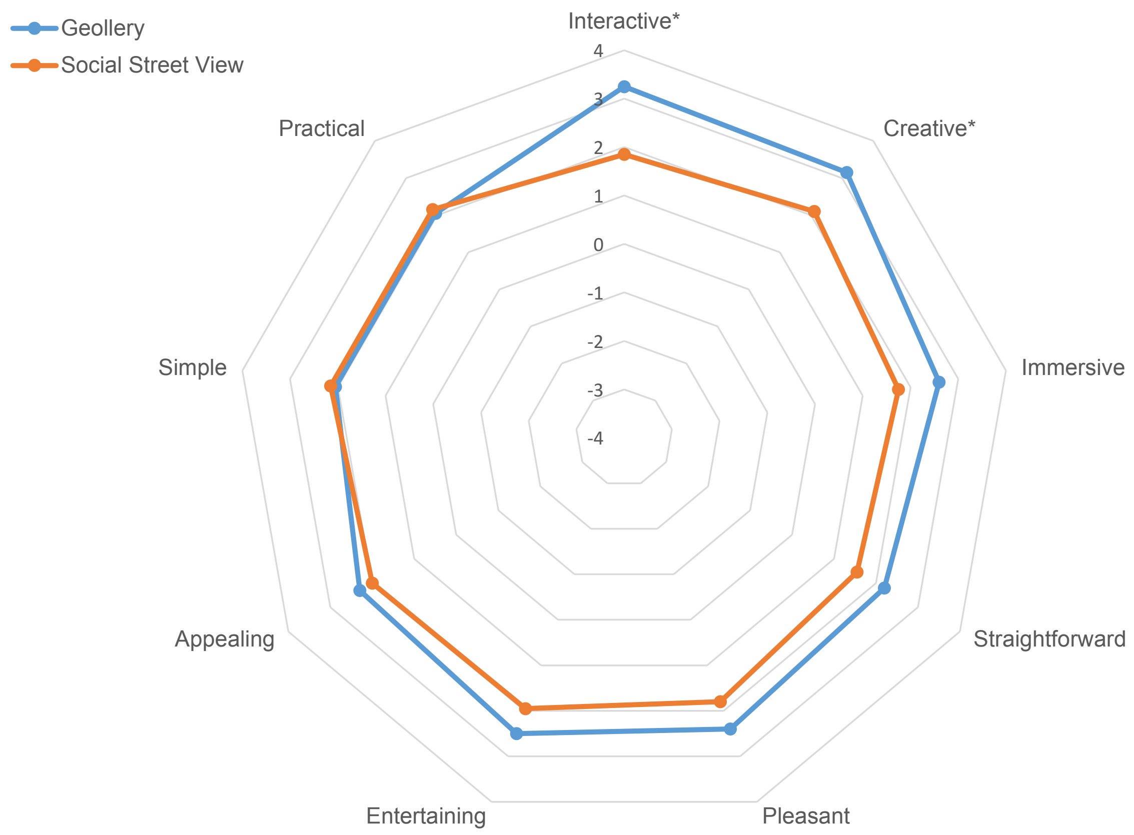


Figure 2.26: The radar chart visualizes the quantitative evaluation between *Geollery* and *Social Street View* along 9 dimensions with 20 participants. With a Welch's paired t test, there is a statistically significant effect ($p < 0.05$) that *Geollery* is rated more interactive and more creative than *Social Street View*. We have indicated this with a star superscript; other dimensions were not found to be different in a statistically significant sense.

tiplayer online (MMO) game: *"That one (Geollery) I was in a game. This one (Social Street View) feels depressing, nothing exciting."* (P18/M); *"I think it's more like a game, it's more fun to interact with the virtual world."* (P19/M); *"Having more people makes the place feel more interesting and immersive."* (P10/F).

In this study, the participants did not try *Geollery* v2 with textured buildings. Some participants preferred *Social Street View* due to the immersive panoramas: *"[In Geollery,] the buildings don't like the buildings in the real world, but Social Street View allows me to explore my environment."* (P13/F); *"I like Social Street*

View better. There, I understand the environments better." (P16/M)

2.9.4.4 The Future of 3D Social Media Platforms

At the end of the user study, we asked the participant to discuss their expectations as users of future 3D social media platforms and the features they would add if they were designers or product managers. We interviewed the participants with the following three questions:

1. Suppose that we have a polished 3D social media platform like *Geollery* or *Social Street View*, how much time would you like to spend on it?

For this question, we categorize our participants into three classes: supporters, followers, protesters. Supporters (75%, 15 out of 20) are generally more optimistic about the future of 3D social media platforms. They envision *Geollery* or *Social Street View* being used for daily exploration or trip planning. Here are some responses: *"I would like to use it every day when I go to work or travel during weekends. [...] I may spend about 8 hours per week using it."* (P4/M); *"If it's not distracting like Facebook and Instagram, I would use it everyday on a couple of things."* (P17/F); *"I love traveling, [so] I would like to use it [Social Street View] to preview my destinations before my trips."* (P3/M).

The followers (4 out of 20) typically preferred to switch to 3D social media platforms once their friends joined. For example, here are some followers' responses: *"I am a follower on most social media sites. I would only join a 3D social media* *platform once my friends are there."* (P4/M); *"If my friends are all on this, I can see myself spend a couple of hours every week. We can have a meet-up point at one place. My friends could go to my home and post social media."* (P12/M); *"It depends on who is using it. If many friends of mine are using it, I would also use it."* (P20/F).

As for protesters (1 out of 20), P2/F responded: *"I don't think I will use this. I prefer to use Yelp to see comments [of nearby restaurants].".*

2. Can you imagine your use cases for *Geollery* and *Social Street View*? What would you like to use 3D social media platforms for?

Many participants (17 out of 20) mentioned food and travel planning as their majority use cases: *"I would like to use it for the food in different restaurants. I am always hesitating of different restaurants. It will be very easy to see all restaurants with street views. In Yelp, I can only see one restaurant."* (P13/F); *"[I will use it for] exploring new places. If I am going on vacation somewhere, I could immerse myself into the location. If there are avatars around that area, I could ask questions."* (P17/F); *"I would like to use it to learn more about the world. If there is a restaurant, I would like to click the restaurant to pull the menu. It makes it easier to communicate with the local people. "* (P20/F)

Family gathering and virtual parties are also potential use cases according to the participants' responses: *"I think it (Geollery) will be useful for families. I just taught my grandpa how to use Facetime last week and it would great if I could* *teleport to their house and meet with them, then we could chat and share photos with our avatars."* (P2/F); *"...for communicating with my families, maybe, and distant friends, [so] they can see New York. And, getting to know more people, connecting with people based on similar interests."* (P19/M); *"We can use it (Geollery) on parties [...] like hide some gifts around the house and ask people to find."* (P4/M).

3. If you were a designer or product manager for *Geollery* or *Social Street View*, what features would you like to add to the systems?

Many participants mentioned texturing the buildings on *Geollery* v1: *"A mapping of the texture, high-resolution texture, will be great."* (P12/M); *"if there is a way to unify the interaction between them, there will be more realistic buildings [and] you could have more roof structures. Terrains will be interesting to add on.* (P18/M).

Participants suggested more data be integrated into 3D social media platforms: *"If I'm shopping around in a mall, if I could see deals and coupons, and live comments..."* (P7/M); *"I would like to add traffic and parking information."* (P6/F).

Many participants also suggested a better avatar system, more 3D objects, and more interactive capabilities in *Geollery*: *"[I would like] the flexibility to build your own avatar. Customizing avatar will be one useful feature."* (P18/M) *"I would like to see kitties and puppies running around, and birds flying in the air."* (P13/F) *"I could also add a bike, add a vehicle, a motorcycle in Geollery, this will add some* *fun.”* (P17/F).

2.9.5 Discussion

In this section, we summarize the key insights we learned from the user study, as well as the further improvement we have achieved since the user study.

2.9.5.1 Insights from User Study

From the user study with 20 participants, we summarize our findings and insights as follows:

1. Data sources of social media play key roles in developing a 3D social media platform. Since many users do not post to social media frequently, obtaining high-quality data from external sources or seed users to generate high-quality content is of great significance.

2. Interactivity and panoramic textures have different levels of importance for different groups of users. Users with better geospatial awareness may appreciate more on interactivity in *Geollery* while others may appreciate more on the panoramic texturing. Nevertheless, we believe that an ideal system would incorporate high-quality textures into *Geollery*, resulting in a faithful and interactive mirrored world in real-time.

3. Customization of avatars, diversity [82], and accessibility [83] are important for developing future 3D social media platforms. All users should be able to represent themselves and share the virtual mirrored world equally.

2.9.5.2 Combining *Geollery* and *Social Street View*

Thanks to the participants' feedback, we develop *Geollery* v2 which combines progressive geometries with street views to create textured buildings. We achieve this by projecting street views from Google onto the building geometries in *Geollery*. In the fragment shader, we compute the directional vector from the closest street view's location and use the spherical coordinates of this vector to sample the street view image. As users walk around in *Geollery*, we continuously update the closest street view and use alpha blending to transition to textures obtained from new street views. This approach works for many urban areas in *real-time*. Nevertheless, as shown in Figure 2.27, this algorithm may project trees or the sky onto the geometries due to the approximation of the digital city. Accurate real-time creation of textured buildings from street view images remains an open challenge even in the state-of-the-art reconstruction systems [20, 21, 24]. Future development may take advantage of deep neural networks to semantically segment the sky [84, 85, 86] or in-paint the pedestrians [87, 88].

2.9.5.3 Mobile and Virtual Reality Modes

In mobile platforms, *Geollery* allows users to track their device (both the location and the orientation) and explore the nearby social media while walking in the real world. Furthermore, users can hide secret gifts in *Geollery* and ask people to seek for it in the real world. We envision a future mobile version of *Geollery* using augmented reality technology to directly overlay geotagged social media on

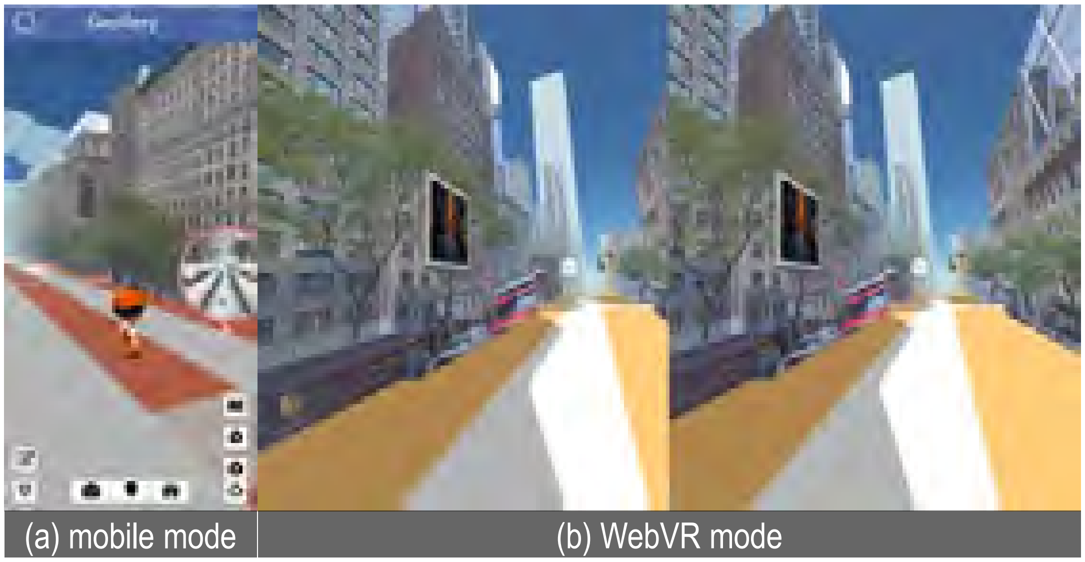


Figure 2.27: We further combine *Geollery* and *Social Street View* by texturing the buildings in *Geollery* v2: (a) shows a screenshot on an Android mobile phone, where the user could track the device's current location and orientation and explore social media in the mirrored world; (b) shows a screenshot of the WebVR mode, where we provide the user with a first-person experience to walk around via a VR controller.

the ground or building walls.

Additionally, via WebXR[20] APIs, *Geollery* allows users to control their avatars with VR controllers (*e.g.,* an Oculus Touch controller or HTC Vive controller) in head-mounted displays (HMD). However, the current generation of commercial HMDs suffers from relatively low resolutions. Consequently, text and images appear blurrier than when displayed on a conventional monitor. In the future, we plan to investigate how virtual reality HMDs will change user experiences in *Geollery* and whether they will be more efficient and immersive than conventional displays.

[20]WebXR: `https://immersive-web.github.io/webxr`

2.10 Conclusions and Future Work

In this chapter, we have presented *Social Street View*, a system to create immersive social maps that blend street view panoramas with geotagged social media. Our contributions include: (a) system architecture to scrape, query, and render geotagged street view and social media together on clients ranging from smartphones to tiled display walls to head-mounted displays using WebGL, (b) techniques to carefully layout and display social media on virtual billboards by a judicious combination of depth maps, normal maps, and maximal Poisson sampling, and (c) validating the efficiency of such mixed-reality visualizations for saliency coverage. We have also presented several potential use cases of exploring social media with temporal and spatial filters and storytelling with spatial context. The supplementary materials and demos are available at `http://socialstreetview.com`.

We have also presented *TopicFields*, a novel system to explore, summarize, and visualize geotagged social media with hybrid topic models and scalar field. *TopicFields* can efficiently estimate the kernel density distribution and visualize the scalar fields of the user-selected topics on a map on the GPU. We have presented the system and its architecture that ingests geotagged Instagram and Twitter messages, extracts topics, hierarchically clusters, and facilitates their interactive visualization on a map. The advantages of using *TopicFields* are that it allows a large volume of spatial and temporal data to be visualized and understood, then correlated with a series of topics. Our system includes an efficient and interactive GPU-driven visualization algorithm for visualizing multi-variate scalar data with kernel density estimation and non-linear normalization methods.

We have further developed the successor of *Social Street View*, *Geollery*, in which we progressively blend immersive maps with 3D buildings, virtual avatars, and different virtual representation of social media. We introduce our system architecture, design choices, and implementation details. We conduct a user study with semi-structured interviews to examine the challenges and limitations of the interfaces, as well as the types of decisions these could influence and their potential impact. The qualitative results indicate that *Geollery* is more interactive and creative than *Social Street View*. The user responses reveal several key use cases including searching for food, travel planning, and social gatherings. Taking the participants' feedback into account, we further combine *Geollery* and *Social Street View* by mapping the closest street view textures onto the building geometries.

There are several future directions for improving *Social Street View* and *Geollery*. First, we plan to fuse multiple street views onto the building geometries in real-time to achieve better photo-realistic rendering. Second, we aim to integrate additional useful information into the 3D world such as geotagged sales, services, and job listings. Adding mental health [89] and sentiments extracted from social media as well as live surveillance videos [90] could prove useful for and law enforcement. Third, we intend to use techniques from previous research [91] to improve the filtering mechanism, encouraging supportive comments and reducing negative emotions in *Geollery*.

As we obtain more users in *Geollery*, we envision future 3D social media platforms playing a significant role in the realm of mixed reality. They may eventually change the way we consume and create data, as well as the way we socialize with other people.

Chapter 3: Spherical Harmonics for Saliency Computation and Virtual Cinematography in 360° Videos

3.1 Introduction

With recent advances in consumer-level virtual reality (VR) head-mounted displays (HMD) and panoramic cameras, omnidirectional videos are becoming ubiquitous. These 360° videos are becoming a crucial medium for news reports, live concerts, remote education, and social media. One of the most significant benefits of omnidirectional videos is immersion: users have a sustained illusion of presence in such scenes. Nevertheless, despite the rich omnidirectional visual information, most of the content is out of the field of view (FoV) of the head-mounted displays, as well as human eyes. The binocular vision system of human eyes can only interpret 114° FoV horizontally, and 135° FoV vertically [92]. As a result, over 75% of the 360° videos are not being perceived. Furthermore, as shown in Table 3.1, almost 90% pixels are beyond the FoV of the current generation of the consumer-level VR HMDs[1].

Therefore, predicting where humans will look at, *i.e.*, saliency detection, has

[1]Data sources: the official websites of Oculus, HTC Vive, Samsung, blog posts `https://goo.gl/eBqpvm`, and `https://goo.gl/n7Vji3`

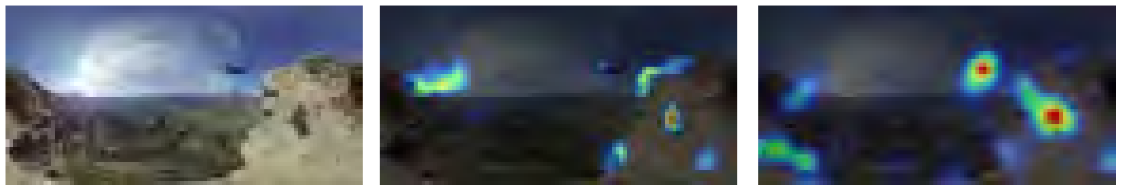

(A) The input 360° video frame **(B)** Saliency map by Itti *et al.*'s model **(C)** Saliency map by our SSR model

Figure 3.1: This chapter presents an efficient GPU-driven pipeline of computing saliency maps of 360° videos using spherical harmonics (SH). (A) shows an input frame from a 360° video. (B) shows the saliency maps computed by the classic Itti *et al.*'s model in 104.46 ms on the CPU. (C) show the saliency maps computed by our spherical spectral residual (SSR) model in 21.34 ms on the CPU and 10.81 ms on the GPU. In contrast to the classic models for images in rectilinear projections, our model is formulated in the $\mathbb{SO}(2)$ space. Therefore, it remains consistent in challenging cases such as horizontal clipping, spherical rotations, and equator biases in 360° videos.

great potential over a wide range of applications, such as:

- efficiently compressing and streaming high-resolution panoramic videos under poor network conditions [93],
- salient object detection in panoramic images and videos [94],
- information overlay in panoramic images [1], videos [95], and for augmented reality displays,
- directing the user's viewpoint to salient objects which are out of the user's current field of view, or automatic navigation and synopsis of the 360° videos [96, 97, 98, 99].

Saliency of regular images and videos has been well studied thoroughly since Itti *et al.*'s work [61]. Previous research has also investigated mesh saliency [100], volume saliency [101], and light-fields saliency [102]. However, unlike classic images which are stored in rectilinear or gnomonic projections, most of the panoramic videos are stored in equirectangular projections. Consequently, classic saliency may not work for 360° videos due to the following challenges, as further shown in Figure 3.6:

- *Horizontal clipping* may slice a salient object into two parts on the left and right edges, which may cause a false negative result.
- *Spherical rotation* may distort the non-salient objects near the north and south poles, which may cause a false positive result.
- *Equatorial bias* is not formulated in the classic saliency detectors.

Visual Medium	**Approximate Field of View (FoV)**		**Ratio Beyond FoV**
	Horizontal	Vertical	
Human Eyes	114°	135°	76.25%
HTC Vive, Oculus Rift	85°	95°	87.53%
Samsung Gear VR	75°	85°	90.16%
Google Cardboard	65°	75°	92.48%

Table 3.1: Comparison of the approximate binocular field of view of human eyes, as well as the current generation of the consumer-level head-mounted displays

In this chapter, we address three interrelated research questions: (a) how should we formulate the saliency in the $\mathbb{SO}(2)$ space with spherical harmonics, (b) how should we speed up the computation by discarding the low-frequency information, and (c) how should we automatically and smoothly navigate the 360° videos with saliency maps? To investigate these questions, we present a novel GPU-driven pipeline for saliency computation and navigation based on spherical harmonics (SH), as shown in Figure 3.1.

In Section 3.2.2, we present the preprocessing for computing the SH coefficients for representing the 360° videos. Our pipeline pre-computes a set of the Legendre polynomials and SH functions and stores them in GPU memory. We adopt the highly-parallel prefix sum algorithm to integrate feature maps of the downsampled 360°frames as 15 bands of spherical harmonics coefficients on the GPU.

In Section 3.4, we introduce the Spherical Spectral Residual (SSR) model. Inspired by the spectral residual approach, we define SSR as the accumulation of the SH coefficients between a low band and a high band. This model reveals the multi-scale saliency maps in the spherical spectral domain and reduces the computational cost by discarding the low bands of SH coefficients. From the experimental results, it outperforms the Itti *et al.*'s model by over $5\times$ to $13\times$ in timing, and runs in real time at over 60 frames per second for $4K$ videos.

In Section 3.5, as a proof-of-concept, we propose and implement a saliency-guided virtual cinematography system for navigating 360° videos. We formulate a spatiotemporal model to ensure large saliency coverage while reducing the camera movement jitter.

The main contributions of our work are:

- formulating saliency *natively and directly* in the special orthogonal group $\mathbb{SO}(2)$ space using the spherical harmonics coefficients, without converting the image to $\mathbb{R}_2$,

- reducing the computational cost and formulating the spherical saliency using the spectral residual model with spherical harmonics,
- devising a saliency-guided virtual cinematography system for automatic navigation in 360° videos,
- implementing the GPU-driven real-time pipeline of computing saliency maps in 360° videos.

3.2 Related Work

Our work builds upon a rich literature of prior art on saliency detection, as well as spherical harmonics.

3.2.1 Visual Saliency

Visual saliency has been investigated in ordinary images [61, 103], videos [104], giga-pixel images [64], 3D meshes [100], volumes [63], and light fields [102]. Here, we mainly focus on image and video saliency.

A region is considered salient if it has perceptual differences from the surrounding areas that are likely to draw visual attention. Prior research has designed bottom-up [61, 105, 106, 107], top-down [108, 109, 110], and hybrid models for constructing a saliency map of images (see the review by Zhao *et al.*[111]). The bottom-up models combine low-level image features from multi-scale Gaussian pyramids or Fourier spectrum. Top-down models usually use machine learning strategies and take advantage of higher-level knowledge such as context or specific tasks for saliency detection. Recently, hybrid models using convolutional neural networks [112, 113, 114, 115, 116, 117] have emerged to improve the accuracy of saliency prediction.

One of the most pivotal algorithms for saliency detection is Itti *et al.*'s model [61]. This model computes the center-surround differences of multi-level Gaussian pyramids of the feature maps, which include intensity, color contrast, and orientations, as conspicuity maps. It further combines the conspicuity maps with non-linear combination methods and a winner-take-all network. Another influential algorithm is the spectral residual approach devised by Hou and Zhang [103]. This model computes the visual saliency by the difference of the original and smoothed log-Fourier spectrum of the image.

However, both approaches assume the input data as rectilinear images, which would not output consistent results for spherical images with horizontal clipping or spherical rotation. Inspired by these two approaches, we formulate the spherical spectral residual model in the $\mathbb{SO}(2)$ space. By efficiently evaluating the SH coefficients between two bands, our model can be easily implemented on the GPU and achieves spherical consistency.

In addition to Itti *et al.*'s model and the spectral residual model, Bruce *et al.*[118] learn a set of sparse codes from example images to evaluate the saliency of new inputs. Wang *et al.*[119] use random graph walks on image pixels to compute image saliency. Goferman *et al.*[108] consider visual organization and high-level features such as human faces in saliency computation.

Nonetheless, all of these prior approaches only work for rectilinear images. Our work, as far as we are aware, is the first to apply spherical harmonics for saliency analysis of 360° videos. The work presented in this chapter is inspired by Itti *et al.*'s model [61] and the spectral residual approach [103] used for image and video saliency.

3.2.2 Spherical Harmonics

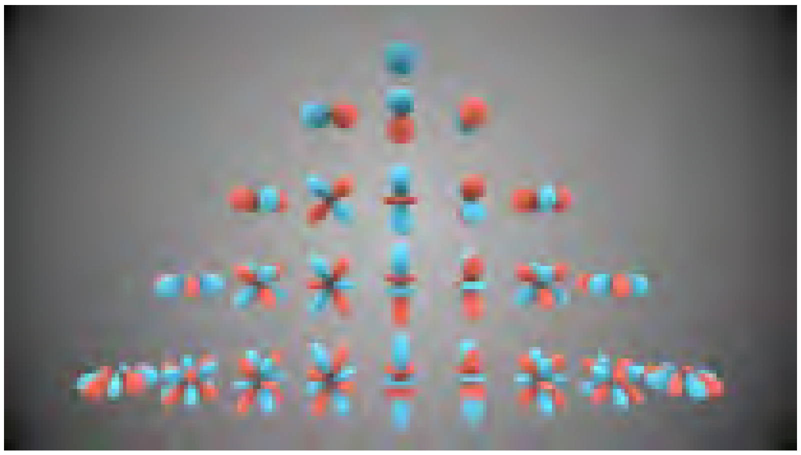


Figure 3.2: This figure shows the first five bands of spherical harmonics functions. Blue indicates positive real values, and red indicates negative real values. Our code and visualization can be viewed online interactively at `https://shadertoy.com/view/4dsyW8`. This demo is built on Íñigo Quílez's prior work.

Spherical harmonics are a complete set of orthogonal functions on the sphere (as visualized in Figure 3.2), and thus may be used to represent functions defined on the surface of a sphere. In visual computing, spherical harmonics have been widely applied to various domains and applications: indirect lighting [120, 121], volume rendering [122], 3D sound [123, 124], and 3D object retrieval [125, 126]. As for lighting, previous work in computer graphics has applied spherical harmonics to calculate global illumination and ambient occlusion [127], refraction [128], scattering [129], as well as precomputed radiance transfer [130].

To the best of our knowledge, we are the first to apply spherical harmonics for saliency detection in panoramic images and videos.

3.3 Computing the Spherical Harmonics Coefficients

Spherical harmonics coefficients are usually computed using Monte Carlo integration over the sphere [121]. 360° videos are mostly stored in equirectangular projections, where each pair of the texture coordinate (u, v) corresponds to a pair of spherical coordinate $(\theta, \phi) \longleftrightarrow (2\pi u, \pi v)$. Therefore, we can directly integrate over the scalar fields of the feature maps by using precomputed spherical harmonics at each texture coordinate. Hence, computation of the spherical harmonics coefficients is reduced to a prefix sum problem on the GPU, which is efficiently solved by the Blelloch Scan algorithm. Finally, we also show that we could downsample the panoramic image to $N \times M$ pixels while maintaining a small error in the resulting spherical harmonics coefficients. We further show that, for L bands of SH coefficients, the computational complexity is $\mathcal{O}(L^2 \log MN)$ on the GPU.

3.3.1 Evaluating SH Functions

To efficiently extract the spherical harmonics coefficients from the 360° videos, we precompute the SH functions at each spherical coordinate (θ, ϕ) of the input panorama of $N \times M$ pixels. Since the values in the feature maps, which are used to define the intensity and color contrast are positive and real, we compute only the real-valued SH functions, also known as the tesseral spherical harmonics, as shown in Figure 3.2.

The SH functions, $Y(\theta,\phi)$, are orthonormal to each other, and defined in terms of the Legendre polynomials P_l^m as follows:

$$Y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)\, P_l^m \cos(\theta), & m > 0 \\ K_l^0 P_l^0 \cos(\theta), & m = 0 \\ \sqrt{2}K_l^m \sin(m\phi)\, P_l^{-m} \cos(\theta), & m < 0 \end{cases} \tag{3.1}$$

where $0 \le l \le L$ is the band index, m is the order of the band, and $-l \le m \le l$. P_l^m are the associated Legendre polynomials:

$$\begin{aligned} P_l^l &= (-1)^l (2l-1)!!(1-x^2)^{l/2} \\ P_l^{l-1} &= [x(2l-1)]\, P_l^l \\ P_l^m &= \left[\frac{x(2l-1)}{l-m}\right] P_{l-1}^m - \left(\frac{l+m-1}{l-m}\right) P_{l-2}^m \end{aligned} \tag{3.2}$$

K_l^m is a scaling factor to normalize the functions:

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} \tag{3.3}$$

3.3.2 Evaluating SH Coefficients

To compute the SH coefficients of the 360° videos, we first extract the feature maps such as the intensity and color contrast, inspired by Itti *et al.*'s model [61] and *SaliencyToolbox* [107]. The intensity is calculated from the red, green, blue channels of each frame (r,g,b) according to [107]:

$$I = (r,g,b)^T \cdot (0.2126, 0.7152, 0.0722) \tag{3.4}$$

We also define the red-green (RG) and blue-yellow (BY) contrast for each pixel as follows:

$$RG = \frac{r-g}{\max(r,g,b)} \tag{3.5}$$

$$BY = b - \frac{\min(r,g)}{\max(r,g,b)} \tag{3.6}$$

For each feature map, the SH coefficients consist of L^2 values for L bands. In the equirectangular representation of the 360°videos, we assume that each feature $f_{i,j}$ at the coordinate $(i,j), 0 \le i < N, 0 \le j < M$ represents the mean value $f(\theta_{i+0.5}, \phi_{j+0.5})$ at the solid angle $(\theta_{i+0.5}, \phi_{j+0.5})$, where θ_i and ϕ_j are defined as:

$$\theta_i = \frac{\pi i}{N}, \phi_j = \frac{2\pi j}{M}, \tag{3.7}$$

Therefore, for the m^{th} element of a specific band l, we evaluate the SH coefficients of the feature map f as:

$$\begin{aligned} c_l^m(\theta,\phi) &= \int_{(\theta,\phi)\in S} f(\theta,\phi) \cdot Y_l^m(\theta,\phi) \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\phi \\ &= \frac{2\pi}{M} \sum_{i=1}^{N} \sum_{j=1}^{M} f_{i,j} \cdot Y_l^m(\theta_{i+0.5}, \phi_{j+0.5}) \left|\cos\theta_{i+1} - \cos\theta_i\right| \end{aligned} \tag{3.8}$$

Let

$$H_{i,j} = \frac{2\pi}{M} Y_l^m(\theta_{i+0.5}, \phi_{j+0.5}) \left|\cos\theta_{i+1} - \cos\theta_i\right| \tag{3.9}$$

we have

$$c_l^m(\theta,\phi) = \sum_{i=1}^{N}\sum_{j=1}^{M} f_{i,j} \cdot H_{i,j} \tag{3.10}$$

Hence, for a given dimension of the input frames, we can precompute the terms $H(i,j)$ and store them in a lookup table. The integration of the SH coefficients is then reduced to a conventional prefix sum problem.

3.3.3 Implementation Details

On the CPU-driven pipeline, we use *OpenMP* to accelerate the evaluation of SH coefficients with 12 threads. On the GPU-driven pipeline, we take advantage of the *Blelloch Scan* algorithm [131] with *CUDA 9* to efficiently aggregate the SH coefficients with 2048 kernels on an NVIDIA GTX 1080. The *Blelloch Scan* algorithm computes the cumulative sum in $\mathcal{O}(\log N)$ for N numbers. Therefore, our algorithm runs at $\mathcal{O}(L^2 \log MN)$ for L^2 coefficients.

Finally, we show the reconstructed image f' with the $1-15$ bands of SH

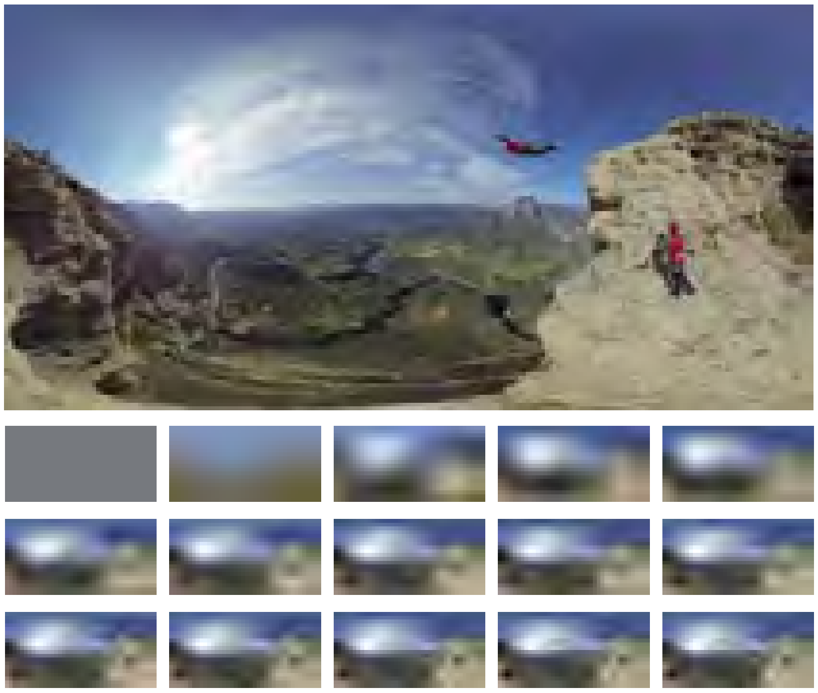


Figure 3.3: The reconstructed images with the first 15 bands of spherical harmonics coefficients extracted from the video frame.

coefficients with regular RGB color maps in Figure 3.3 with the following equation:

$$f'(\theta,\phi) = \sum_{l=0}^{L} \sum_{m=-l}^{l} c_l^m \cdot Y_l^m(\theta,\phi) \tag{3.11}$$

Note that the low-band SH coefficients capture the background information, such as sky and mountains, while the high-band SH coefficients capture the details, such as parachuters.

3.4 Spherical Spectral Residual Model

With the spherical harmonics coefficients, we present a novel approach to compute saliency for spherical 360° videos using the idea of spherical spectral residuals (SSR).

3.4.1 Spherical Spectral Residual Approach

As shown in Figure 3.3, spherical harmonics bands can be used to compute the contrast directly across multiple scales in the frequency space. In the space of $\mathbb{SO}(2)$, we define the spherical spectral residual (SSR) as the difference between the higher bands (up to Q) of SH coefficients and the lower bands (up to P) of SH coefficients:

$$\begin{aligned} \mathfrak{R}(\theta,\phi) &= \sum_{l=0}^{Q}\sum_{m=-l}^{l} c_l^m \cdot Y_l^m(\theta,\phi) - \sum_{l=0}^{P}\sum_{m=-l}^{l} c_l^m \cdot Y_l^m(\theta,\phi) \\ &= \sum_{l=P+1}^{Q}\sum_{m=-l}^{l} c_l^m \cdot Y_l^m(\theta,\phi) \end{aligned} \tag{3.12}$$

in which $Y_l^m(\phi,\theta)$ are pre-computed associated Legendre polynomials in the preprocessing stage. The SSR represents the salient part of the scene in the spectral domain and serves as a compressed representation using spherical harmonics.

For better visual effects, we square the spectral residual to reduce the estimation errors. For better visual effects, we smooth the spherical saliency maps using a Gaussian:

$$\boldsymbol{S}(\theta, \phi) = \mathfrak{G}(\sigma) * [\mathfrak{R}(\theta, \phi)]^2 \tag{3.13}$$

where $\mathfrak{G}(\sigma)$ is a Gaussian filter with standard deviation σ ($\sigma = 5$ for the results presented in this chapter).

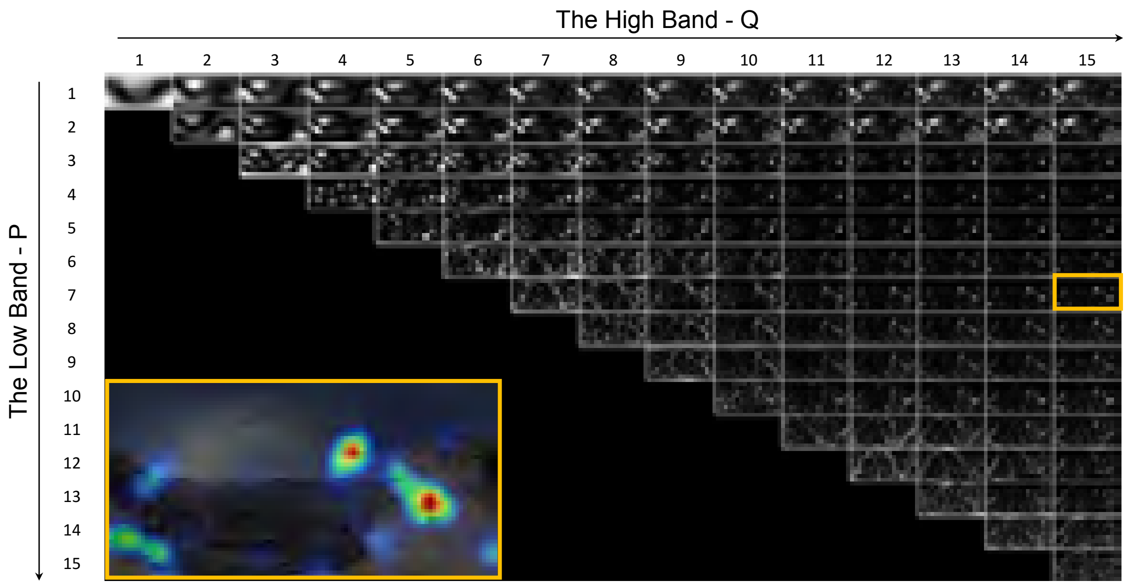


Figure 3.4: The spectral residual maps between different bands of spherical harmonics. The number along the horizontal axis indicates the high band Q, while the vertical axis indicates the low band P. Note that the saliency maps within or close to the orange bounding box successfully detect the two people in the frame.

We show the SSR results of the intensity channel with all different pairs of the lower band P and the higher band Q in Figure 3.4. As P increases, the low-frequency information such as the sky and mountains are filtered out. The spectral residual results within and close to the orange bounding box reveal the salient objects, such as the two people.

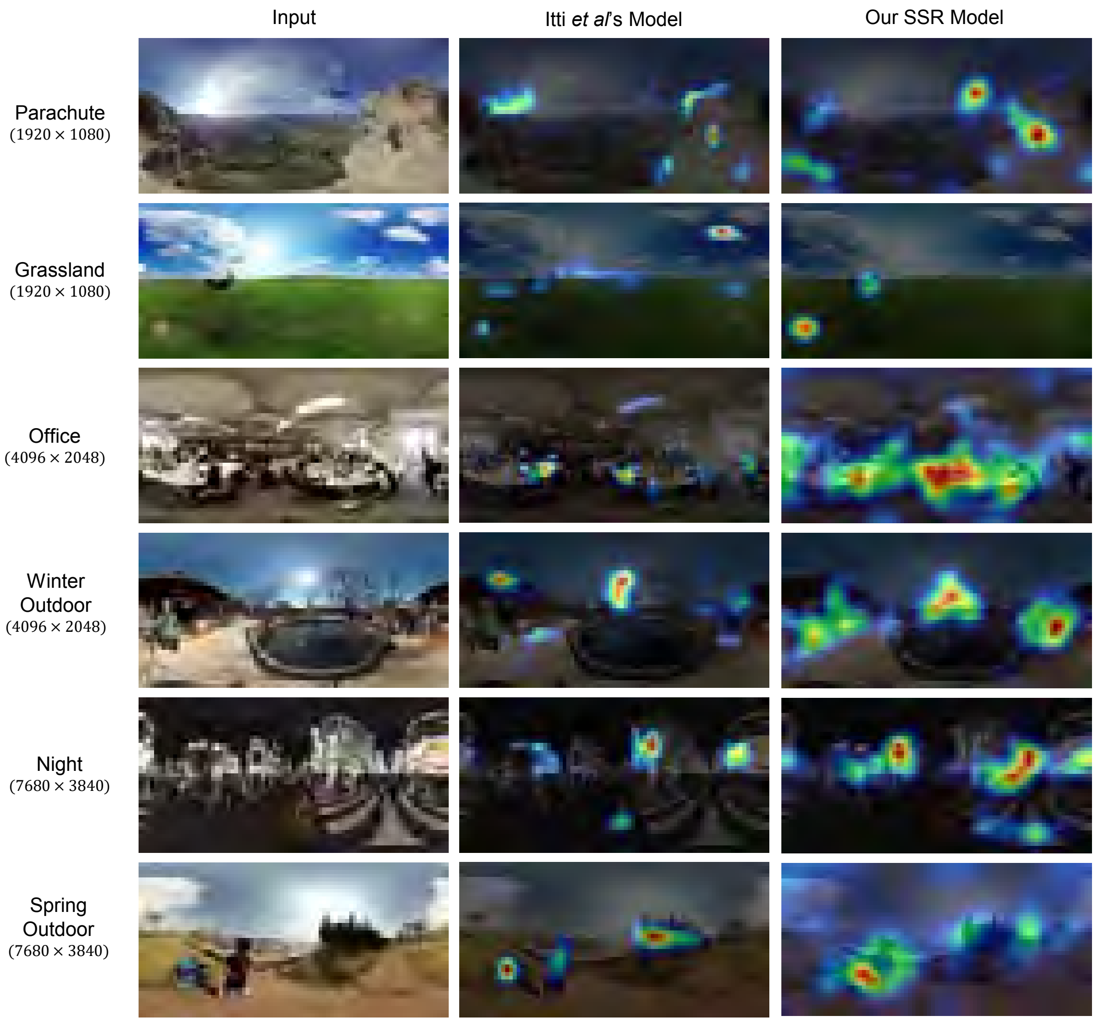


Figure 3.5: The visual comparison between the Itti *et al.*'s model and our SSR model. Note that while the results are visually similar, our SSR model are 5× to 13× times faster than the Itti *et al.*'s model.

3.4.2 Temporal Saliency

In addition to intensity and color features, we further extract temporal saliency in the spherical harmonics domain.

For the SH coefficients extracted from the three feature maps, we maintain two sliding temporal windows to estimate the temporal contrast. The smaller window w_0 stores the more recent SH coefficients from the feature maps, and the larger window w_1 stores the SH coefficients over a longer term. For each frame, we calculate the estimated SH coefficients $\bar{c}_l^m, \bar{\bar{c}}_l^m$ from both windows, using two probability density functions from the Gaussian distribution ($|w_0| = 5, |w_1| = 25, \sigma = 7.0$). We use formulation similar to the spatial saliency to measure the spherical spectral residual between two temporal windows:

$$\mathfrak{R}(F_{\text{temporal}}, \theta, \phi) = \left| \sum_{l=P+1}^{Q} \sum_{m=-l}^{l} \left(\bar{\bar{c}}_l^m(\theta, \phi) - \bar{c}_l^m(\theta, \phi) \right) \cdot Y_l^m(\theta, \phi) \right| \tag{3.14}$$

We further apply Equation 3.13 to compute the smoothed temporal saliency maps.

3.4.3 Saliency Maps with Nonlinear Normalization

Following Itti *et al.*[61], we apply the non-linear normalization operator $\mathscr{N}(\cdot)$ to all the six saliency maps: intensity, red-green, and blue-yellow contrasts, both statically and temporally. This operator globally promotes maps which contain a few peak responses and suppresses maps with a large number of peaks.

$$\mathcal{S} = \frac{1}{N} \bigoplus_{i=1}^{N} \mathscr{N}\left(\boldsymbol{S}(F_i)\right) \tag{3.15}$$

After the non-linear normalization, we linearly combine all saliency maps into the final saliency map. Empirically, we choose $Q = 15, P = 7$. The final composed result is shown at the bottom left corner in Figure 3.4, as well as in the accompanying video.

3.4.4 Comparison Between the Itti *et al.*'s and SSR Model

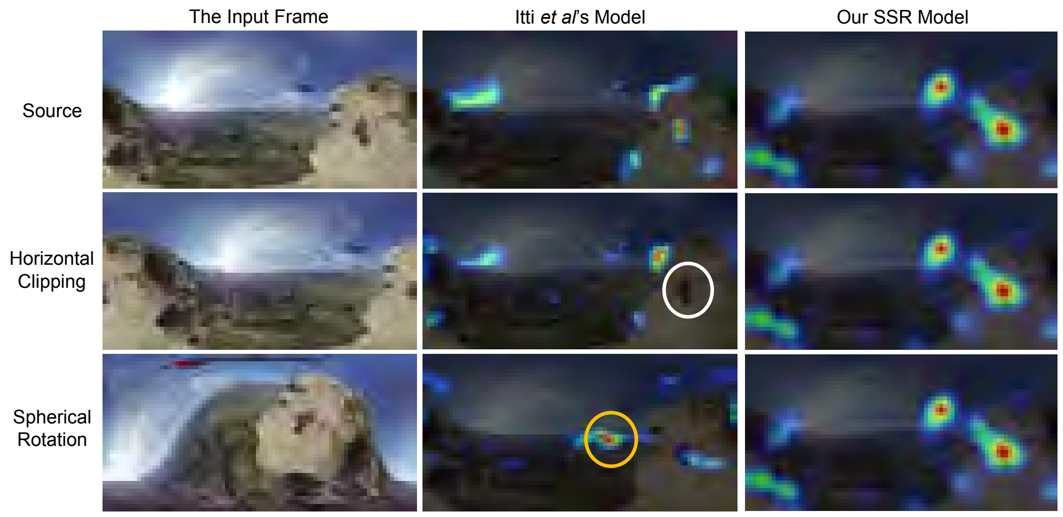


Figure 3.6: This figure shows the comparison between Itti *et al.*'s model and our SSR model with horizontal translation and spherical rotation in the 360° video frame. White circles indicate the false negative result from Saliency Toolbox and orange ones indicate false positive result from Saliency Toolbox. Meanwhile, the results from our SSR model remain consistent, regardless of horizontal clipping and spherical rotation. Note that we use custom shaders to transform the spherical images, compute the saliency, and apply the inverse transformation for intuitive visualization.

As shown in Figure 3.6, our SSR model is visually better than Itti *et al.*'s model. In addition, our experimental results below compare the classic Itti *et al.*'s model and our model.

We use six videos from the Insta360[2] and the 360Rize[3]. The video resolutions vary from 1920×1080 to 7680×3840 pixels.

The experiments are conducted on a workstation with an NVIDIA GTX 1080 and an Intel Xeon E5-2667 2.90GHz CPU with 32 GB RAM. Both Itti *et al.*'s model

[2]Insta360: `https://www.insta360.com`

[3]360 Rize: `http://www.360rize.com`

Resolution	The Average Timing Per Frame		
	Itti *et al.* (CPU)	SSR (CPU)	SSR (GPU)
1920x1080	104.46 ms	21.34 ms	10.81 ms
4096x2048	314.94 ms	48.18 ms	13.20 ms
7680x3840	934.26 ms	69.53 ms	26.58 ms

Table 3.2: Timing comparison between the Itti *et al.*'s model and our SSR model

and the SSR model are implemented in C++ and OpenCV. The GPU versions of the SSR model is developed using CUDA 8.0. We measure the average timing of saliency computation, as well as the visual results between the Itti *et al.*'s model and our SSR model. Note that the timings do not include the uploading time for each frame from system memory to GPU memory. We believe that our algorithms would map well to products such as NVIDIA DrivePX[4] in which videos are directly loaded onto the GPU memory.

We measure the average computational cost of the initial 600 frames across three resolutions: 1920×1080, 4096×2048, and 7680×3840, as shown in Table 3.2. All frames are preloaded into the CPU memory to eliminate the I/O overhead. Both the CPU and GPU versions of our SSR model outperform the classic Itti *et al.*'s model, with the speedups ranging from 4.8× to 13.4×, depending on various resolutions. We show the example input, and the output from both models in Figure 3.5.

[4]`https://NVIDIA.com/en-us/self-driving-cars/drive-px`

3.5 Saliency-guided Virtual Cinematography

With advances in the 360° video cameras and network bandwidth, more events are being live-streamed as high-resolution 360° videos. Nevertheless, while the user is watching the 360° video in a typical commodity HMD, almost 90 percent of the video is beyond the user's field of view, as shown in Table 3.1. Therefore, methods to automatically control the path of the virtual camera (virtual cinematography) becomes a vital challenge for streaming and navigating 360° videos in real time. Inspired by the prior work on camera path selection and interpolation [96, 99, 132, 133, 134, 135, 136, 137, 138], we investigate how saliency maps could guide automatic camera control for 360° videos.

First, we compute the saliency maps by linearly combining the saliency maps based on intensity, color, and motion, and then perform a non-linear normalization, as introduced in the previous section.

However, for 360° videos, the most salient objects may vary from frame to frame, due to the varying occlusions, colors, and self-movement. As a result, an approach that relies on just tracking the most salient objects may incur rapid motion of the camera, and worse still, may induce motion sickness in virtual reality.

In this section, we propose a spatiotemporal optimization model of the virtual camera's discrete control points and further employ a spline interpolation amongst the control points to achieve smooth camera navigation.

3.5.1 Optimization of the Virtual Camera's Control Points

To estimate the virtual camera's control points, we formulate an energy function $\boldsymbol{E}(C)$ in terms of camera location $C = (\theta, \phi)$. The energy function

$$\boldsymbol{E}(C) = \lambda_{\text{saliency}} \cdot \boldsymbol{E}_{\text{saliency}}(C) + \lambda_{\text{temporal}} \cdot \boldsymbol{E}_{\text{temporal}}(C) \tag{3.16}$$

consists of a saliency coverage term $\boldsymbol{E}_{\text{saliency}}$ and a temporal motion term $\boldsymbol{E}_{\text{temporal}}$, thus taking both saliency coverage and temporal smoothness into consideration.

3.5.2 Saliency Coverage Term

This spatial term $\boldsymbol{E}_{\text{saliency}}$ penalizes the coverage of the saliency values beyond the field of view. As for a specific virtual camera location C, this term would be written as:

$$\boldsymbol{E}_{\text{saliency}}(C) = \frac{\sum_{\theta,\phi} \boldsymbol{S}(\theta,\phi) \cdot \boldsymbol{O}(C,\theta,\phi)}{\sum_{\theta,\phi} \boldsymbol{S}(\theta,\phi)} \tag{3.17}$$

where $\boldsymbol{O}(C,\phi,\theta)$ indicates whether an arbitrary spherical point (ϕ,θ) is observed by the camera centered at the location C_i:

$$\boldsymbol{O}(C,\theta,\phi) = \begin{cases} 1 & , (\theta,\phi) \text{ is observed by virtual camera at } C \\ 0 & , \text{otherwise} \end{cases} \tag{3.18}$$

Thus, $\boldsymbol{E}_{\text{saliency}}(C)$ measures the coverage of the saliency values beyond the field of view of the virtual camera centered at C. To reduce the computation, we compute the saliency coverage terms over 2048 points (θ, ϕ), that are uniformly distributed over the sphere.

3.5.3 Temporal Motion Term

For the i^{th} frame in the sequence of the discrete control points, $\boldsymbol{E}_{\text{temporal}}(C)$ measures the temporal motion of the virtual camera as follows:

$$\boldsymbol{E}_{\text{temporal}}(C) = \begin{cases} \|C_{i-1}, C_i\|_2 & , i \geq 1 \\ 0 & , i = 0 \end{cases} \tag{3.19}$$

3.5.4 The Optimization Process

Based on this spatiotemporal model, we evaluate the energy functions over 32×64 pairs of discrete (θ, ϕ). This process is highly parallel and can be efficiently implemented on the GPU. For each frame, we compute the optimal camera point as follows:

$$\mathring{C} = \underset{C}{\operatorname{argmin}}\, \boldsymbol{E}(C) \tag{3.20}$$

In this way, we extract a sub-sequence of discrete spherical coordinates $Seq = \{\mathring{C}_i | \mathring{C}_i = (\phi_i, \theta_i)\}$ of the optimal camera location in the saliency maps every K frames, $K = 5$ in our examples. Since these locations are discrete and sampled at a lower frame rate, we further perform spline interpolation with C^2 continuity.

3.5.5 Interpolation of Quaternions

To achieve superior interpolation over a sphere, we convert the spherical coordinates to quaternions:

$$Q(\theta,\phi) = (0, \sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)) \tag{3.21}$$

We use the spherical spline curves with C^2 continuity to compute the smooth trajectory of the camera cruise path over the quaternions. For an arbitrary timestamp x, we need to compute the interpolated spherical coordinates $\mathscr{Q}_i(x)$. We denote t_i as the most recent timestamp to x, which corresponds to the i^{th} video frame, and Q_i as the corresponding quaternion. Hence, we compute the interpolated quaternion $\mathscr{Q}_i(x)$ as follows:

$$\begin{aligned}\mathscr{Q}_i(x) = {} & \frac{\nabla^2 Q_i (x - t_{i-1})^3}{6h_i} + \frac{\nabla^2 Q_{i-1} (t_i - x)^3}{6h_i} \\ & + \left[\frac{Q_i}{h_i} - \frac{\nabla^2 Q_i h_i}{6}\right](x - t_{i-1}) + \left[\frac{Q_{i-1}}{h_i} - \frac{\nabla^2 Q_{i-1} h_i}{6}\right](t_i - x)\end{aligned} \tag{3.22}$$

where $\nabla^2 Q_i$ is the second derivative of the quaternion at the i_{th} frame, and $h_i = t_i - t_{i-1}$. Figure 3.7 shows the locations of the global maximas, as well as the interpolated spline path over the sphere.

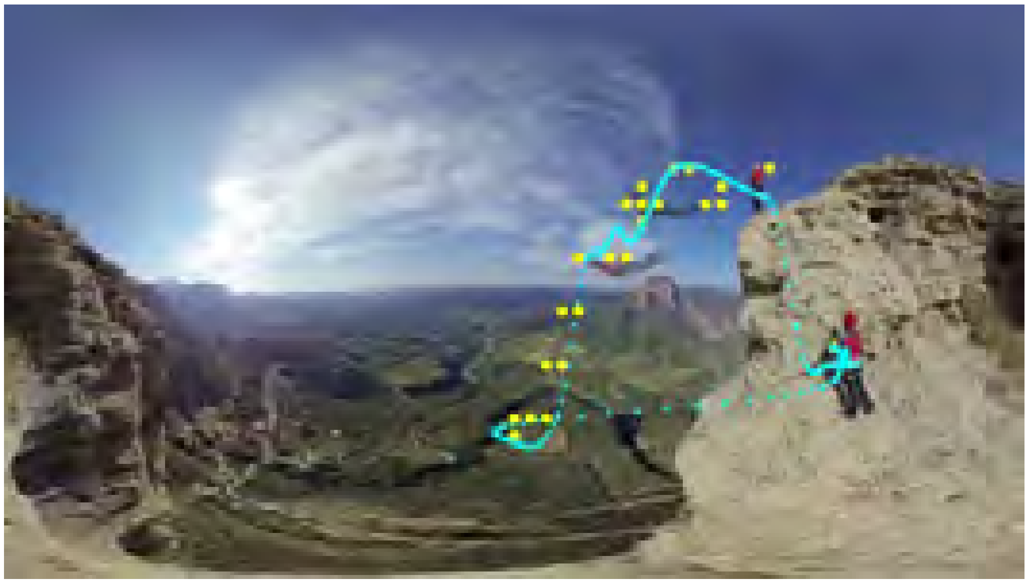


Figure 3.7: This figure shows the interpolation amongst the global maximums of the saliency maps in the spherical space. The yellow dots show the discrete optimal location using the energy function, and the blue dots show the interpolation using the spherical spline curve with C^2 continuity.

3.5.6 Evaluation of the SpatioTemporal Optimization Model

We compare our method with the MaxCoverage model which determines the camera position for the maximal coverage of the saliency map. We evaluate the temporal motion terms for the same video sequence and plot the data in Figure 3.8.

From the quantitative evaluation, as well as the complementary video, we have validated that the SpatioTemporal Optimization model reduces the temporal jittering of the camera motion compared to MaxCoverage model for virtual cinematography in 360°videos.

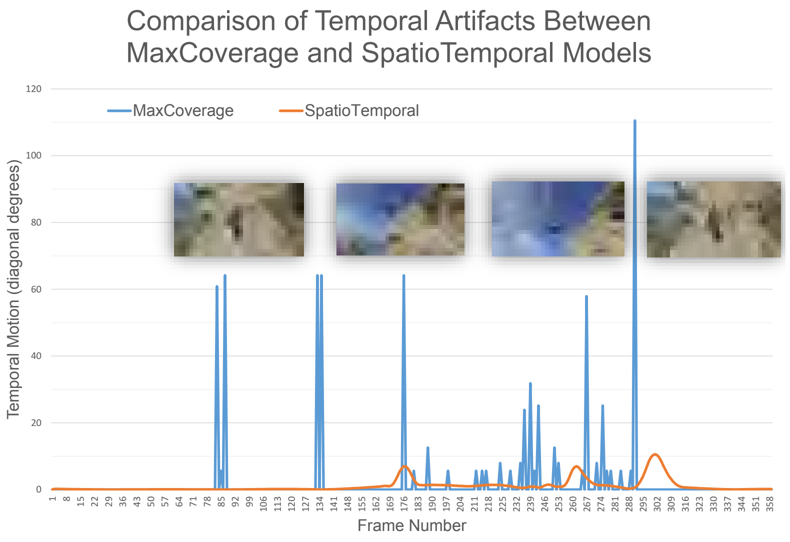


Figure 3.8: Quantitative comparison between the MaxCoverage model and the SpatioTemporal Optimization model. We visualize the temporal motion of the virtual camera across 360 frames. Compared with the MaxCoverage model, the SpatioTemporal Optimization model significantly reduces the temporal jitters.

3.6 Conclusions and Future Work

In this chapter, we have presented a novel GPU-driven pipeline which employs spherical harmonics to directly compute the saliency maps for 360° videos in the $\mathbb{SO}(2)$ space. In contrast to the traditional method, our method remains consistent for challenging cases like horizontal clipping, spherical rotations, and equatorial bias, and is 5× to 13× faster than the classic Itti *et al.*'s model.

We demonstrate the application of using spherical harmonics saliency to automatically control the path of the virtual camera. We present a novel spatiotemporal optimization model to maximize the spatial saliency coverage and minimize the temporal jitters of the camera motion.

In future, we plan to further develop our SSR model for stereoscopic saliency detection [139] in 360° videos. We aim to collect a large scale dataset with stereo 360° videos and human eye tracking data. Another future direction is to generate hyper-lapse rectilinear videos [140] from 360° videos, using a variant of our virtual cinematography model.

We would like to open source our toolbox for computing spherical harmonics from 360° videos, saliency maps from SSR models, and virtual cinematography. We believe the spherical representation of saliency maps will inspire more research to think out of the rectilinear space. We envision our techniques will be widely be used for live streaming of events, video surveillance of public areas, as well as templates for directing the camera path for immersive storytelling. Future research may explore how to naturally place 3D objects with spherical harmonics irradiance in 360° videos, how to employ spherical harmonics for foveated rendering in 360° videos, and the potential of compressing and streaming 360° videos with spherical harmonics.

Chapter 4: Video Fields: Fusing Multiple Surveillance Videos into a Dynamic Virtual Environment

4.1 Introduction

Surveillance videos play a crucial role in monitoring a variety of activities in shopping centers, airports, train stations, and university campuses. In conventional surveillance interfaces, where multiple cameras are depicted on a display grid, human operators endure a high cognitive burden in fusing and interpreting multiple camera views (Figure 4.2). In the field of computer vision, researchers have made great strides in processing surveillance videos for segmenting people, tracking moving entities, as well as classifying human activities. Nevertheless, it remains a challenging task to fuse multiple videos from RGB cameras, without prior depth information, into a dynamic 3D virtual environment. Microsoft Kinect, for example, cannot capture the infrared structured lighting patterns in the sunlight. In this project, we present our research on the fusion of multiple video streams taking advantage of the latest WebGL and WebVR technology.

Nowadays, there is an increasing demand for real-time photo-realistic dynamic scene generation. As an exploratory work, we investigate the following research

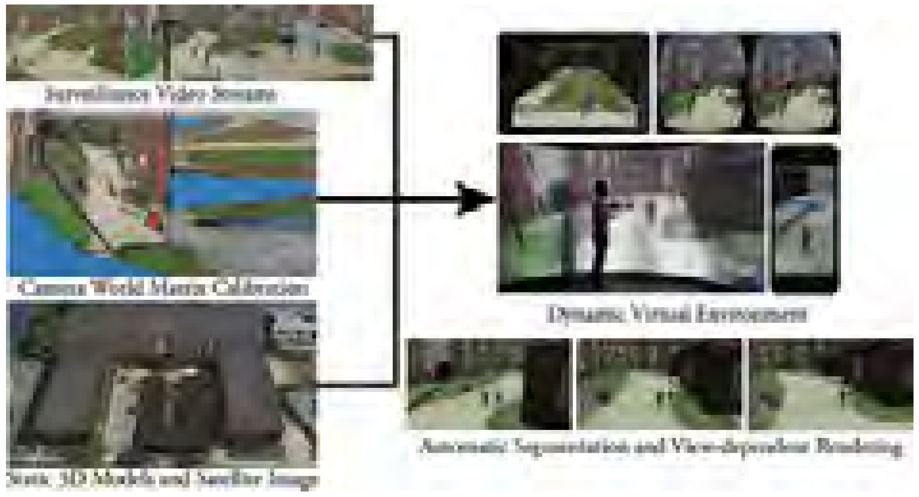

Surveillance Video Streams
Camera World Matrix Calibration
Static 3D Models and Satellite Image
Dynamic Virtual Environment
Automatic Segmentation and View-dependent Rendering

Figure 4.1: *Video Fields* system fuses multiple videos, camera-world matrices from a calibration interface, static 3D models, as well as satellite imagery into a novel dynamic virtual environment. *Video Fields* integrates automatic segmentation of moving entities during the rendering pass and achieves view-dependent rendering in two ways: early pruning and deferred pruning. *Video Fields* takes advantage of the WebGL and WebVR technology to achieve cross-platform compatibility across smart phones, tablets, desktops, high-resolution tiled curved displays, as well as virtual reality head-mounted displays. See the supplementary video at `http://videofields.com`

questions:

1. Can we efficiently generate dynamic scenes from surveillance videos for VR applications?
2. Can we use a web-based interface to allow human operators to calibrate the cameras intuitively?
3. Can we use the-state-of-the-art web technologies to achieve interactive video-based rendering?
4. Can we render moving entities as 3D objects in virtual environments?

In the *Video Fields* project, we present our early solutions to create dynamic

Figure 4.2: This photograph shows conventional surveillance interface where multiple monitors are placed in front of the operators. One of the greatest challenges for the users is to mentally fuse and interpret moving entities from multiple camera views.

virtual reality from surveillance videos using web technologies. The contributions are summarized as follows:

1. conception, architecture, and implementation of *Video Fields*, a mixed-reality system that fuses multiple surveillance videos into an immersive virtual environment,
2. integrating automatic segmentation of moving entities in the *Video Fields* rendering system,
3. presenting two novel methods to fuse multiple videos into a dynamic virtual environment: *early pruning* and *deferred pruning* of geometries,
4. achieving cross-platform compatibility across a range of clients including smartphones, tablets, desktops, high-resolution large-area wide field-of-view tiled display walls, as well as head-mounted displays.

4.2 System Design

Video Fields system consists of a camera world calibration interface, a back-end server to process and stream the videos, and a web-based rendering system. The flow chart of the system is shown in Figure. 4.3.

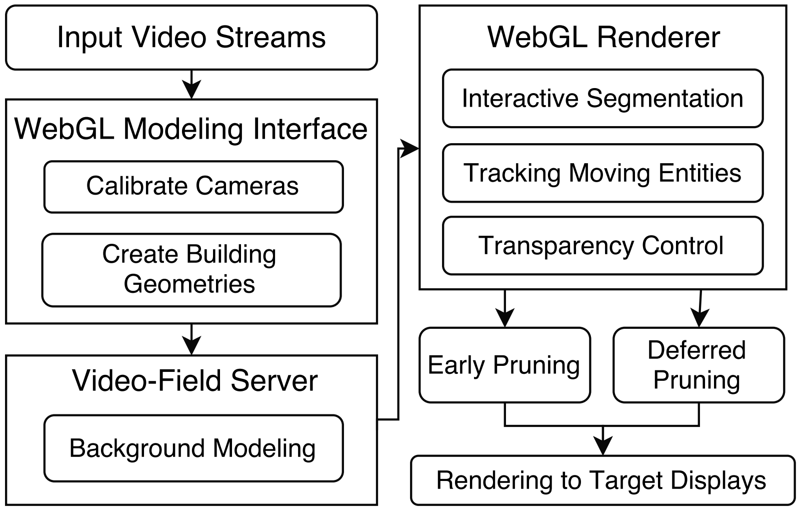


Figure 4.3: This figure shows the workflow of Video Fields. Our system imports video streams as video textures in WebGL. In the WebGL camera world calibration interface, the user may create the ground, calibrate the cameras, and add initial geometries. The videos are then sent to the Video Fields server for generating a background model.

4.2.1 Camera-World Calibration Interface

The camera-world calibration interface for Video Fields has been built based on WebGL and WebVR. We use the open-source library `Three.js`[1] to create the interface and render the mixed-reality scene. We have designed a four-stage workflow for Video Fields. First, users import their videos into the system. During this stage, the users can also alter the video timeline to crop and synchronize the videos

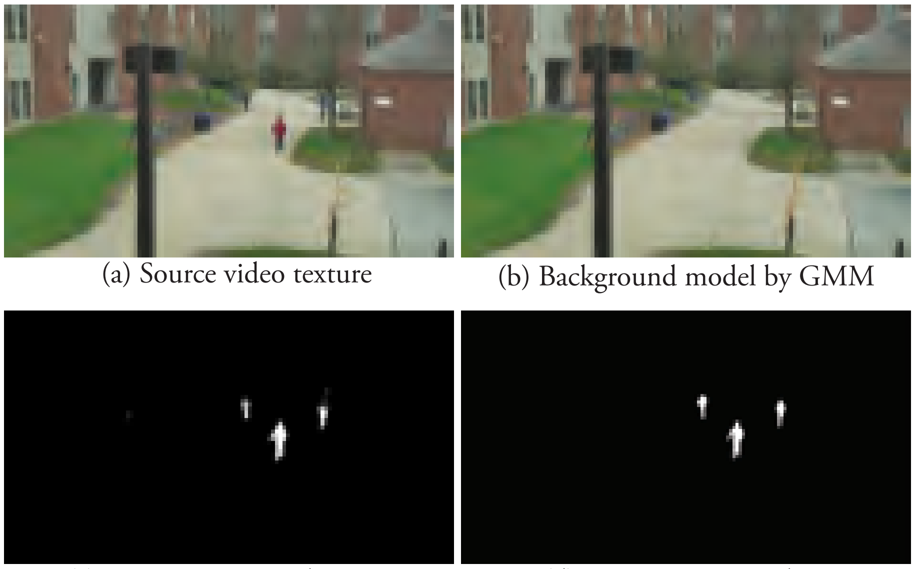
[1]Three.js: `http://threejs.org/`

manually, if needed. Then users can define the ground projection plane for the projection. Second, users adjust the position and the rotating quaternion for each camera in the videos as if they were orienting flashlights onto the ground projection plane. We have used the transform controller in the `Three.js` library to provide such an interface. Third, users can drag geometric primitives for constructing buildings for the 3D world. The user can watch real-time rendering results interactively as they are dragging the geometric primitives. Finally, users can use any number of display devices, including the VR headsets, to experience the world that they have just created. Using the estimated position and size of each geometry, we can attach textures to the user-defined building geometry. We can also use a sky sphere to enhance the visual immersion. The position, scaling, and rotation of the cameras (as camera world matrices $\boldsymbol{C}$) and the 3D models are exported and saved in the JSON format.

4.2.2 Background Modeling

The motivation of background modeling is to provide a background texture for each camera to identify moving entities in the rendering stage and to reduce the network bandwidth requirements when streaming videos from the web-server. With a robust estimation of the background, only the pixels corresponding to the moving entities will need to be streamed for rendering and not the rest of the video for every frame.

To estimate robust background images, we take advantage of Gaussian Mixture



(a) Source video texture

(b) Background model by GMM

(c) Segmentation without Gaussian convolution

(d) Segmentation with Gaussian convolution

Figure 4.4: Segmentation results with Gaussian mixture models of the background. (a) shows a reference frame of the video texture, (b) shows the background model learnt by the Gaussian Mixture Models approach, (c) shows segmentation without Gaussian convolution, and (d) shows the segmentation with Gaussian convolution

Models (GMM) for background modeling. Compared with the mean filter or the Kalman filter, GMM is more adaptive with different lighting conditions, repetitive motions of scene elements, as well as moving entities in slow motion [141].

Given the video texture $\boldsymbol{T}$, for each pixel $\boldsymbol{t}_{uv}$ at the texture coordinates (u, v), we model the background value of the pixels as a mixture of N Gaussian distributions $\mathscr{P}_{uv}$. For each frame at time $i \in \{1, \ldots, M\}$, we denote $\boldsymbol{T}(u,v)_i$ as the set of all previous pixels at $\boldsymbol{t}_{uv}$, $\mathscr{P}(\boldsymbol{T}(u,v)_i)$ as the probability of the background color of the pixel at (u, v):

$$\begin{aligned}\boldsymbol{T}(u,v)_i &= \{\boldsymbol{T}(u,v,j), 1 \le j \le i\} \\ \mathscr{P}(\boldsymbol{T}(u,v)_i) &= \sum_{j=1}^{N} \mathscr{N}(\boldsymbol{T}(u,v)_i | \mu_{ij}, \Sigma_{ij}) \cdot \omega_{ij}\end{aligned} \tag{4.1}$$

where N is the total number of Gaussian distributions. We set $N \leftarrow 3$ in our implementation and $\mathscr{N}$ is the Gaussian probability density function:

$$\mathscr{N}(\boldsymbol{T}(u,v)_i | \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}} \cdot \frac{1}{\Sigma^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{T}(u,v)_i - \mu_{ij})^T \sigma_{ij}^{-1} (\boldsymbol{T}(u,v)_i - \mu_{ij})} \tag{4.2}$$

where Σ is the covariance matrix. We assume that the red, green, and blue channels are independent and have the same variances. Thus, the GMM can be trained at a lower cost of computing the covariance matrix. To update the Gaussian distributions, we used an online K-means algorithm to learn the weights ω_{ij}:

$$\omega_{ij} \leftarrow (1 - \alpha)\omega_{i(j-1)} + \alpha \mathcal{M}_{ij} \tag{4.3}$$

where α is the learning rate and $\mathcal{M}_{ij}$ indicates whether the model is matched to the current pixel. An example of the computed background model is shown in Figure 4.4(b).

4.2.3 Segmentation of Moving Entities

After learning the background model for each video, we achieve real-time interactive segmentation of moving entities during the fragment-shader rendering pass by taking advantage of the many-core computing on the GPU. To alleviate noise and smooth the boundaries, given input video texture $\boldsymbol{T}$ and its corresponding background model $\boldsymbol{B}$, we convolve $\boldsymbol{T}$ and $\boldsymbol{B}$ with a Gaussian kernel $\mathscr{G}$ at scale σ:

$$\boldsymbol{T}' \leftarrow \mathscr{G}(\sigma) \otimes \boldsymbol{T}, \boldsymbol{B}' \leftarrow \mathscr{G}(\sigma) \otimes \boldsymbol{B} \tag{4.4}$$

We segment the moving entities by the thresholding function δ:

$$\boldsymbol{F} \leftarrow \delta(|\boldsymbol{I}' - \boldsymbol{B}'|) \tag{4.5}$$

A comparative example showing the advantage of using the Gaussian convolution is shown in Figure 4.4 (c) and (d) using the same threshold of 0.08. This parameter is passed to the fragment shader as a GLSL floating-point number, thus enabling the user to interactively alter the thresholding function in the WebGL browser. After obtaining the foreground $\boldsymbol{F}$, we also calculate the set of bounding rectangles $\boldsymbol{R}$ of moving entities in $\boldsymbol{F}$.

4.3 Video Fields Mapping

To map video fields onto the geometries in the 3D virtual environment, we need to establish a bidirectional mapping between the texture space and the 3D world space. Here, we use the ground model as an example of an arbitrary geometry in the 3D scene. The challenges we address here are as follows:

- Given a vertex on the ground model, we need to calculate the corresponding

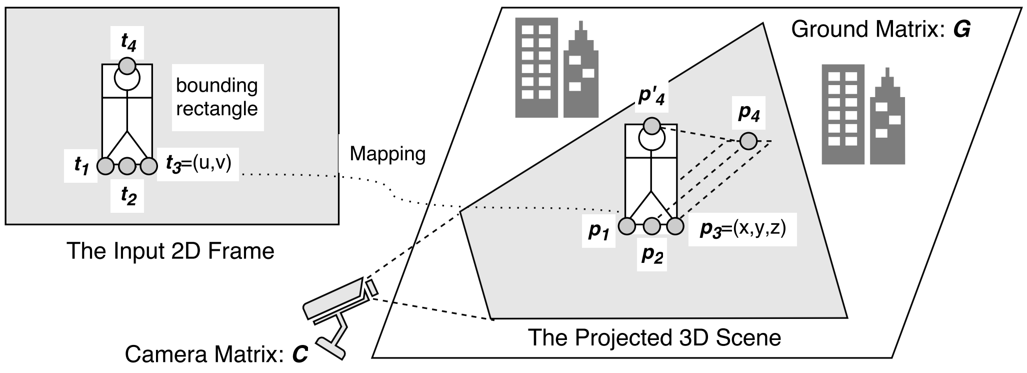


Figure 4.5: This figure shows the the overview of the *Video Fields* mapping by projecting 2D imagery from the streaming videos onto a 3D scene; $t_1, \ldots, t_4$ indicate the key points of the foreground bounding boxes in the input 2D frames and $p_1, \ldots, p_4$ indicate the corresponding projected points.

pixel in the texture space. This is important for visualizing the color of the ground model.

- Given a pixel in the texture space, we need to calculate the corresponding vertex on the ground to project that pixel. This is important for projecting a 2D segmentation of a moving entity to the 3D world.

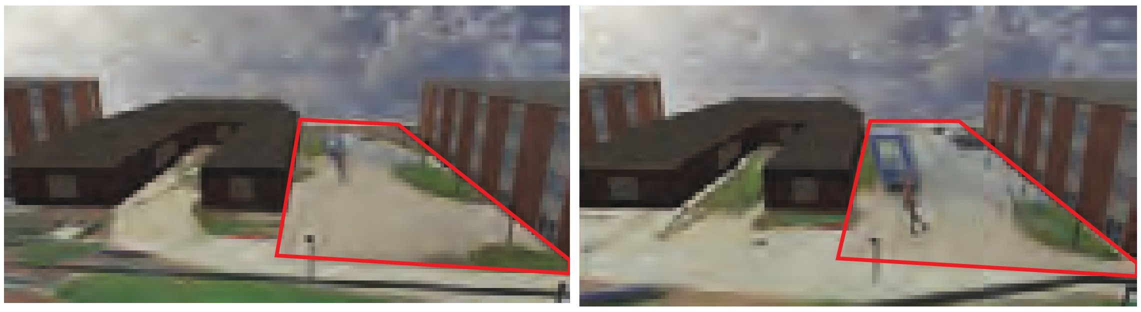
The first challenge could be solved by projection mapping, but we need to correct for the inherent perspective in the acquired video. Given the camera-world matrix $\boldsymbol{C}$, which is obtained from the WebGL-based camera-world calibration interface, and the model matrix of the ground $\boldsymbol{G}$. For each vertex $\boldsymbol{p}_{xyz}$ on the ground, we first use the camera-world matrix and the ground-model matrix to convert its coordinates to the homogeneous coordinates in the camera space:

$$\hat{\boldsymbol{p}}_{xyzw} \leftarrow \boldsymbol{C} \cdot \boldsymbol{G} \cdot (\boldsymbol{p}_{xyz}, 1.0) \tag{4.6}$$

To obtain the texture coordinates $\boldsymbol{t}_{uv}$, we need to carry out a perspective correction:

$$\boldsymbol{t}_{uv} \leftarrow \left(\frac{\hat{\boldsymbol{p}}_x + \hat{\boldsymbol{p}}_w}{2\hat{\boldsymbol{p}}_w}, \frac{\hat{\boldsymbol{p}}_y + \hat{\boldsymbol{p}}_w}{2\hat{\boldsymbol{p}}_w} \right) \tag{4.7}$$

Figure 4.6 shows the results before and after the perspective correction.



(a) Video Fields mapping before perspective correction (b) Video Fields mapping after perspective correction

Figure 4.6: This figure shows the results before and after the perspective correction. The texture in (a) has the same perspective as the original video, but is not projected correctly in the 3D scene.

The second challenge, to convert a 2D point to 3D, is non-trivial because projection from 3D to 2D is irreversible. To do this, we first compute a dense grid contained within the ground projection of the video. For each 3D vertex of this grid, we compute the corresponding 2D coordinates in the video texture using equations 4.4 and 4.5. We store the results in a hash function $\mathscr{H}$. Since we are using a dense grid, all points in the video texture can be mapped to a 3D vertex on the ground. Finally, we store the results in a hash function:

$$\mathscr{H} : \boldsymbol{t}_{uv} \longmapsto \boldsymbol{p}_{xyz}. \tag{4.8}$$

Once calculated, this hash map can be stored on the server side and be used for mapping 2D texture points back to the 3D world. For each vertex $\boldsymbol{p}$ on the ground, we also calculate the angle between the camera ray and the ground surface $\theta_{\boldsymbol{p}}$.

4.3.1 Early Pruning for Rendering Moving Entities

To render moving entities in the 3D world, we remove the background pixels from the video texture and correct the projection so that the moving entities are vertical on the ground model. In the *Early Pruning* approach, we discard the pixels that do not belong to the foreground as soon as the foreground is identified after the Gaussian convolution and thresholding by equations 4.4 and 4.5. Then the foreground pixels are transformed into a 3D point cloud and projected to the 3D world space. We use point clouds to implement the early pruning technique and optimize the rendering performance of *Video Fields* since they are extremely efficient to render. The detailed algorithm is described in Algorithm 3. View-dependent results of our visualization technique are shown in Figure 4.4.

4.3.2 Deferred Pruning for Rendering Moving Entities

Though pruning the background pixels at an early stage is useful for our videos where most pixels belong to the background, we have also developed the deferred pruning approach for better anti-aliasing, bi-linear sampling, and faster visibility testing. In this approach, we dynamically project videos on moving billboards. The background subtraction is completed in the fragment shader of each billboard. After

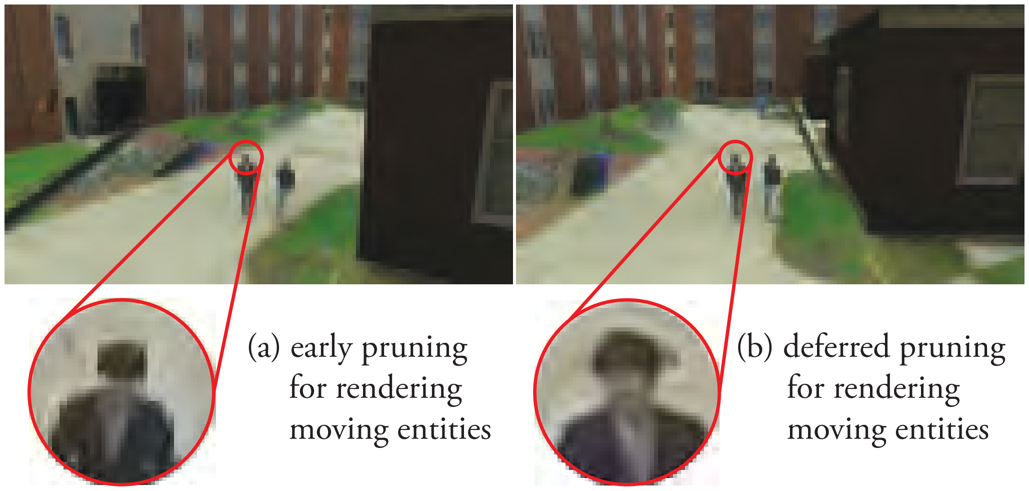
ALGORITHM 3: Early Pruning for Rendering Moving Entities

```
Input: foreground $\boldsymbol{F}$ and the set of bounding rectangles $R$ of moving entities
Output: a 3D point cloud $P$ visualizing the moving entities
Initialize a set of points for the video visualization. (Run once);
For each pixel $\boldsymbol{t}$ inside the bounding box, calculate the intersection point $\boldsymbol{t}_\perp$
  between its perpendicular line and $\boldsymbol{t}_1\boldsymbol{t}_3$;
for each pixel $\boldsymbol{t}$ from the video do
    if $\boldsymbol{t} \notin \boldsymbol{F}$ then
        discard $\boldsymbol{t}$ and continue;
    end
    set the color of the pixel: $c \leftarrow \textbf{texture2D}(\boldsymbol{F}, \boldsymbol{t})$;
    look up the corresponding projected points in the 3D scene:
      $\boldsymbol{p} \leftarrow \mathscr{H}(\boldsymbol{t}), \boldsymbol{p}_\perp = \mathscr{H}(\boldsymbol{t}_\perp)$;
    update the $z$ coordinate of the 3D point: $\boldsymbol{p}_z \leftarrow |\boldsymbol{p} - \boldsymbol{p}_\perp| \cdot \tan(\theta_{\boldsymbol{p}})$ ;
    use the $x, y$ coordinates of $\boldsymbol{t}_\perp$ to place the point vertically: $\boldsymbol{p}_{xy} \leftarrow \boldsymbol{t}_{uv}$ ;
    render the point $\boldsymbol{p}$;
end
```

the world matrix of a billboard is determined, we test the visibility of the billboard. We then render foreground pixels onto the billboard and discard the background pixels. We describe the details in Algorithm 4.

4.3.3 Visibility Testing and Opacity Modulation

One of the biggest advantages for visualizing videos in an immersive 3D virtual environment is that the system can adjust the opacity of every object, thus allowing the users to "see-through buildings". The users can therefore observe the video-recorded activities from cameras that would otherwise not be viewable from a user's given vantage point. We achieve this by doing a visibility test using ray-casting from the current camera to every moving entity in the scene. If a moving entity is found to be occluded, we can modulate the opacity of that occluding object and render the moving entities. Figure 4.8 shows an example before and after the visibility test

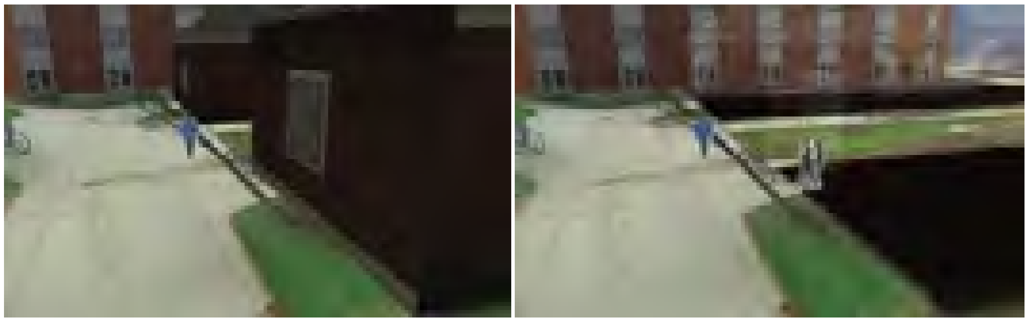


(a) early pruning for rendering moving entities
(b) deferred pruning for rendering moving entities

Figure 4.7: This figure shows the segmentation of moving entities, view-dependent rendering, as well as zoom-in comparison between the early pruning algorithm and the deferred pruning algorithm.

ALGORITHM 4: Deferred Pruning for Rendering Moving Entities

```
Input: foreground $\boldsymbol{F}$ and the set of bounding rectangles $\boldsymbol{R}$ of moving entities
Output: a set of billboards rendering the moving entities
Initialize a set of billboards to display moving objects. (Run once);
for each detected bounding box $\boldsymbol{r}$ in $\boldsymbol{R}$ do
    calculate the bottom-left, bottom-middle, bottom-right and top-middle
      points $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4$ in $\boldsymbol{r}$, as illustrated in Figure 4.5;
    look up the corresponding projected points in the 3D scene:
      $\boldsymbol{p}_i \leftarrow \mathscr{H}(\boldsymbol{t}_i), i \in \{1,2,3,4\}$.;
    calculate the width of the billboard in the 3D space:
      $w \leftarrow |\boldsymbol{p}_3 - \boldsymbol{p}_1|, h \leftarrow |\boldsymbol{p}_4 - \boldsymbol{p}_2| \cdot \tan(\theta_{\boldsymbol{p}_4})$.;
    Reposition a billboard to the position $\frac{\boldsymbol{p}_1+\boldsymbol{p}_3}{2}$ with width and height $w$ and
      $h$;
    In the fragment shader of the billboard, sample the color from $I$ as
      described in Equation. 4.6 and 4.7, but replace $G$ with the current
      billboard's model matrix; discard pixels which does not belong to the
      foreground $F$;
end
```


(a) rendering without visibility test and transparency control

(b) rendering with visibility test and transparency control

Figure 4.8: This figure shows the rendering results before and after the visibility test and opacity adjustment. Note that the two people behind the building are correctly rendered through the semi-transparent meshes.

and opacity modulation.

4.4 Experiments and Evaluation

In our experiment, we recorded three 10-minute video clips with a resolution of 1280×720 pixels. We tested both the early- and deferred-pruning algorithms in the following three settings. The first two tests were conducted on a desktop workstation with an NVIDIA Quadro K6000 graphics card running Windows 8.1 on Google Chromium 48.0.2544.0 with WebVR enabled. We tested on a regular desktop display with a resolution of 2560×1440 as well as Oculus Rift DK2 head-mounted display with a resolution of 950×1080 for each eye. The last test was conducted in an immersive curved screen environment with 15 projectors driven by 4 NVIDIA Quadro K6000 graphics cards. The rendering resolution was 6000×3000 pixels. The results were rendered with the same software setup. The experimental results are shown in Table 4.1:

Table 4.1: The experimental results of early pruning and deferred pruning in *Video Fields* for different display devices and resolutions

Render Algorithm	Resolution	WebVR	Framerate (FPS)
Early Pruning	2560×1440	No	60.0
	$2 \times 960 \times 1080$	Yes	55.2
	6000×3000	No	48.6
Deferred Pruning	2560×1440	No	60.0
	$2 \times 960 \times 1080$	Yes	41.5
	6000×3000	No	32.4

From the table above, both deferred pruning and early pruning achieve interactive rates in desktop settings. However, for stereo rendering using a WebVR-enabled browser and high-resolution rendering, deferred pruning suffers from lower frame rate. This is because deferred pruning carries out the texture sampling after the vertex transformation for each billboard. However, the advantage of the deferred pruning approach is that it achieves better anti-aliasing results than early pruning as shown in Figure. 4.7. On the other hand, the early pruning approach samples the colors at an early stage and discards unnecessary background pixels in the fragment shader, making it a faster approach than the deferred pruning.

Please visit the websites `http://video-fields.com` for supplementary materials related to this paper.

4.5 Conclusions and Future Work

In this chapter, we have described a web-based rendering system that fuses multiple surveillance videos to create a dynamic virtual environment.

Our approach leverages the recent advances in web-based rendering to design and implement the *Video Fields* system to provide a more immersive and easier-to-use dynamic virtual environment for visualizing multiple videos in their appropriate spatial context. We have compared two new ways of fusing moving entities into the 3D world: early pruning and deferred pruning. We found that each has its relative advantages. The choice of which technique to use will depend on the characteristics of the environment being recorded as well as the preferred display device.

Future systems may scale up our approach to handle hundreds of surveillance videos spread over a wider area. To do so effectively, one may use techniques from image and mesh saliency [63, 64, 100, 142] to extract saliency moving entities. As we broaden the scale and scope of our work, more efficient distributed systems and parallel computing algorithms will be necessary to achieve interactive rendering rates. We also plan to explore the integration of our efforts with scalable, distributed, and parallel web-services platforms such as Amazon's S3.

Video Fields system renders a dynamic scene yet with simple geometries. In Chapter 6, we introduce a real-time solution for fusing multiview video textures with complex dynamic meshes.

Chapter 5: Integrating Haptics and Visual Cryptography into Virtual Environments

In addition to panoramas, images, text, multiview videos, and meshes, VR and AR may also be empowered by gesture recognition, haptic feedback, and visual cryptography. In this chapter, we introduce two projects, VRSurus and ARCrypt, to explore these applications.

5.1 VRSurus: Enhancing Interactivity and Tangibility of Puppets in Virtual Reality

Puppets are widely used in storytelling for puppetry on the stage as well as playing games among children. However, as physical inanimate objects, traditional puppets can hardly provide immersive auditory and haptic feedback. Previous research such as 3D Puppetry [143] and Video Puppetry [144] used a *Microsoft Kinect* or an overhead camera to create digital animation with rigid puppets. Nevertheless, they did not allow users to *wear* a flexible puppet in an immersive virtual reality environment. As exploratory work, we started with the following questions: "Could we endow puppets with more interactivity in both real and virtual world? Could we make puppetry more immersive with virtual reality? Could we allow puppeteers to feel the life of puppets via haptic feedback?”

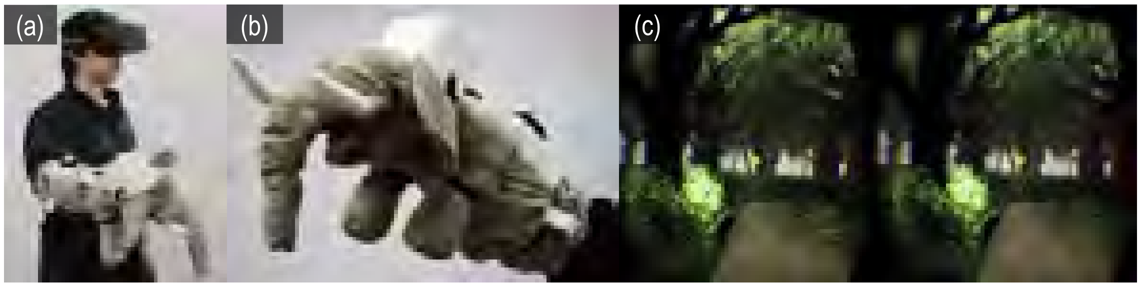


Figure 5.1: Overview of *VRSurus*. (a) a puppeteer wearing an elephant puppet with *VRSurus* playing a VR game, (b) shows a closer view of the elephant puppet wearing a custom 3D-printed cap that encapsulates the Arduino microcontroller, an inertial measurement unit and other electronic modules, and (c) shows the educational VR game empowered with gesture recognition and tactile feedback.

Inspired by our previous project HandSight [145, 146, 147], which supports the activities of daily living (ADLs) by sensing and feeding back non-tactile information about the physical world as it is touched, we have designed and developed *VRSurus* (Figure 5.1), a smart device that enhances interactivity and tangibility of puppets using gesture recognition and tactile feedback in a virtual reality environment. The conceptual design of *VRSurus* is illustrated in Figure 5.2. *VRSurus* uses simple machine-learning algorithms and an accelerometer to recognize three gestures. It is also built with servo motors, solenoids and vibration motors to render tactile sensations. *VRSurus* is designed in the “hat” form so that it could be mounted upon any puppet. We have also implemented a serious VR game where the puppeteer manipulates a virtual elephant puppet (corresponding to the physical puppet) to clear the litter in the forests, splash water onto the lumbermen and destroy factories that pollute the air. The game is designed to educate children on environmental protection.

Our main contribution is the concept and development of an interactive puppet for virtual reality powered by tactile feedback and gesture recognition, as well as our specific mechanical and software design. The main benefit of our approach is its applicability to VR games and puppetry performance. We have presented this game at the *ACM UIST 2015 Student Innovation Contest* in Charlotte, North Carolina. Please see our supplementary video for a demonstration: `www.vrsurus.com`

5.1.1 Hardware Design

To empower the puppet's I/O abilities, we have leveraged Pololu MinIMU-9[1] (only accelerometer is used in the current prototype; gyroscope and compass are reserved for future use) to capture the puppeteer's gestures (input) and small push-pull solenoids, servo motors, and vibration motors to simulate tactile feedback and tangible animations on the arm (output). As shown in Figure 5.3, two solenoids, one on either side of the forearm, indicate the directions of the targets. Two servo motors may pull parts of the puppet using a string to simulate physical animations. Additionally, vibration motors can generate vibration feedback on the entire puppet.

5.1.2 Gesture Recognizer

We have trained our classifier using the decision tree algorithm in Weka [4]. In total, we used the following sixteen features: the sum of mean values on all axes, the differences between each pair of axes, the power of each axis, the range

[1]Pololu MinIMU-9: `https://www.pololu.com/product/1265`

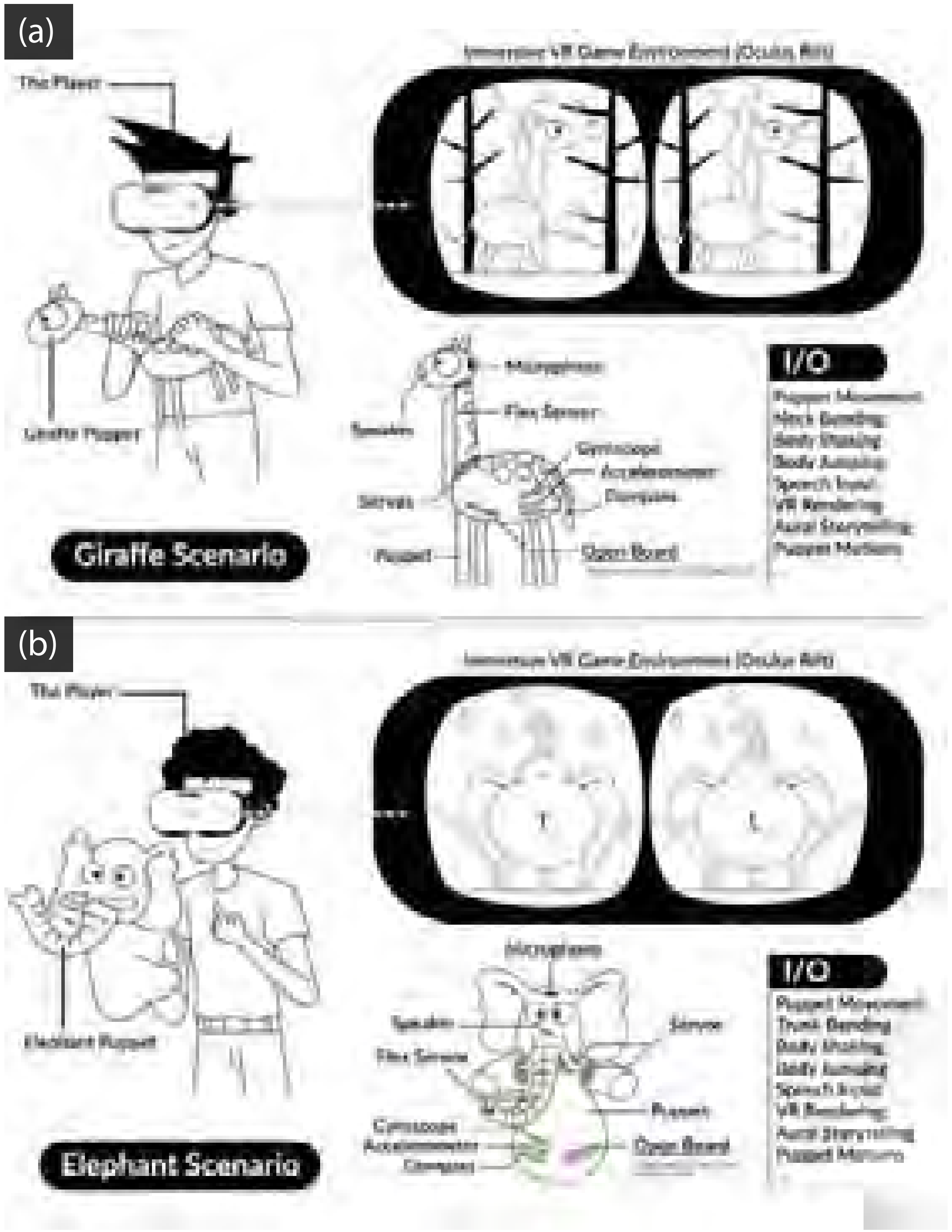


Figure 5.2: Sketches of our conception of *VRSurus*. (a) shows a puppeteer controlling a giraffe puppet to play a platform VR game; (b) shows a puppeteer controlling an elephant puppet to play a first-person VR game. Initially, we planned to sew sensors directly onto the puppet. In subsequent iterations, we created a hat-like attachment which allows *VRSurus* to be mounted onto any puppet.

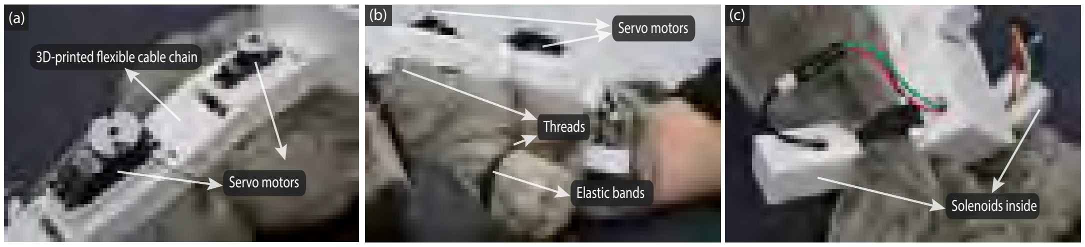


Figure 5.3: Mechanical design of *VRSurus* in details: (a) shows how two servo motors are seated inside the 3D-printed flexible cable chain, (b) shows how servo motors pull the forelegs of the puppet via threads and elastic bands, and (c) shows how solenoids were embedded inside 3D-printed modules around the puppeteer's arm.

of each axis, the cross product between every two axes and the standard deviation of values along each axis. To recognize the four gestures: idle, swiping, shaking and thrusting (Figure 5.4), we collected 240 sets of raw accelerometer values for each gesture from 4 lab members (60 sets per gesture per person). We achieved an average of 97% accuracy using 5-fold cross-validation. The classifier and all signal processing program is written in Java.

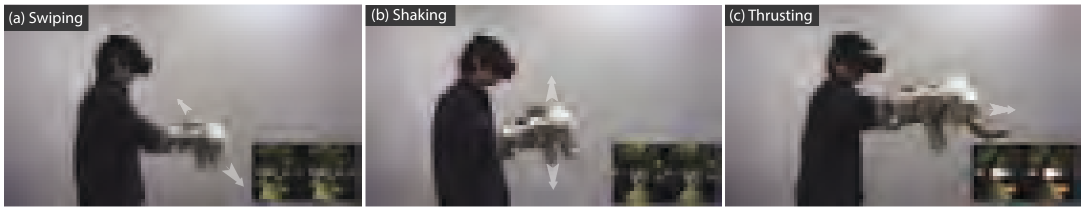


Figure 5.4: This figure shows three gestures we designed to control the virtual character: (a) swiping from left to right triggers the character to swing the nose and blow wind, (b) shaking up and down triggers the character to splash water, and (c) thrusting from inward to outward triggers the character to howl, trample on the ground and throw fireballs. A video demonstration is included at `www.vrsurus.com`.

5.1.3 VR Educational Game

We designed and implemented a proof-of-concept serious VR game based on WebGL and WebVR written in Javascript and PHP (Figure 5.5). We use `Three.js`[2] for cross-platform rendering and `Sea3D Studio`[3] for rendering the animation in WebGL. In the VR game, the player acts as a little elephant called Surus to prevent evil human beings from invading the forest. The player is able to use defined gestures to commit attacks as stated in the Ideation section. The target can be litter, a lumberman and a polluting factory. The goal of the game is to defeat as many enemies as possible in a limited time. We designed a tutorial session before the game, encouraging players to familiarize themselves with each gesture and all the enemies. When a lumberman appears, the game will play the 3D audio of axe-chopping according to the orientation between the player and the lumberman. Meanwhile, the game sends signals to *VRSurus* to enable the left or right solenoid to tap on the user's arm. When the factory is destroyed, the game will output vibration feedback along with the 3D audio of an explosion. After each gesture is successfully performed, the servo motors are signaled to string the elephants' ears and forelegs backward and forward, simulating physical movements for the puppet. We have also written a web-server in PHP that exchanges signals between the gesture recognizer and the VR game every 10ms.

[2]Three.js: `http://www.threejs.org`
[3]Sea3D: `http://sea3d.poonya.com`

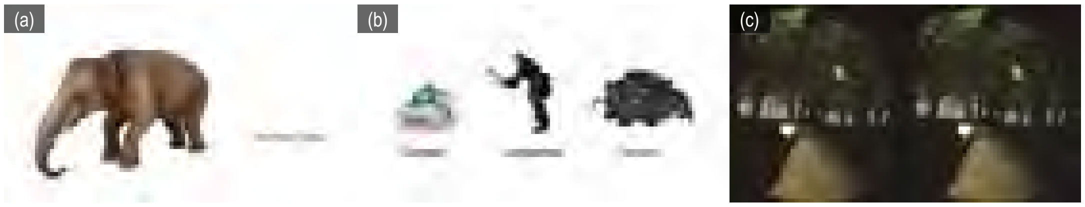


Figure 5.5: Virtual characters in the VR game. (a) shows the hero **Surus** controlled by the puppeteer, (b) shows the the enemies, (c) shows the Oculus Rift HMD rendering results of the lumberman being defeated by Surus.

5.1.4 Deployment

VRSurus was initially deployed for the *ACM UIST 2015 Student Innovation Contest* in Charlotte, NC. It was demonstrated live throughout the entire 2.5-hour contest (Figure 5.6). Each session lasted about two minutes (one exact minute for the VR game and about one minute for the introduction and hardware preparation). In total, there were 63 participants from the UIST community including two invited K12 specialists, who tried out our device. There were more than a hundred people who watched our device as well as the gaming procedure. Overall, the audience's reactions were positive. After successfully destroying an enemy in the VR environment, many participants cheered and some even laughed loudly. Some people mentioned that the tutorial greatly helped them learn the gestures. People were also greatly encouraged when the game told them that how many trees they "saved" in the end.

We also received some criticisms and suggestions for future work. One major concern was that the current prototype is a little fragile and heavy to wear over an extended period, although the device is small enough to be put upon the puppet. Some people also mentioned that the gesture recognizer failed to recognize

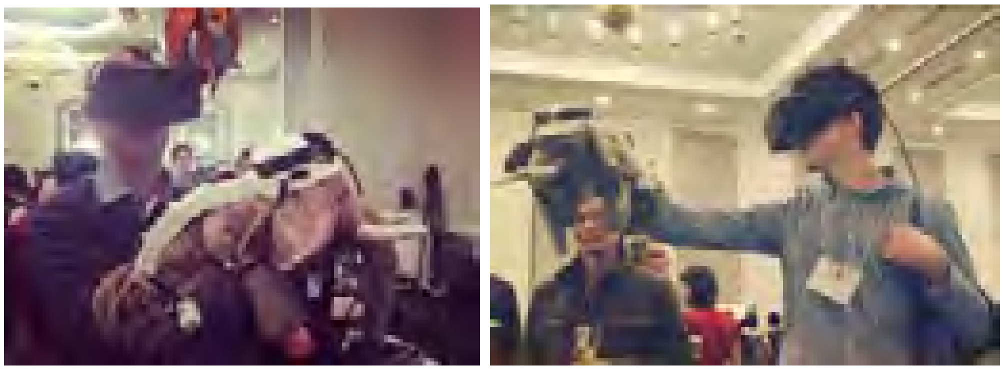


Figure 5.6: During the preliminary deployment, participants from the UIST community interacted with our prototype of *VRSurus* at the ACM UIST 2015 Student Innovation Contest

their gesture when moving slowly and the tutorial did not give them feedback. So during the deployment, we sometimes instructed people to move faster to enable the accelerometer to generate better data. In our next iteration, we plan to train the recognizer with more gestures from more participants and at varying rates of movements.

5.2 ARCrypt: Visual Cryptography with Misalignment Tolerance using Augmented Reality Head-Mounted Displays

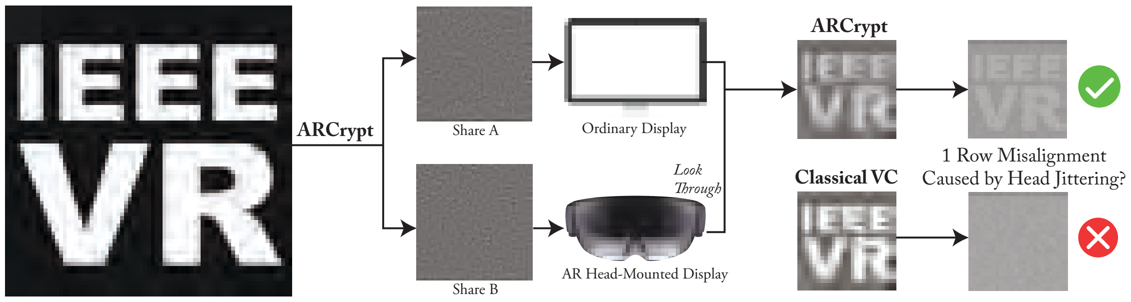


Figure 5.7: Results and overview of our system, *ARCrypt*. *ARCrypt* is able to split a confidential message into two shares of images, which guarantees that the original information could not be revealed with either share of the image alone. The *ARCrypt* system then transmits the two shares to an ordinary display and an augmented reality head-mounted display, respectively. When the user looks through the two aligned images, the secret message is revealed directly to the user's human visual system. Nevertheless, head jittering may cause the two images to misalign with each other. The proposed *ARCrypt* algorithm outperforms the original visual cryptography algorithm in the presence of one or two rows of misalignment.

A vast amount of private data is displayed on our monitors every day. For example, the social security numbers from a company's documents or paychecks, credit card numbers shown in the bill of payment, the plain text of passwords appearing in the registration emails, as well as keycodes from the two-factor authentication messages. Nevertheless, all of these confidential data can easily be eavesdropped by skilled and persistent hackers via cyber-attacks, captured by secret video cameras, or even spied upon by random passersby. Consequently, there is a great demand to have a secure mechanism for protecting information displayed on the screen.

In the past decades, researchers and scientists have designed sophisticated cryptographic algorithms to encrypt and decrypt messages. However, the device that executes the decryption algorithm may be under threats from attackers. Some other solutions incur new trusted computing base (TCB) such as smart-phones [148] and two-factor authentication [149] to address the problem. Nevertheless, the new TCB can also be compromised as a result of an implementation flaw in the secure protocol during the communication. For example, the heart-bleed bug, found in OpenSSL, a popular secure protocol, was taken advantage of to steal information over the SSL/TLS encryption [150].

In this section, we introduce *ARCrypt*, a practical and robust cryptographic system that eliminates every device from the TCBs and assumes no connection between the TCBs. First, the encryption algorithm in the *ARCrypt* system splits the confidential information (as an image) into two shares. One share of data is displayed on the ordinary screen while the other share of data is displayed on the untethered augmented reality headset, such as *Microsoft HoloLens*. Eventually, the human operator has to conduct the decryption operation by themselves by manually aligning the two shares of information together. Our work was built upon the pioneering research by Andrabi *et al.*[5], which first demonstrates the usability of using tethered augmented reality headsets like Google Glass to reveal secret messages using the visual cryptography system induced by Naor and Shamir [151]. However, our work differs from their system by taking advantage of a novel visual cryptography algorithm with misalignment tolerance, which makes the system more practical to use.

We have modeled the misalignment of head jittering using a 2D Gaussian distribution. We have developed a novel algorithm to enhance the visibility of the classical visual cryptography via diffusion with Gaussian kernels, thus enabling the algorithm to be tolerant with one or two rows of misalignment.

5.2.1 Background and Related Work

Our work builds upon the recent advances in augmented reality (AR) headsets, as well as the theory of visual cryptography (VC) and its recent implementations on AR headsets.

5.2.1.1 Augmented Reality Head Mounted Displays

In recent years, there has been an increasing number of commercial AR head-mounted displays, such as *Google Glass*, *ODG Smart Glasses*, *Meta Headsets*, *Epson Moverio*, and *Microsoft HoloLens*. These displays blend the virtual information rendered by the computer with the real scene observed by the user.

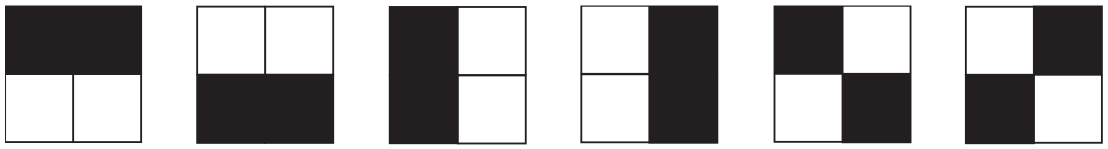
There are several factors to be considered in selecting AR displays and designing rendering algorithms and content for these displays, in which field of view (FoV) and resolution are the dominant ones. In this project, we chose *Microsoft HoloLens* because it has a wider FoV and a better resolution (30°FoV, 1268×720 pixels), compared with *Google Glass* (14°, 640×360 pixels) and *Epson Moverio* (23°, 960×540 pixels). Moreover, *Microsoft HoloLens* itself is a standalone and untethered computer, which is more likely to be considered as a trusted computing base when the Internet connection is switched off.

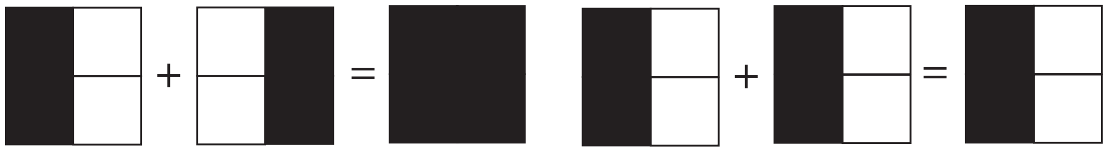
As most of the AR headsets, *Microsoft HoloLens* uses additive color mixing strategy to project visual information onto the display [152]. Consequently, for each pixel, the lower RGB values it has, the less visible it is through the display. Besides, the black color indicates total transparency in HoloLens; thus rendering a black quad through HoloLens to observe a white quad yields a white quad in the user's perception; a semi-transparent red quad through the *HoloLens* overlaid on a green quad yields a yellow quad.

5.2.1.2 Visual Cryptography

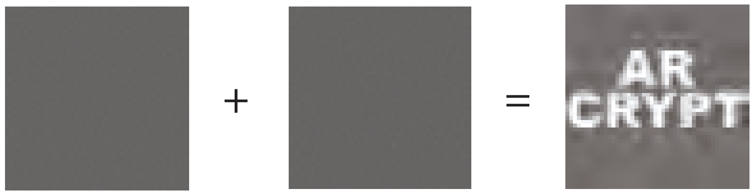
Visual cryptography is a cryptographic scheme in which decodes a secret image without any computational cryptographic operations. The well-known visual cryptography fundamental theory was firstly developed by Naor and Shamir [151]. An example of visual cryptography with two shares is shown in Figure 5.8. First, the algorithm splits one message into N shares with different transparency. Suppose the original image has $w \times w$ pixels, each share will have $\frac{w}{2} \times \frac{w}{2}$ blocks of 2×2 pixels. Each block will be one of the six patterns as shown in Figure 5.8(a). Meanwhile, it ensures that a person with any K shares of the data can visually restore the image by stacking their transparencies (as in Figure 5.8(b) and (c)), but any $K-1$ shares of the data cannot restore the image.

Carlo *et al.*[153] advanced the theoretical foundations of visual cryptography for use with grey-scale images. Zhou *et al.*[154] have combined dithering techniques with visual cryptography to encrypt gray-scale images. Hou *et al.*[155] have presented novel algorithms of visual cryptography for color images. Bin *et al.*[156] have


(a) Valid horizontal, vertial, and diagnoal blocks for an image share



(b) Revealing a black or (noisy) white block using two shares of images



(c) A concrete example of visual cryptography

Figure 5.8: Examples of visual cryptography proposed by Naor and Shamir [151]. (a) shows six valid patterns for one 2×2 block of pixels to be selected for an image share, (b) shows the *addition* operator when fusing one share with the other share, and (c) shows our example of revealing the characters “AR CRYPT” from two image shares.

proposed an edge-preserving technique for dithering to improve visual quality for visual cryptography.

Recently, Andrabi *et al.*[5] conducted the first formal user study to investigate the feasibility and usability of using *Google Glass* and *Epson Moverio* for reading visual cryptography (Figure 5.9). Their system requires users to use a chin rest to minimize head jittering effects. Even with the chin rest, users spent a considerable amount of time initially aligning the image shares: ranging from 18.49 to 313.32 seconds. Apart from the effort in initial alignment, the participants spent approximately 8 seconds to decode and recognize a single plaintext character.

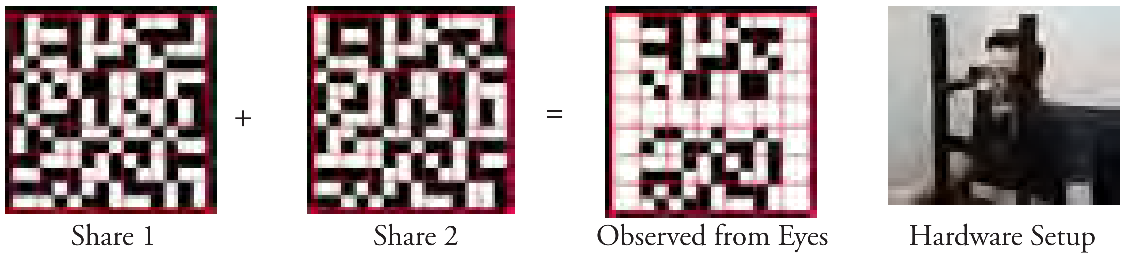


Figure 5.9: The pioneering AR-based visual cryptography system, developed by Andrabi *et al.*[5], is able to encrypt a single character with the help of a chin rest.

Our work is motivated by the challenge of misalignment, inspired by the study conducted by Andrabi *et al.*[5]. Specifically, we desire to enable the user to see the fused image even though the two images are not perfectly aligned. Although the latest *Microsoft HoloLens* has the capability of detecting the depth map and stabilizing the image at a 3D position of the reconstructed world, the depth mapping, and tracking is still far from perfect. It is likely that the image could move a couple of pixels with a little bit of the user's head jitter. An example taken from *HoloLens*'s mixed-reality capture is shown in Figure 5.10.

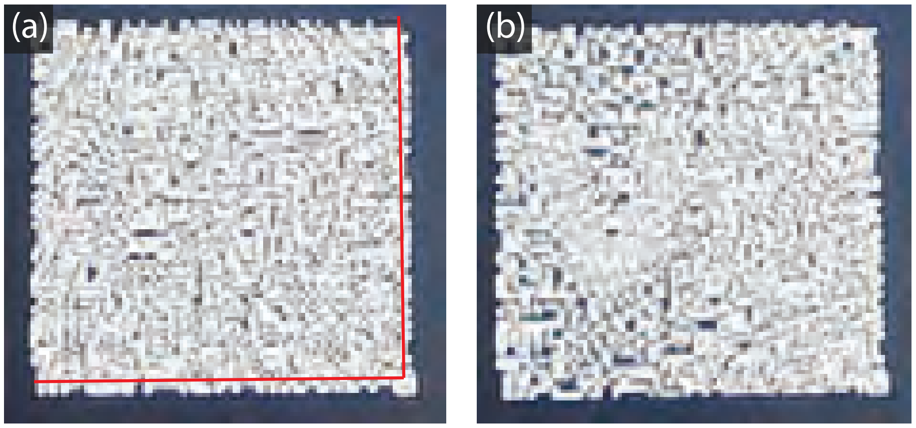


Figure 5.10: A real case of misalignment challenge for visual cryptography using augmented reality headsets. The red line shows the border of the overlaid image. Compared with (a), (b) has better alignment than (a), so the left character "V" appears from the HoloLens. "R" cannot be observed clearly from (b), because the HoloLens's debugging camera has a little translation to the actual human eyes. In reality, both "V" and "R" are shown in (b)'s condition but not in (a).

5.2.2 ARCrypt Algorithms

In this section, we describe the core algorithm of *ARCrypt*, which has tolerance for limited misalignment. The main idea behind *ARCrypt* is: for each pixel p in one share, model the probability of misalignment on another pixel q as a 2D Gaussian distribution centered at the pixel p. In this way, we sacrifice a little contrast in the fused result for better clarity when one or two rows of misalignment occurs.

5.2.2.1 Preprocessing

Following [5] and [151], given a confidential visual image $\boldsymbol{I}$, we first generate a binary image $\hat{\boldsymbol{I}}$ by thesholding every 2×2 block of pixels in $\boldsymbol{I}$. Here, we denote $\mathscr{F}(\hat{\boldsymbol{I}})$ and $\mathscr{B}(\hat{\boldsymbol{I}})$ as the set of foreground (white) and background (black) pixels of $\hat{\boldsymbol{I}}$, respectively.

Next, we model the range of misalignment as a $s \times s$ square. We generate a $s \times s$ 2D Gaussian kernel $\mathscr{G}(x,y,\sigma)$ at scale σ:

$$\mathscr{G}(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{5.1}$$

where σ indicates the standard deviation of the misalignment. In our experiments, we choose $s = 3, \sigma = 1.0$ and $s = 5, \sigma = 2.0$.

5.2.2.2 Generation of Two Shares

ARCrypt algorithm generates the first share as the classical VC approach, as shown in Figure 5.8. For each 2×2 block of pixels in the first share, we randomly choose one of the six VC patterns. Next, we carry out two solutions to deal with the possible misalignment:

1. **ARCrypt***: for the second share, we only diffuse the foreground pixels: each foreground pixel has a probability to be misaligned with one of its surrounding pixels; in this way, when the two shares match perfectly, the background is unchanged, but the foreground is darker.
2. **ARCrypt**: for the second share, we diffuse both the background and foreground pixels to enhance the contrast: every pixel has a probability to be misaligned with one of its surrounding pixels.

The pseudo code of the core algorithm is shown in Algorithm 5:

ALGORITHM 5: ARCrypt: Visual Cryptography With Misalignment Tolerance

```
**Input:** binary secret image $\hat{\boldsymbol{I}}$
**Output:** two shares of information $\boldsymbol{I}^\alpha$ and $\boldsymbol{I}^\beta$
Generate a random share of $\boldsymbol{I}^\alpha$ using the six random patterns;
for each $2 \times 2$ block $b_{rc}$ of $\hat{\boldsymbol{I}}$ do
    for each $2 \times 2$ block $b_{ij}$ of $\hat{\boldsymbol{I}}$, where $|r-i| \le \frac{s}{2}, |c-j| \le \frac{s}{2}$ do
        if $b_{ij} \in \mathscr{F}(IB)$ or AllowBackgroundDiffusion then
            Look up the probability of misaligning $b_{rc}$ with $b_{ij}$ from the from
              the Gaussian kernel $\mathscr{G}(x,y,\sigma)$: $\mathscr{P}(b_{rc}, b_{ij}) \leftarrow \mathscr{G}(r-i, c-j, \sigma)$ ;
            Increase the normalization factor of $b_{rc}$: $\mathscr{N}_{rc} \leftarrow \mathscr{N}_{rc} + \mathscr{P}(b_{rc}, b_{ij})$;
        end
    end
    Normalize the probabilities: $\mathscr{P}(b_{rc}, b_{ij}) \leftarrow \mathscr{P}(b_{rc}, b_{ij}) / \mathscr{N}_{rc}$;
    Generate a random number from a uniform distribution: $r \in [0,1]$;
    Set the accumulated probabilities: $\mathscr{A}_{rc} \leftarrow 0$ for each $2 \times 2$ block $b_{ij}$ of $\hat{\boldsymbol{I}}$,
      where $|r-i| \le \frac{s}{2}, |c-j| \le \frac{s}{2}$ do
        if $b_{ij} \in \mathscr{F}(IB)$ or AllowBackgroundDiffusion then
            $\mathscr{A}_{rc} \leftarrow \mathscr{A}_{rc} + \mathscr{P}(b_{rc}, b_{ij})$;
            if $r \le \mathscr{A}_{rc}$ then
                $(p,q) \leftarrow (i,j)$;
                break;
            end
        end
    end
    if $b_{pq} \in \mathscr{B}(\hat{\boldsymbol{I}})$ then
        $\boldsymbol{I}^\beta_{rc} \leftarrow \boldsymbol{I}^\alpha_{pq}$ ;
    end
    else
        $\boldsymbol{I}^\beta_{rc} \leftarrow WHITE - \boldsymbol{I}^\alpha_{pq}$ ;
    end
end
```

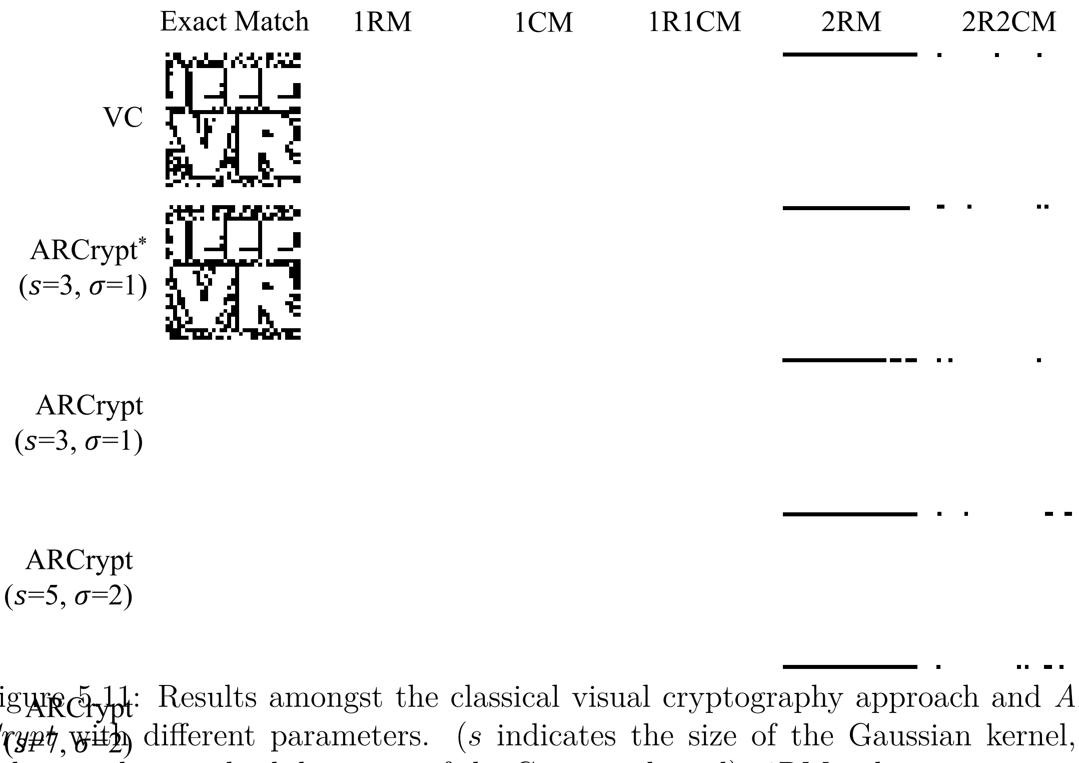


Figure 5.11: Results amongst the classical visual cryptography approach and *ARCrypt* with different parameters. (s indicates the size of the Gaussian kernel, σ indicates the standard deviation of the Gaussian kernel). 1RM indicates one row of misalignment, 1CM indicates one column of misalignment, and so forth. From the results, we observe that both *ARCrypt* and *ARCrypt* outperform the visual cryptography algorithm for 1RM or 1CM misalignment. *ARCrypt* provides better contrast than *ARCrypt* when 1RM or 1CM misalignment occurs.

5.2.3 Experimental Results

Considering that the resolution of *HoloLens* is 1268×720 pixels, we generate visual cryptography images at the resolution of 1024×1024 pixels using a custom $C{+}{+}$ program using the proposed *ARCrypt* algorithm, as well as the classical visual cryptography algorithm. We render the misalignment under five conditions: one row (2 pixels) misalignment (1RM), one column misalignment (1CM), one row and one column misalignment (1R1CM), two rows misalignment (2RM), and two rows and two columns misalignment (2R2CM). The images are rendered in a high PPI (pixels per inch) monitor (PPI = 264). Lower contrast may be expected in a lower PPI monitor.

As shown in Figure 5.11, we arrive at the following insights:

1. The classical visual cryptography algorithm does not work with even a single row or column of misalignment, making it hard to interpret the image using augmented reality headsets.

2. *ARCrypt** can deal with one row or one column misalignment (2 pixels) while preserving as good a contrast as the original visual cryptography algorithm. However, the contrast drops with misalignment.

3. *ARCrypt* provides better contrast than *ARCrypt** when misalignment occurs and even works for the 1R1CM case (2 pixels misaligned both horizontally and vertically). After increasing the size and scale of the Gaussian kernel, we can still see the secret message even with 2 rows (4 pixels) of misalignment.

4. *ARCrypt* cannot deal with the 2R2CM case (4 pixels misaligned both horizontally and vertically).

5.3 Conclusions and Future Work

In this section, we have adapted gesture recognition, tactile feedback, and visual cryptography for current-generation virtual and augmented reality headsets.

First, we have described *VRSurus*, a prototype device that enhances the interactivity of puppets with gesture recognition and haptic feedback for virtual reality gaming. We received initial feedback from participants at the *ACM UIST 2015 Stu-*

dent Innovation Contest. We have identified several directions for future work. One is to extend more input modules for better animations. For example, by embedding flex sensors in different parts of the puppet (*e.g.*, ears and nose), the virtual character can animate corresponding parts when the puppeteer manipulates the physical one in the real world. By using a microphone, we can facilitate performance which kinetic input cannot support, such as creating a conversation with other virtual roles. Additionally, it could be a promising storytelling tool to express children's creativity. For instance, a child can experience others' stories with recorded sound, tactile feedback and physical movements all at the same time.

Next, our system *ARCrypt* uses a novel visual cryptography algorithm which is tolerant to users' head jitter and misalignment of the two shares of encrypted visual information. We achieve this by modeling the misalignment through a 2D Gaussian distribution of the visual cryptography's random patterns. This allows us to trade off precise alignment with perceived contrast. We believe *ARCrypt* provides a versatile, commodity, off-the-shelf solution for embedding encrypted augmented reality information in the real-world displays, thereby protecting confidential data while facilitating an easy-to-use visual decryption. Future system may take advantage of robust tracking algorithms to automatically align the two shares of encrypted visual information.

Chapter 6: Montage4D: Real-time Seamless Fusion of Multiview Video Textures

6.1 Introduction

With recent advances in consumer-level virtual and augmented reality, several dynamic scene reconstruction systems have emerged, including *KinectFusion* [157], *DynamicFusion* [158], *Free-Viewpoint Video* [159], and *Holoportation* [160]. Such 4D reconstruction technology is becoming a vital foundation for a diverse set of applications such as 3D telepresence for business, live concert broadcasting, family gatherings, and remote education.

Among these systems, *Holoportation* is the first to achieve real-time, high-fidelity 4D reconstruction without any prior knowledge of the imaged subjects. The success of this system builds upon the breakthrough of fast non-rigid alignment algorithms in fusing multiview depth streams into a volumetric representation by the *Fusion4D* system [161]. Although *Holoportation* is able to mitigate a variety of artifacts using techniques such as normal-weighted blending and multilevel majority voting, some artifacts persist. In a previous user study on *Holoportation* [160], around 30% of the participants did not find that the reconstructed model real com-

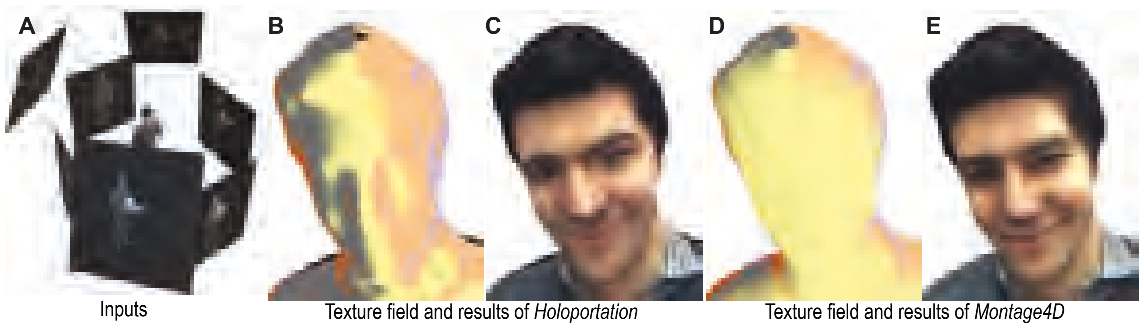

A
B
C
D
E
Inputs
Texture field and results of *Holoportation*
Texture field and results of *Montage4D*

Figure 6.1: The Montage4D algorithm stitches multiview video textures onto dynamic meshes seamlessly and at interactive rates. (A) inputs: dynamic triangle meshes reconstructed by the Fusion4D algorithm, multiview video textures, and camera poses; (B) merged texture result using Holoportation, which employs normal-weighted blending with dilated depth discontinuities and a majority voting algorithm; (C) the corresponding color-coded field of scalar texture weights of (B), which we call a *texture field*; (D) result using Montage4D, which favors the dominant view, ensures temporal consistency, and also reduces seams between camera views; (E) the corresponding texture field of (D).

pared with a real person. We believe that this is a significant challenge that must be addressed before telepresence can be embraced by the masses. We also note that the user feedback about visual quality was much less positive than other aspects (speed and usability). This is caused by the blurring and visible seams in the rendering results, especially on human faces, as shown in Figure 6.1.

Blurring arises because of two reasons. First, texture projection from the camera to the geometry leads to registration errors around visible seams. Second, normal-weighted blending of the different views with different appearance attributes (specular highlights and inconsistent color calibration), leads to an inappropriate mixing of colors and therefore blurring.

We further characterize visible seams into: *Misregistration seams* are caused by imprecisely reconstructed geometry with missing or extruded triangles. *Occlu-*

sion seams arise out of discontinuous texture transitions across the field of view of multiple cameras and self-occlusions.

In this chapter, we address both blurring and visible seams and achieve seamless fusion of video textures at interactive rates. Our algorithm estimates the misregistration and occlusion seams based on the self-occlusion from dilated depth discontinuities, multi-level majority voting, foreground segmentation, and the field-of-view of the texture maps. To achieve a smooth transition from one view to another, we compute geodesic distance fields [162] from the seams, to spatially diffuse the texture fields to the visible seams. In order to prevent view-dependent texture weights from rapidly changing with the viewpoints, we extend the scalar texture field as shown in Figure 6.1(C) to a temporally changing field to smoothly update the texture weights. As shown in Figure 6.1(D) and 6.9, our system achieves significantly higher visual quality at interactive rates compared to the state-of-the-art *Holoportation* system. Please refer to `www.montage4d.com` for the supplementary video, slides, and future code and datasets release.

The main contributions of the *Montage4D* work are:

- formulation and quantification of the misregistration and occlusion seams for fusing multiview video textures,
- use of equidistance geodesics from the seams based on discrete differential geometry concepts to diffuse texture fields,
- temporal texture fields to achieve temporal consistency of the rendered imagery, and

- a fast computational pipeline for high-fidelity, seamless video-based rendering, enabling effective telepresence and customized real-time stylization.

6.2 Related Work

We build upon a rich literature of prior art on image-based 3D reconstruction, texture stitching, and discrete geodesics.

6.2.1 Image-based 3D Reconstruction

Image-based 3D reconstruction has been researched extensively in the past decades. The pioneering work of Fuchs *et al.*[163, 164] envisioned that a patient on the operating table could be acquired by a sea of structured-light cameras, and a remote doctor could conduct medical teleconsultation with a head-mounted display. Kanade *et al.*[165] invented one of the earliest systems that use a dome of cameras to generate novel views via triangulated depth maps. Its successor, *3D Dome* [166], reconstructs explicit surfaces with projected texture. Towles *et al.*[167] achieve real-time 3D telepresence over networks using 3D point clouds. Goldluecke *et al.*[168] adopt spatiotemporal level sets for volumetric reconstruction. Furukawa *et al.*[169] reconstruct deformable meshes by optimizing traces of vertices over time. While compelling, it takes two minutes on a dual Xeon 3.2 GHz workstation to process a single frame. De *et al.*[170] present a system that reconstructs space-time coherent geometry with motion and textural surface appearance of actors performing complex and rapid moves. However, this also suffers from slow processing speed (approximately 10 minutes per frame), largely due to challenges in stereo matching and optimization. Since then, a number of advances have been made in dealing with video constraints and rendering quality [90, 159, 171, 172, 173, 174, 175, 176, 177, 178, 179], but rendering dynamic scenes in real time from video streams has remained a challenge. Zitnick *et al.*[180] present an efficient rendering system which interpolates the adjacent two views with a boundary layer and video matting. However, they consider a 2.5D layered representation for the scene geometry rather than a general mesh model that can be viewed from all directions. Their work inspires us with the computation of depth discontinuity and seam diffusion.

With recent advances in consumer-level depth sensors, several reconstruction systems can now generate dynamic point-cloud geometries. *KinectFusion* [157, 181] is the first system that tracks and fuses point clouds into dense meshes using a single depth sensor. However, the initial version of *KinectFusion* cannot handle dynamic scenes. The systems developed by Ye *et al.*[182] and Zhang *et al.*[183] are able to reconstruct non-rigid motion for articulated objects, such as human bodies and animals. Further advances by Newcombe *et al.*[158] and Xu *et al.*[184] have achieved more robust dynamic 3D reconstruction from a single Kinect sensor by using warp-fields or subspaces for the surface deformation. Both techniques warp a reference volume non-rigidly to each new input frame. Guo *et al.*[185, 186] and Yu *et al.*[187] have realized real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. However, the reconstructed scenes still suffer from the occlusion issues since the data comes from a single depth sensor. In addition, many 3D reconstruction systems rely on a volumetric model that is used for model fitting, which is limited in accommodating fast movement and major shape changes.

Collet *et al.*[159] have demonstrated the *Free-Viewpoint Video*, an offline pipeline to reconstruct dynamic textured models in a studio setup with 106 cameras. However, it requires controlled lighting, calibration, and approximately 28 minutes per frame for reconstruction, texturing, and compression. Furthermore, Prada *et al.*[177, 178] present a unified framework for evolving the mesh triangles and the spatiotemporal parametric texture atlas. Nonetheless, the average processing time for a single frame is around 80 seconds, which is not yet applicable for real-time applications.

Orts *et al.*[160] present *Holoportation*, a real-time pipeline to capture dynamic 3D scenes by using multiple RGBD cameras. This system is highly robust to sudden motion and large changes in meshes. To achieve real-time performance, their system blends multi-view textures according to the dot product between surface normals and the camera viewpoint directions.

Our system extends the *Holoportation* system and solves the problems of fuzziness caused by normal-weighted blending, visible seams caused by misregistration and occlusion, while ensuring temporal consistency of the rendered images.

In the state-of-the-art work by Dou *et al.*[188] with depth maps generated up to 500Hz [189, 190], a detail layer is computed to capture the high-frequency details and atlas mapping is applied to improve the color fidelity. Our rendering system is compatible with the new fusion pipeline, by integrating the computation of seams, geodesic fields, and view-dependent rendering modules.

6.2.2 Texture Stitching

View-dependent texture-mapping on the GPU has been widely applied for reconstructed 3D models since the seminal work by Debevec *et al.*[191, 192]. However, seamlessly texturing an object by stitching RGB images remains a challenging problem due to inexact geometry, varying lighting conditions, as well as imprecise calibration matrices.

Previous work has considered using global optimization algorithms to improve color-mapping fidelity in static models. For example, Gal *et al.*[193] present a multi-label graph-cut optimization approach that assigns compatible textures to adjacent triangles to minimize the seams on the surface. In addition to the source images, their algorithm also searches over a set of local image transformations that compensate for geometric misalignment using a discrete labeling algorithm. While highly creative and elegant, their approach takes 7 to 30 minutes to process one frame on a mesh with 10,000 to 18,000 triangles. Markov Random Field (MRF) optimization-based approaches [194, 195, 196] are also similarly time intensive. To reduce the seams caused by different lighting conditions, Zhou *et al.*[197] introduce *TextureMontage*, which automatically partitions the mesh and the images, driven solely by feature correspondences. *TextureMontage* integrates a surface texture in-painting technique to fill in the remaining charts of the surface with no corresponding texture patches. However, their approach takes over 30 minutes per frame to process. Zhou *et al.*[198] optimize camera poses in tandem with non-rigid correction functions for all images at the cost of over 30 minutes per frame. Narayan *et al.*[199]

jointly optimize a non-linear least squares objective function over camera poses and a mesh color model at the cost of one to five minutes per frame. They incorporate 2D texture cues, vertex color smoothing, and texture-adaptive camera viewpoint selection into the objective function.

A variety of optical-flow-based approaches have been used to eliminate blurring and ghosting artifacts. For example, Eisemann *et al.*[200] introduce *Floating Texture*, a view-dependent rendering technique with screen-based optical-flow running at 7-22 frames per second.[1] Casas *et al.*[201] extend their online alignment with spatiotemporal coherence running at 18 frames per second. Volino *et al.*[202] employs a surface-based optical flow alignment between views to eliminate blurring and ghosting artifacts. However, the major limitations of optical-flow-based approaches are twofold. First, surface specularity [200], complex deformations, poor color calibration and low-resolution of the textures [201] present challenges in the optical flow estimation. Second, even with GPU computation, the computational overhead of optical flow is still a limitation for real-time rendering. This overhead increases even further with more cameras.

In studio settings, Collet *et al.*[159] have found that with diffused lighting condition and precisely reconstructed surface geometry, direct image projection followed by normal-weighted blending of non-occluded images yields sufficiently accurate results. However, for real-time reconstruction systems with a limited number of cameras, the reconstructed geometries are often imperfect.

[1]We tested *Floating Texture* on a GTX 1080 under target resolution of 1024×1024 and 2048×2048.

Our work focuses on improving the texture fusion for such real-time applications. Building upon the pioneering research above as well as the work of several others, our approach is able to process over 130,000 triangles at over 100 frames per second.

6.2.3 Geodesic Distance Fields

The field of discrete geodesics has witnessed impressive advances over the last decade [203, 204, 205]. Geodesics on smooth surfaces are the straightest and locally shortest curves and have been widely used in a variety of graphics applications such as optimal movement of an animated subject. Mitchell *et al.*[206] devise an exact algorithm for computing the *"single source, all destinations"* geodesic paths. For each edge, their algorithm maintains a set of tuples (windows) for the exact distance fields and directions, and updates the windows with a priority queue like the *Dijkstra* algorithm. However, the worst running time could be $\mathcal{O}\left(n^2 \log n\right)$, and the average is close to $\mathcal{O}\left(n^{1.5}\right)$ [6, 162]. Recently, Qin *et al.*[207] proposes a 4-15 times faster algorithm using window pruning strategies. However, their algorithm aims for the exact geodesic paths and requires $\mathcal{O}\left(n^2\right)$ space like the previous approaches. Kapoor [208] proposes a sophisticated approach for the "single source, single destination" case in $\mathcal{O}\left(n \log^2 n\right)$ time. As for approximate geodesics, Lanthier [209] describes an algorithm that adds many extra edges into the mesh. Kanai and Suzuki [210] and Martinez *et al.*[211] use iterative optimization to converge the geodesic path locally. However, their methods require a large number of iterations.

In this work, we compute geodesics distance fields for weighting the texture fields, so as to assign low weight near the seams and progressively larger weight up to some maximum distance away from the seams. Our goal is to solve the geodesics problem for the *"multiple sources, all destinations"*. Bommes *et al.*[162] have introduced an accurate algorithm for computation of geodesic distance fields. In this paper, we follow a variant of the efficient algorithm developed by Surazhsky *et al.*[6] to measure the approximation of the geodesics fields in $O(n \log n)$ time for a small number of vertices (**seam** vertices are approximately 1% of the total vertices) in a few iterations (typically $15-20$).

6.3 System Overview

In this section we present the workflow of the *Montage4D* system as shown in Figure 6.2:

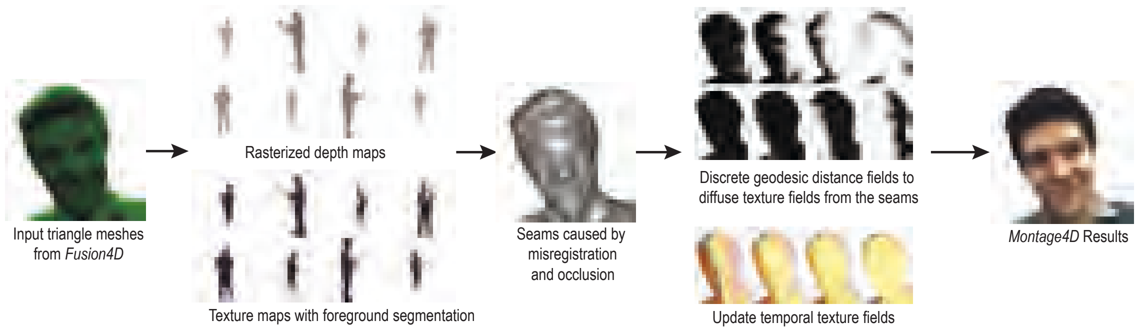

Input triangle meshes from *Fusion4D*
Rasterized depth maps
Texture maps with foreground segmentation
Seams caused by misregistration and occlusion
Discrete geodesic distance fields to diffuse texture fields from the seams
Update temporal texture fields
Montage4D Results

Figure 6.2: The workflow of the Montage4D rendering pipeline.

1. **Streaming of Meshes and Videos**: Our system streams polygonal meshes and video textures from a reconstruction server that runs the *Fusion4D* algorithm. The calibration parameters for projective mapping from camera to

model space are only transferred for the initial frame.

2. **Rasterized depth maps and segmented texture maps**: For each frame, Montage4D estimates rasterized depth maps from each camera's viewpoint and perspective in parallel on the GPU. The video textures are processed with a background subtraction module, using the efficient real-time algorithm for performing mean field inference [212].

3. **Seam identification with dilated depth discontinuities**: The renderer estimates the dilated depth discontinuities from the rasterized depth maps, which are bounded by an estimated reconstruction error e. This is crucial for reducing ghosting artifacts, that arise when missing geometry and self-occlusion cause incorrect color projection on to surfaces. The renderer uses the texture maps to calculate the seams due to each camera's limited field of view.

4. **Geodesic fields**: After the identification stage of all the seams, the renderer calculates each vertex's distance field to the seams based on the straightest discrete geodesic. The distance fields non-linearly filter the texture fields, while ensuring spatial smoothness of the resulting texture fields.

5. **Temporal texture fields**: Using the parameters of the current rendering camera, the renderer also determines the view-dependent weights of each texture. However, if an abrupt jump in viewpoint should occur, the texture weights field may alter rapidly. To overcome this challenge, *Montage4D* em-

ploys the concept of temporal texture weights so that each texture weight alters smoothly towards the target texture weight over time.

6. **Color synthesis and post-processing**: We fuse the sampled color using the temporal texture fields for each pixel in the screen space. Our system also provides an optional post-processing module for screen-space ambient occlusion.

6.4 Algorithms

In this section, we describe the Montage4D algorithms.

6.4.1 Formulation and Goals

For each frame, given a triangle mesh and N video texture maps $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_N$ streamed from the dedicated *Fusion4D* servers, our goal is to assign for each mesh vertex $\mathbf{v}$ a vector $(\mathscr{T}_{\mathbf{v}}^1, \ldots, \mathscr{T}_{\mathbf{v}}^N)$ of scalar texture weights. Let the **texture field** $\mathscr{T}$ denote the piecewise linear interpolation of these vectors over the triangle mesh. For each non-occluded vertex $\mathbf{v} \in \mathbb{R}^3$, we calculate a pair of corresponding (u, v) coordinates for each texture map using back-projection. Finally, the resulting color $c_{\mathbf{v}}$ is fused using the *normalized* texture field $\mathscr{T}_{\mathbf{v}}$ at vertex $\mathbf{v}$:

$$c_{\mathbf{v}} = \sum_{i=1}^{N} c_{\mathbf{v}}^i \cdot \mathscr{T}_{\mathbf{v}}^i = \sum_{i=1}^{N} \text{texture2D}(\mathbf{M}_i, u, v) \cdot \mathscr{T}_{\mathbf{v}}^i \tag{6.1}$$

In order to achieve high-quality rendering, we need to take the following factors into consideration:

1. **Smoothness**: The transition between the texture fields of adjacent vertices should be smooth, because human perception is especially sensitive to texture discontinuities.

2. **Sharpness**: The rendered image should preserve the fine-scale detail of the input textures. However, due to imprecisely reconstructed geometry, fusing all the textures onto the mesh usually results in blurring or ghosting artifacts.

3. **Temporal Consistency**: The texture fields should vary smoothly over time as the mesh changes and as a user's viewpoint changes.

6.4.2 Normal Weighted Blending with Dilated Depth Maps and Coarse-to-Fine Majority Voting Strategy

Our baseline approach is derived from the real-time implementation in the *Holoportation* project. This approach uses normal-weighted blending of non-occluded textures, together with a coarse-to-fine majority voting strategy. For each vertex $\mathbf{v}$, the texture field $\mathscr{T}_{\mathbf{v}}^{i}$ for the i_{th} view is defined as

$$\mathscr{T}_{\mathbf{v}}^{i} = \mathscr{V}_{\mathbf{v}} \cdot \max(0, \hat{\mathbf{n}}_{\mathbf{v}} \cdot \hat{\mathbf{v}}_i)^{\alpha}, \tag{6.2}$$

where $\mathscr{V}_{\mathbf{v}}$ is a visibility test using dilated depth maps and multi-level majority voting algorithm introduced later, $\hat{\mathbf{n}}_{\mathbf{v}}$ is the smoothed normal vector at vertex $\mathbf{v}$, $\hat{\mathbf{v}}_i$ is the view direction of the i_{th} camera, and α determines the smoothness of the transition, and favors the frontal views. This approach determines the texture fields purely based on the geometry, which may have missing or extruded triangles. The resulting texture fields of adjacent vertices may favor completely different views, thus introducing visible seams.

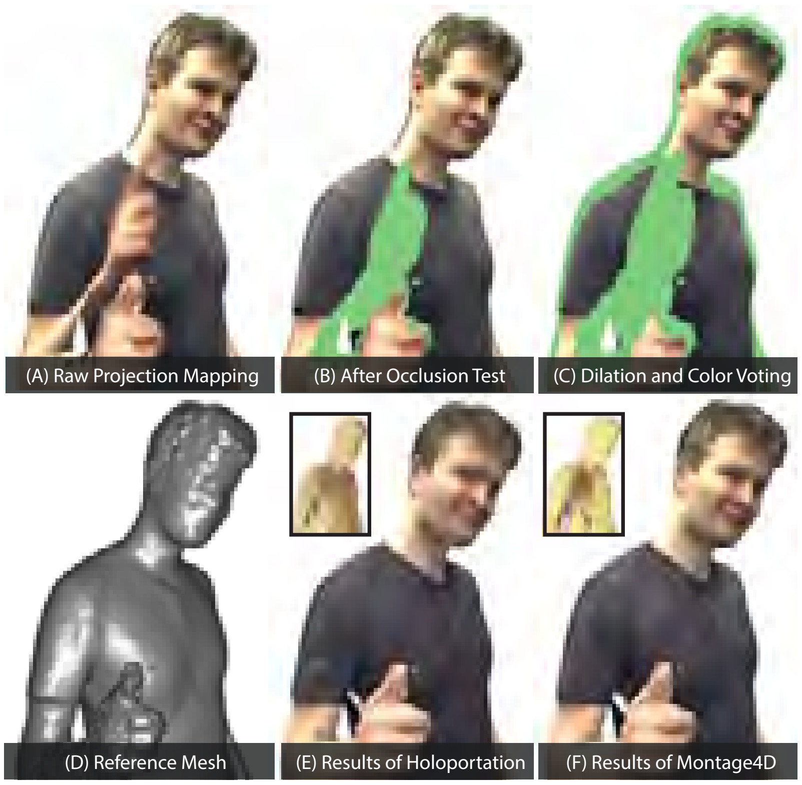

(A) Raw Projection Mapping
(B) After Occlusion Test
(C) Dilation and Color Voting
(D) Reference Mesh
(E) Results of Holoportation
(F) Results of Montage4D

Figure 6.3: This figure shows how texture weight fields improve the rendering quality compared to previous work (the baseline approach). The *Holoportation* approach removes several ghosting artifacts by taking advantage of dilated depth maps and majority voting algorithm (top row), however, the rendering still suffers from fuzziness and visible seams (bottom row). (A) shows the raw projection mapping result from an input video texture, (B) shows the culling result after the occlusion test, (C) shows the culling result after using dilated depth maps and majority voting algorithm, (D) shows the input mesh, (E) and (F) respectively shows the rendering results from the baseline approach and our algorithm, together with the corresponding texture weight fields for comparison.

In order to remove the ghosting effect, we adopt the method from the *Holoportation* project, which uses a dilated depth map to detect the occluded regions as shown in Figure 6.3(C), thus removing many artifacts caused by inexact geometries: At each textured point on a surface, and for each input view, we search for any depth discontinuities in its projected 2D neighborhood within a rasterized depth map with a radius determined by $\epsilon = 4$ pixels. If such a discontinuity is found, we set $\mathscr{T}^i_{\mathbf{v}} = 0$.

In addition, we also adopt the same multi-level majority voting strategy. For a given vertex $\mathbf{v}$ and texture map $\mathbf{M}_i$, we search from coarse to fine levels, the sampled color $c^i_{\mathbf{v}}$ is trusted if at least half of the visible views (we denote the number of visible views as X) agree with it in the *Lab* color space ($\delta = 0.15$):

$$\sum_{j=1, j\neq i}^{N} (|c^i_{\mathbf{v}} - c^j_{\mathbf{v}}| < \delta) \geq \left\lfloor \frac{X}{2} \right\rfloor \tag{6.3}$$

Although the dilated depth maps and multilevel majority voting strategy can mitigate most of the ghosting effects in real time (Figure 6.3(C)) the rendering results still suffer from blurring and visible seams, as shown in Figure 6.3(E).

6.4.3 Computing Misregistration and Occlusion Seams

Our algorithm identifies each triangle as a misregistration or occlusion seam when any of the following three cases occur:

1. **Self-occlusion**: One or two vertices of the triangle are occluded in the dilated depth map while the others are not.

2. **Majority voting**: The triangle vertices have different results in the majority voting process, which may be caused by either misregistration or self-occlusion.

3. **Field of View**: One or two triangle vertices lie outside the camera's field of view or in the subtracted background region while the rest are not.

Some of these examples are shown in Figure 6.4.

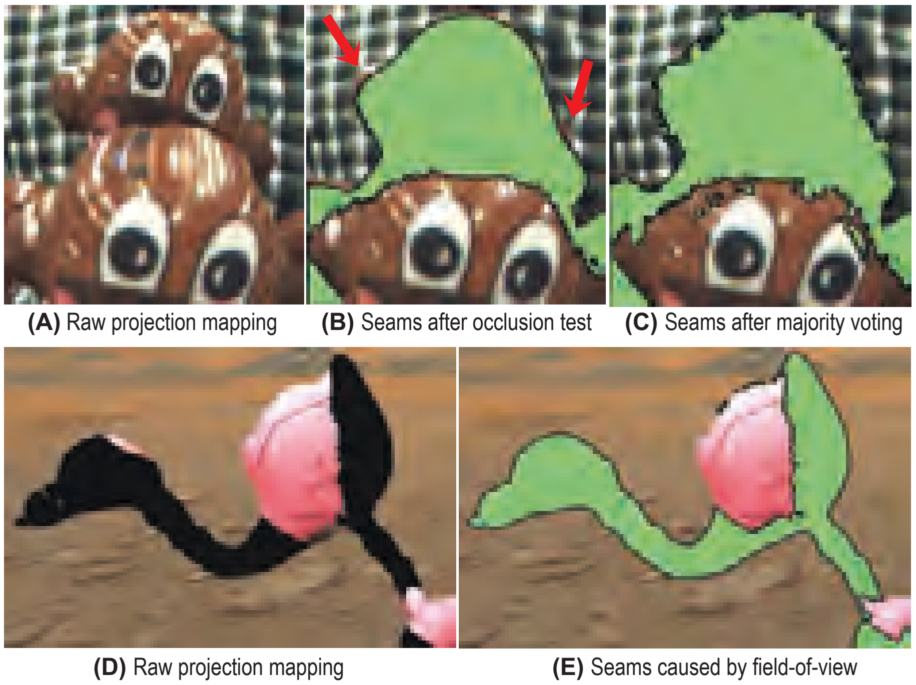

(A) Raw projection mapping
(B) Seams after occlusion test
(C) Seams after majority voting
(D) Raw projection mapping
(E) Seams caused by field-of-view

Figure 6.4: Examples of misregistration and occlusion seams. (A) shows the raw projection mapping result of a monkey toy in front of a plaid shirt, (B) shows the seams after the occlusion test with dilated depth maps, and (C) shows the seams after the majority voting test. Note that while (B) fails to remove some ghosting artifacts from the monkey toy, (C) removes most of them. (D) shows another projection onto a crane toy, (E) shows the seams identified by the field-of-view test.

For the datasets acquired for real-time telepresence applications, we have observed the fraction of seam triangles to be less than 1%. This observation has guided us to process the triangles adjacent to the seams, using a propagation procedure by calculating the geodesics directly on the GPU.

6.4.4 Discrete Straightest Geodesics for Diffusing Seams

We efficiently diffuse the texture fields using the geodesic distance fields, by making a tradeoff between accuracy and efficiency. We follow a variant of the highly efficient approximation algorithm described in [6].

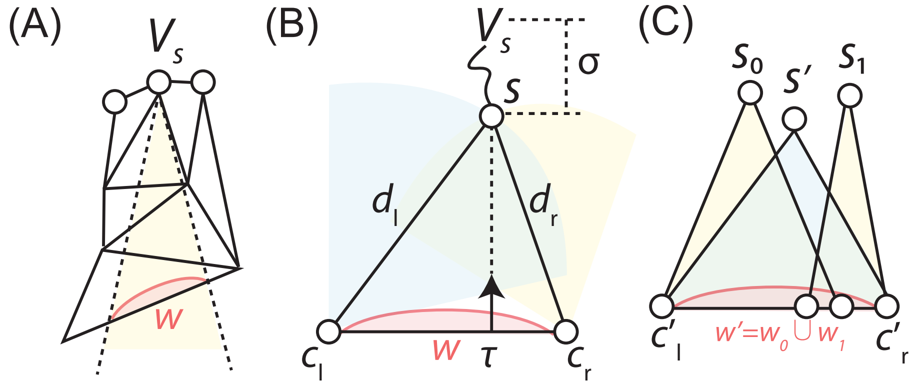


Figure 6.5: Illustration of computing the approximate geodesics. (A) shows the concept of the geodesic window from a single source vertex. (B) shows the components within a window. (C) shows the merging process of two overlapping windows for approximiation.

Let S be a piecewise planar surface defined by the triangle mesh. We define the geodesic distance function as $\mathcal{D} : S \mapsto \mathbb{R}$. In an earlier stage, we extracted the vertices from the seam triangles $V_s \in S$ as the source vertices. For any point $p \in S$, the algorithm returns the length of the geodesic path $\mathcal{D}(p)$ from p back to the closest seam vertex $v \in V_s$. We iteratively diffuse across the triangles from the seams towards the non-occluded triangles.

As illustrated in Figure 6.5, for each edge e, we maintain a small number of windows $\mathbf{w}(e)$ consisting of a pair of coordinates (c_l, c_r) (counterclockwise), the corresponding geodesic distance (d_l, d_r) to the closest pseudosource source s, the direction of the geodesic path τ, and the geodesic length $\sigma = \mathcal{D}(s)$. The position of s

can be calculated by intersecting two circles. As suggested by [6], when propagating a window $w_1(e)$ with an existing window $w_0(e)$ on the same edge, we try to merge the two windows $w' \leftarrow w_0(e) \cup w_1(e)$, if the directions τ_0, τ_2 agree with each other, and the estimated geodesic lengths are within a bounded error: $|\mathcal{D}(w_0) - \mathcal{D}(w_1)| < \varepsilon$.

In order to achieve interactive rates for rendering, we march at most $k = 15$ triangles from the seams in $K = 20$ iterations. In this propagation process, we maintain two windows per edge and discard the rest. We chose the parameter $k < K$ so that each vertex's minimum geodesic distance field could be updated from the vertices that are $K - k$ edges away. As Figure 6.6 shows, this compromise gives us visually pleasing results for diffusing the texture fields spatially near the seams.

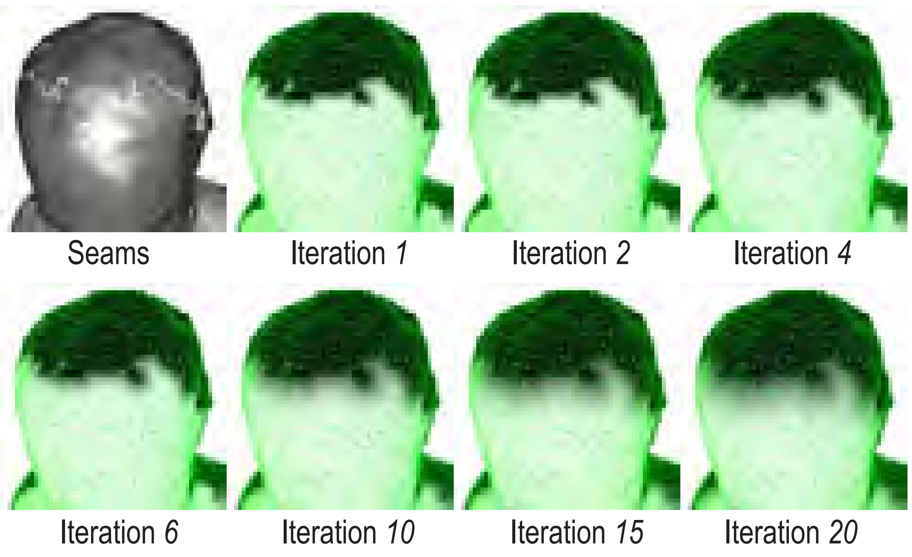


Figure 6.6: Examples of the initial seam triangles and the propagation process for updating the geodesics.

6.4.5 Temporal Texture Fields

To prevent the texture weights from changing too fast during view transitions, we use *target texture fields* and *temporal texture fields*. The target texture fields are determined using view-dependent texture weights and occlusion seams:

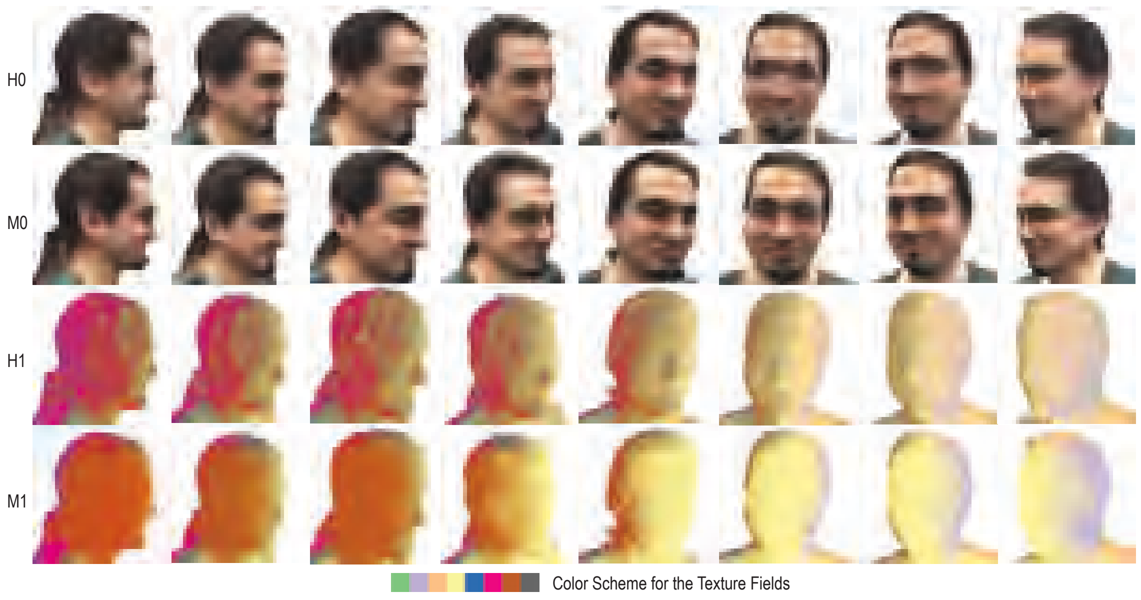


Figure 6.7: Spatiotemporal comparison of the rendered results (H0, M0) and corresponding texture fields (H1, M1) for *Holoportation* (H0, H1) and *Montage4D* (M0, M1) across 8 viewpoints and 40 frames. As shown in the figures, *Montage4D* takes advantage of view-dependent rendering while mitigating visible seams. In addition, temporal texture weights facilitate smooth transitions in space and time. Please see the accompanying video for a temporal comparison.

$$\mathscr{T}_{\mathbf{v}}^{i} = \mathscr{V}_{\mathbf{v}} \cdot g^{i} \cdot \gamma_{\mathbf{v}}^{i} \cdot \max(0, \hat{\mathbf{v}} \cdot \hat{\mathbf{v}}_{i})^{\alpha}, \tag{6.4}$$

where, $\mathscr{V}_{\mathbf{v}}$ is the original visibility test at vertex $\mathbf{v}$ with dilated depth maps and multi-level majority voting, g^{i} is a normalized global visibility score of each view, which is calculated by the number of visible vertices from each view. Therefore, g^{i} reduces weights for less significant views. $\gamma_{\mathbf{v}}^{i}$ is the minimum length of the equidistance geodesics to the seams for the texture map $\mathbf{M}_i$, $\hat{\mathbf{v}}$ is the view vector from the current user's camera to the vertex $\mathbf{v}$, $\hat{\mathbf{v}}_i$ is the view vector of the i_{th} camera, and α determines the smoothness of the transition. We use temporal texture fields to handle the temporal artifacts as follows:

$$\mathscr{T}_{\mathbf{v}}^{i}(t) = \mathscr{T}_{\mathbf{v}}^{i}(t-1) + \lambda \nabla \mathscr{T}_{\mathbf{v}}^{i}(t) \tag{6.5}$$

where, $\mathscr{T}_{\mathbf{v}}^{i}(t)$ represents the texture field at vertex $\mathbf{v}$ at frame t, λ determines the transition rate of the texture fields, and $\nabla \mathscr{T}_{\mathbf{v}}^{i}(t)$ is the smoothed gradient of the texture fields at frame t, $\lambda = 0.05$.

We normalize the texture fields and fuse the sampled colors using the Equation 6.1. For highly occluded regions, if $\sum_i \gamma_{\mathbf{v}}^{i} < 1$, we preserve the result from normal-weighted blending to fill in the black pixels. We discuss the limitations of this compromise in Section 6.7. Figure 6.7 shows comparative results between *Holoportation* and *Montage4D*. Holoportation typically mixes three or more views for each vertex in the texture field, while *Montage4D* favors the frontal views and blends the seam with the help of a geodesic distance field. In Figure 6.8, we show an example of temporal adaptation within a fraction of a second after the user quickly switches the viewpoint from the side to the front. Note that both the texture quality and color balance transition smoothly over time to avoid abrupt changes in the rendering results.

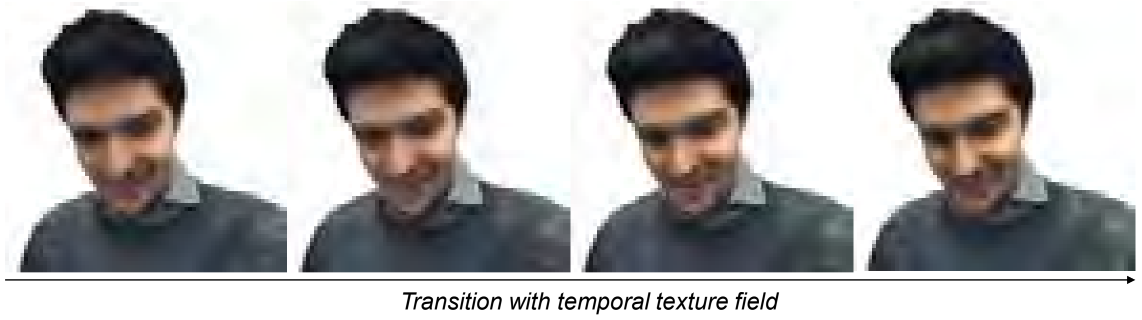


Figure 6.8: Temporal transition within half a second after an abrupt change in viewpoint. Note that texture quality gradually improves as the texture fields progressively favor the frontal view. In addition, the color balance adjusts slightly from cold to warm, due to the different settings of white balance in the video textures.

6.5 Experimental Results

6.5.1 Comparison with the Holoportation approach

We implement our rendering pipeline using the multi-pass compute, vertex, and fragment shaders with *DirectX 11*, and conduct quantitative analysis on a commodity workstation with a *GeForce GTX 1080* graphics card with 8 GB frame buffers. We evaluate our results with five recorded datasets with a *Fusion4D* program running in the background to feed the reconstructed meshes and video textures to *Montage4D*. These datasets cover a wide range of subjects, including children, adults, and air-inflated toys with specular highlights. Each dataset contains at least 500 frames, and each frame contains at least $130,000$ vertices, $250,000$ triangles, and 8 video texture maps at the resolution of 2048×2048 pixels. The average frame rate of the video textures is around 25 frames per second (FPS). The detailed statistics of the datasets are listed in Table 6.1. As in the *Holoportation* project, all incoming data is decompressed using the LZ4 algorithm prior to its ingestion in the rendering

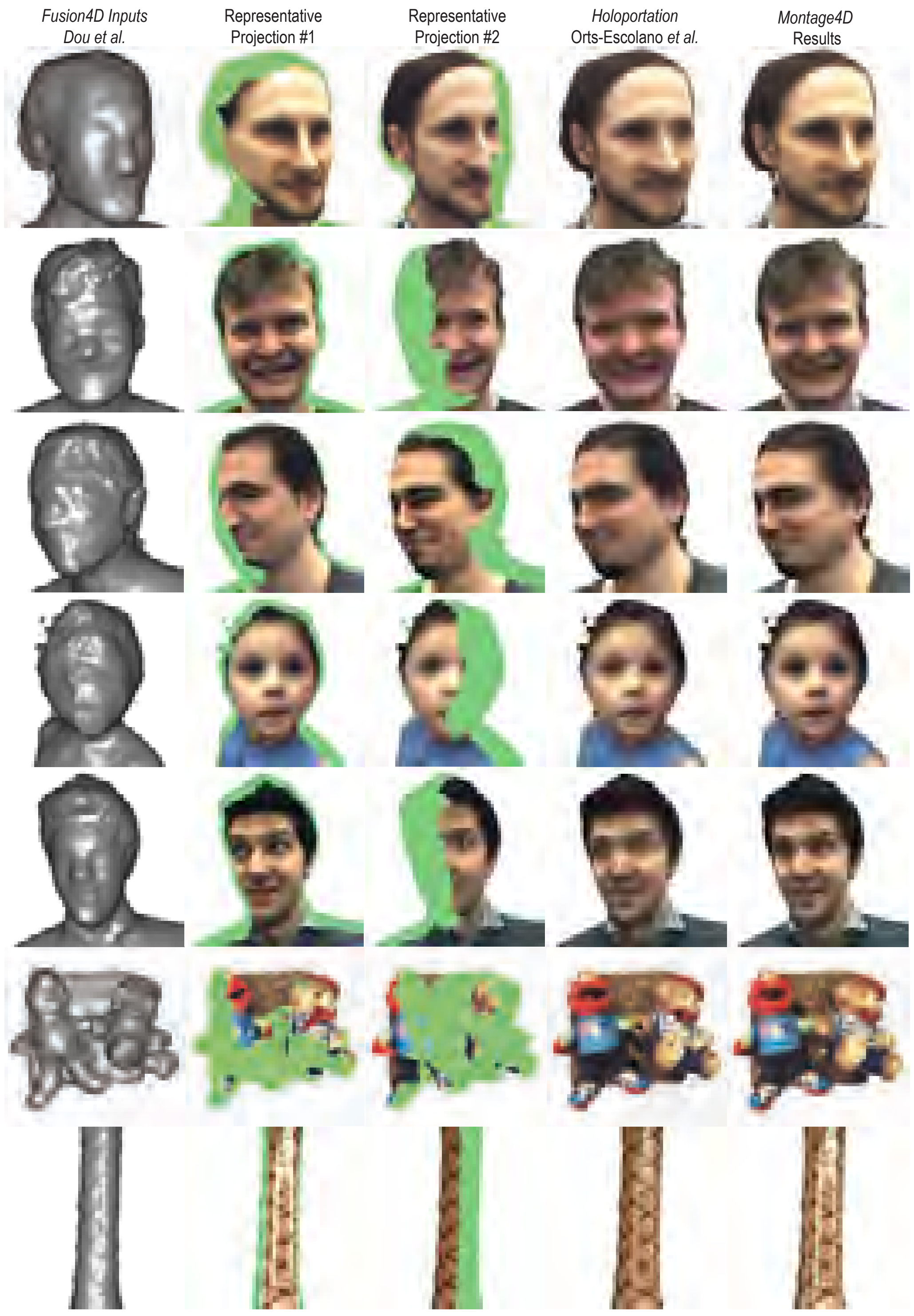

Fusion4D Inputs Dou *et al.*
Representative Projection #1
Representative Projection #2
Holoportation Orts-Escolano *et al.*
Montage4D Results
* The parameters of Fusion4D are tuned for real-time Holoportation experience, which may result in coarser meshes.

Figure 6.9: Comparison with the Holoportation approach. From left to right: the input mesh generated by Fusion4D, two representative back-projected images, and the rendering results from Holoportation and our Montage4D system.

pipeline.

First, we conduct a cross-validation experiment over the five datasets between the ground truth image from each camera's perspective and the rendering results of the *Holoportation* or *Montage4D* renderer. We demonstrate the quantitative results using the average of the root mean square error (RMSE) of the RGB color values, peak signal-to-noise ratio (PSNR), and the structural similarity (SSIM) [213][2]. The results are shown in Table 6.2. We can see that *Montage4D* achieves higher image quality (lower RMSE, higher SSIM and PSNR) while maintaining interactive frame rates for virtual reality applications.

Next, we visually compare the quality and sharpness of the rendered images from novel views, as illustrated in Figure 5.11. We also show the input meshes and representative back-projected images. Although the approach taken in the *Holoportation* project is able to render textured meshes smoothly and eliminates most of the ghosting artifacts, it often fails to preserve the fine details such as human faces. In contrast, the *Montage4D* renderer preserves the details from the dominating views using view-dependent texture weights and transitions smoothly using the temporal texture fields. Meanwhile, the diffusion process in *Montage4D* is able to remove most of the misregistration and occlusion seams that occur in the representative back-projections.

Additionally, we use the *Unity profiler* to analyze and compare the timing for a typical frame of the *Sergio* dataset. As shown in Table 6.3, the main difference

[2]A toolkit for comparing image quality: `https://github.com/ruofeidu/ImageQualityCompare`.

between the two approaches is data transfer time between CPU and GPU. In addition to copying buffers for vertex indices and positions, the *Montage4D* system also transfers compute buffers for geodesics, texture fields, and seam factors, which induces a small overhead over the original approach. However, dispatching the diffusion kernels does not impact the frame rate much and the overall timing is still satisfactory for interactive applications.

6.5.2 Comparison with the Floating Textures Approach

We further compare our rendering results with two other blending algorithms: Filtered Blending [214] and Floating Textures [200][3].

We compiled and ran the implementations of the above approaches with OpenGL 4.5 using an *Nvidia GTX 1080* graphics card. Since their implementations are offline approaches taking several seconds per frame to load the models and textures into the GPU memory, we only compare the rendering results for static frames visually in Figure 6.10. We use a black background since their optical flow algorithm assumes that black represents motionless pixels.

The Filtered Blending algorithm runs smoothly at over 60 FPS, while the Floating Textures algorithm runs at between 7 and 22 FPS, depending on the target resolution ($1K \times 1K$ to $2K \times 2K$) and number of iterations within the optical flow computation (we use the default parameters, 30 iterations).

As shown in Figure 6.10, the Filtered Blending algorithm produces ghosting effects on human faces. The Floating Textures algorithm improves results signifi-

[3]The source code is available at: https://sourceforge.net/projects/floatingtexture/

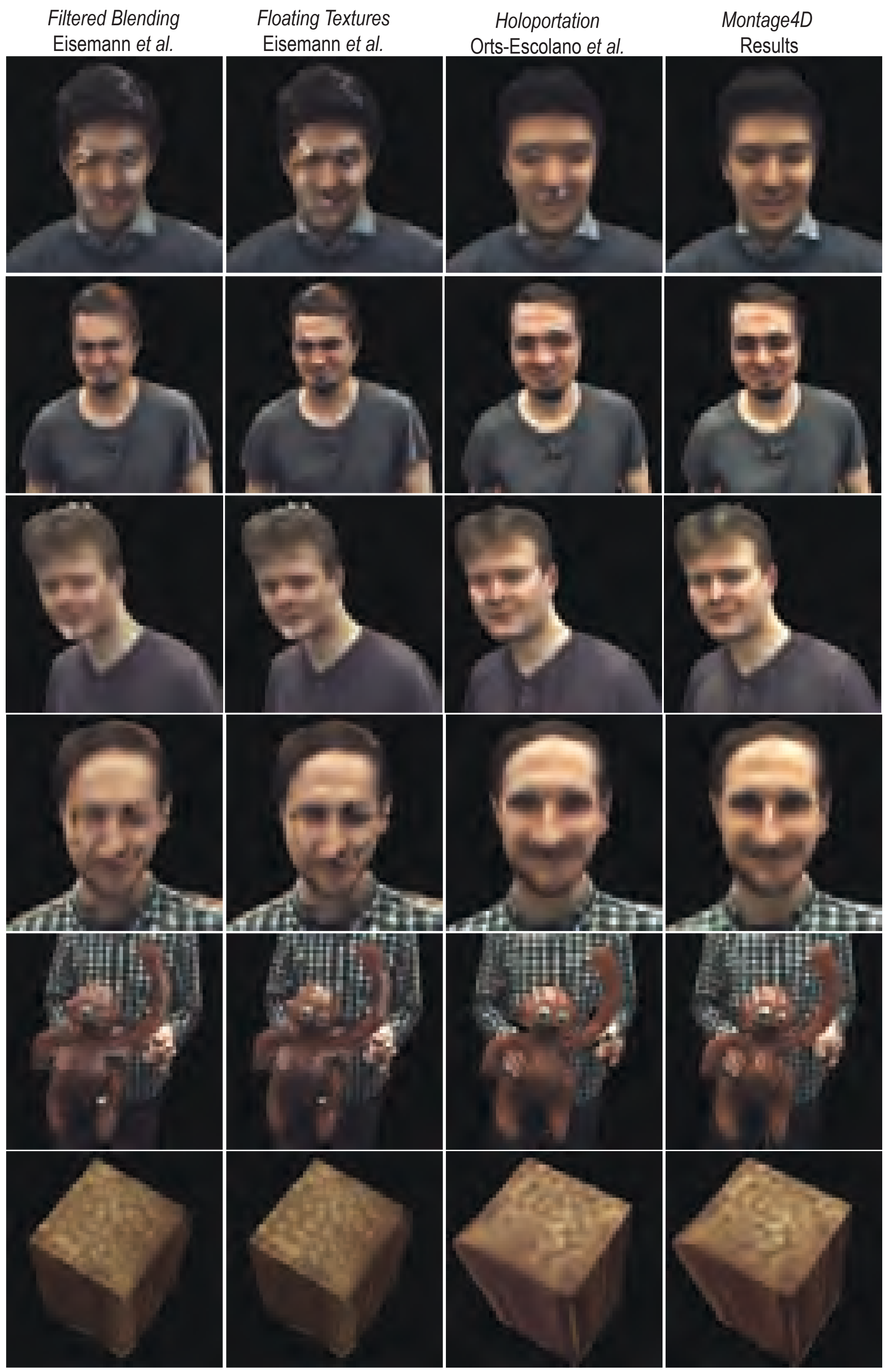


Figure 6.10: Comparison of different texturing algorithms. From left to right: Filtered Blending, Floating Textures, Holoportation, and our Montage4D system.

cantly by using optical flow to address screen-space misalignment. While an elegant approach, it sometimes fails to produce visually convincing results due to several reasons:

First, the optical flow in Floating Textures is calculated in screen space instead of across the 3D mesh. As a result, view morphing leads to visual appearance errors. Second, the algorithm does not employ dilated depth maps, color voting strategies, or seam diffusion in the 3D space (the soft visibility map used by Floating Textures is a diffusion process in screen space), thereby introducing artifacts caused by inaccurate geometry. Finally, the algorithm does not take temporal coherence into account. Consequently, appearance can change significantly with an abrupt change in viewpoint. In contrast, we compute the target texture fields for each vertex, and use temporal texture fields to update these, making the temporal animation smooth.

Table 6.1: Statistics of the datasets, including the average number of vertices and triangles per frame, the total size of the datasets (compressed with the LZ4 algorithm), and the total length of the multiview videos. Note that the average frame rate of the input videos ranges from 21 to 28 FPS, depending on the bandwidth.

dataset	#frames	#vertices/ rame	#triangles/frame	size(GB)	duration(s)
Timo	837	131K	251K	5.29	33.6
Yury	803	132K	312K	5.29	32.6
Sergio	837	215K	404K	6.96	39.8
Girl	1192	173K	367K	7.43	42.4
Julien	599	157K	339K	4.52	25.5

6.6 Real-time Stylization

Image and video filters have been widely applied in social media platforms such as Instagram and Facebook. We envision that real-time stylization may be

Table 6.2: Comparison between *Holoportation* and *Montage4D* in cross-validation experiments. Note that *Montage4D* outperforms the Holoportation approach while maintaining an average frame rate of over 100 Hz, which exceeds the maximum refresh rate of the current generation of VR headsets such as Oculus Rift.

Dataset	*Holoporation*				*Montage4D*			
	RMSE	PSNR	SSIM	FPS	RMSE	PSNR	SSIM	FPS
Timo	5.63%	0.9805	227.2	38.60dB	3.27%	0.9905	135.0	40.23dB
Yury	5.44%	0.9695	222.8	39.20dB	3.01%	0.9826	130.5	40.52dB
Sergio	7.74%	0.9704	186.8	29.84dB	4.21%	0.9813	114.3	30.09dB
Girl	7.16%	0.9691	212.56	36.28dB	3.73%	0.9864	119.4	36.73dB
Julien	12.63%	0.9511	215.18	33.94dB	6.71%	0.9697	120.6	35.05dB

Table 6.3: Timing comparison between *Holoportation* and *Montage4D* for a new frame of geometry. Geometry and textures are streamed at around 21 fps.

Procedure	Timing (ms)	
	Holoportation	Montage4D
Communication between CPU and GPU	4.83	9.49
Rendering and Texture Sampling	0.11	0.30
Rasterized Depth Maps calculation	0.14	0.13
Seams Identification	N/A	0.01
Approximate Geodesics estimation	N/A	0.31
Other events	0.12	0.18
Total	5.11	10.40

a popular component of future live reconstruction platforms. In this section, we introduce two real-time stylization methods which enhance visual quality: sketchy stippling and relighting.

6.6.1 Sketchy Stippling

Stippling is an artistic style which uses small dots for drawing a picture. The density of the dots naturally corresponds to the intensity of the image. Previous stip-

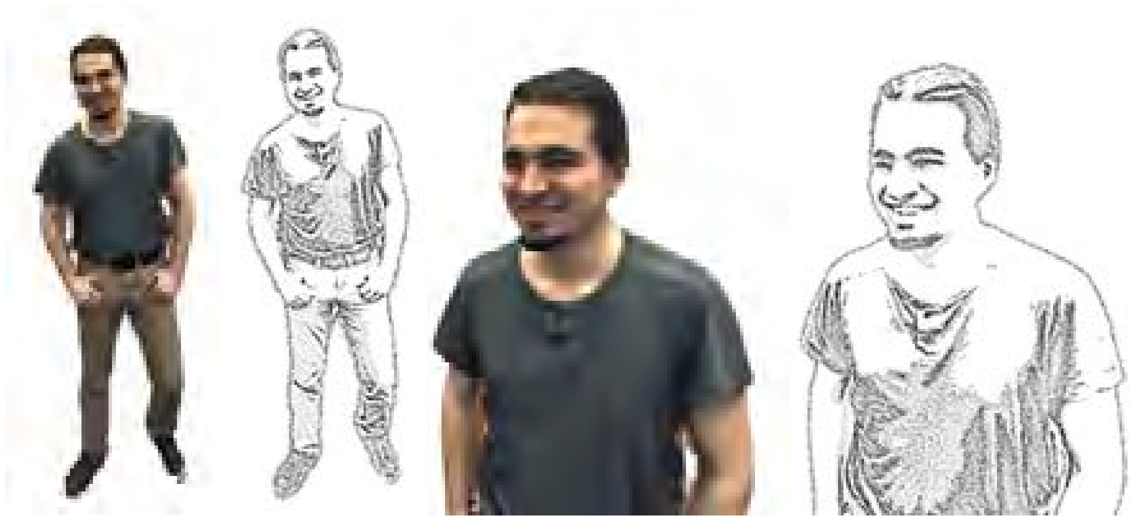


Figure 6.11: Results before and after applying the sketchy stippling effects.

pling methods have employed weighted centroidal Voronoi diagrams [215], Voronoi relaxation [216], or feature-guided approaches [217]. However, prior arts typically require multiples passes or hardly run in real time at a high resolution. Here, we present a real-time sketchy stippling approach, which combines both sketchy and stippling styles with only two texture lookups. Our approach only requires a single screen-space postprocessing pass. We show two examples of sketchy stippling results in Figure 6.11.

Our sketchy stippling shader consists of four steps, as shown in Figure 6.12. We open source the code with detailed comments and present a live demo at ShaderToy: `https://www.shadertoy.com/view/ldSyzV`.

1. Set up the input buffer with the output from the *Montage4D* pipeline (the first texture lookup).

2. Generate a dotted blurred image of the input buffer, then invert its color space. We sample the stipples randomly based on the screen coordinates with

the second texture lookup.

3. Composite the intermediate results by computing the color dodge between the input buffer and the dotted blurred image. This step preserves the stippling dots in the previous step and shades the dark regions with sketchy effects.

4. Compute the grayscale image and boost the contrast via power functions.

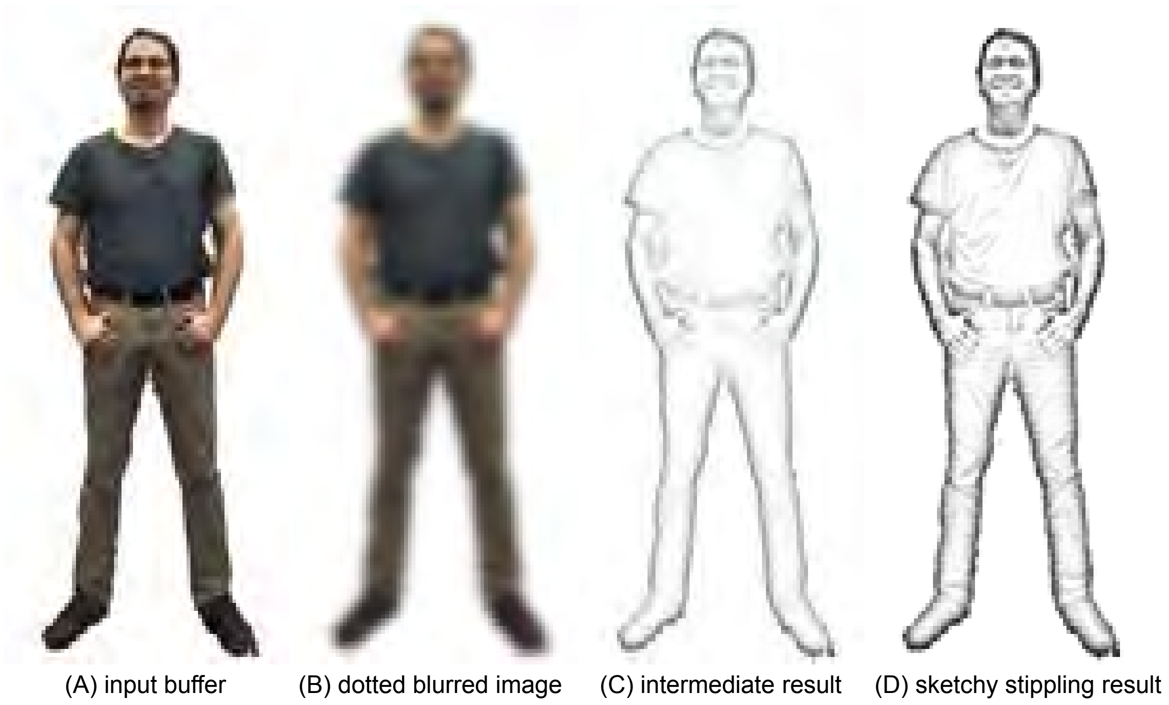


Figure 6.12: The step-by-step results of our sketchy stippling stylization approach with two texture lookups (one for the original buffer, the other for the blurred image).

6.6.2 Relighting

When applying *Montage4D* to live telepresence, it is usually preferred that the lighting condition of the 3D models agrees with the virtual environments. Here we present our initial efforts for relighting live streamed 3D models.

First, we convert the normal vector $\hat{\mathbf{n}}$ per vertex to the normal vector in the view-space $\hat{\mathbf{n}}_{xy}$, where $\mathbf{V}^{-1}\mathbf{M}^{-1}$ is the inverse model view matrix:

$$\hat{\mathbf{n}}_{xy} = \mathbf{V}^{-1}\mathbf{M}^{-1}\hat{\mathbf{n}} \tag{6.6}$$

Next, we sample the light L from a precomputed light probe buffer P, as shown in Figure 6.13.

$$\mathbf{L} = \text{texture}\left(P, \hat{\mathbf{n}}_{xy}\right) \tag{6.7}$$

Finally, for each pixel, we use a non-linear blending approach (Equation 6.8) to mix the final color $\mathbf{F}$ with the light color $\mathbf{L}$ and the texture color $\mathbf{T}$. Let $C_{\mathbf{X}}$ be one of the red, green, and blue channels in color $\mathbf{X}$. We lighten the model if $C_{\mathbf{L}}$ has a high intensity and darken the model if $C_{\mathbf{L}}$ has a low intensity. The resulting color is unaffected if $C_{\mathbf{L}}$ is of 50% intensity. We use a power function to lighten or darken the model. A simple demo is presented at `https://www.shadertoy.com/view/XldcDM`.

$$C_{\mathbf{F}} = C_{\mathbf{T}} + |1 - 2C_{\mathbf{L}}|\left(C_{\mathbf{T}}^{1.5 - C_{\mathbf{L}}} - C_{\mathbf{T}}\right), C \in \{\text{Red, Green, Blue}\} \tag{6.8}$$

For future work, if the intrinsic color of each vertex could be computed in real time, a simple multiplication of the reflectance and shading will composite the resulting color. One may take advantage of intrinsic video [218] and material estimation [219] to achieve this.

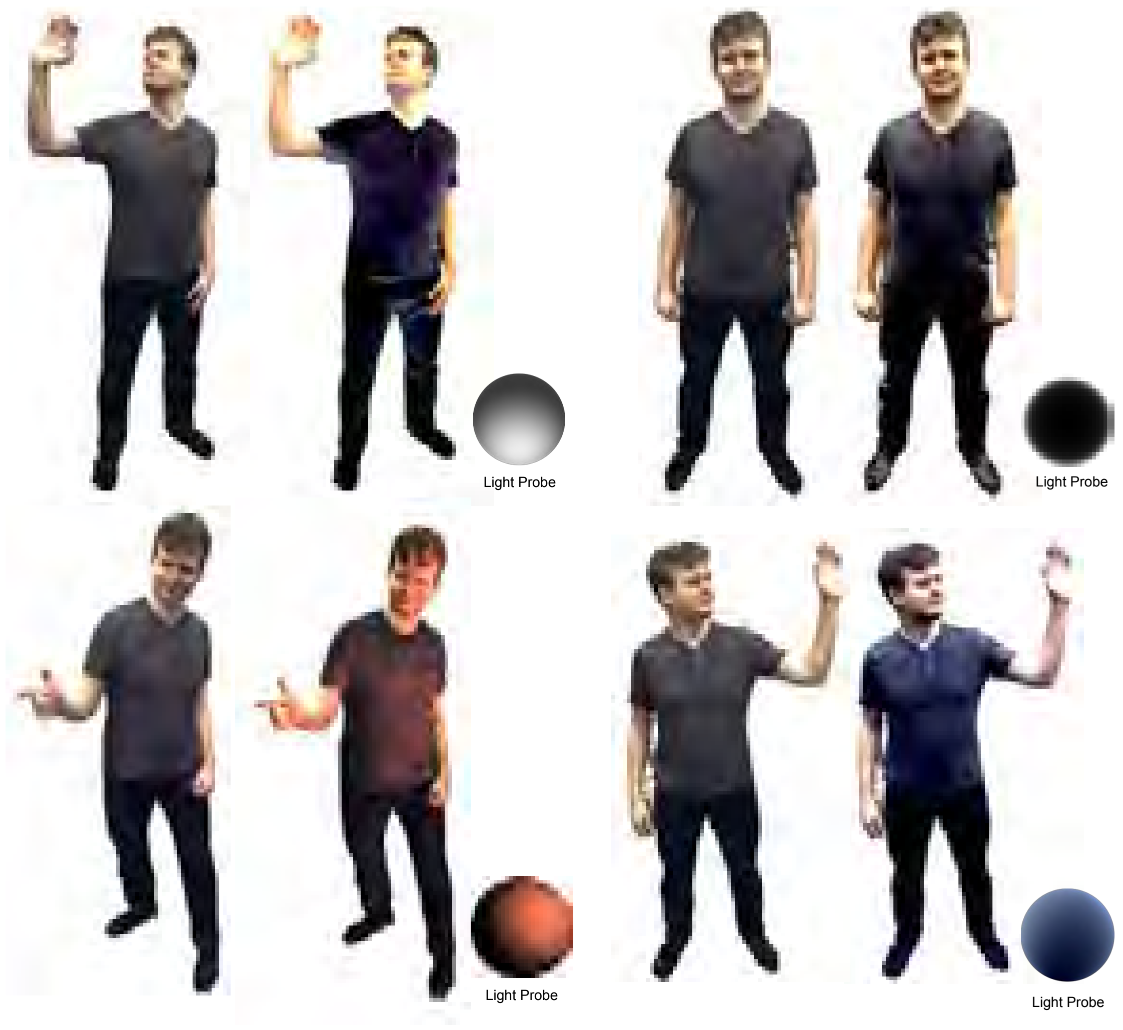


Figure 6.13: Results before and after the interactive relighting pass with different light probes. Note that the black trousers and shoes are lightened with more details in the top row, while the shirts are lightened with different colors in the bottom row.

6.7 Limitations

Figure 6.14 presents limitations of our approach. The top row shows the artifacts resulting from the extruded triangles reconstructed during very fast motion. It should be possible to use a remeshing algorithm to tackle such problems. The bottom row shows the challenging issue caused by insufficient reliable colors. Such problems may be solved by user-guided in-painting and seamless cloning, which are proposed in the *TextureMontage* system [197]. However, for interactive applications, it will be ideal if one could achieve such interpolation with minimal latency without the user's intervention.

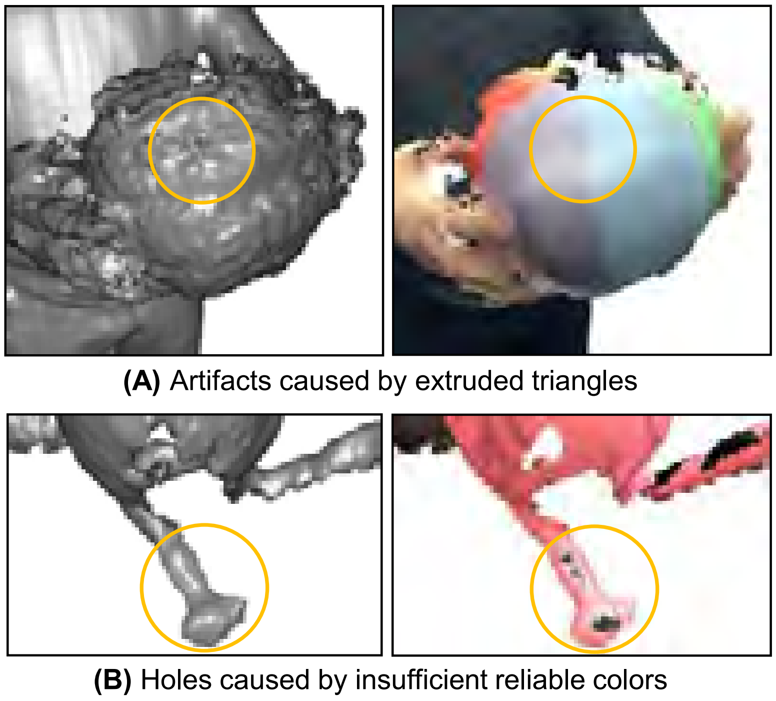

(A) Artifacts caused by extruded triangles
(B) Holes caused by insufficient reliable colors

Figure 6.14: Limitations of our approach. Extruded triangles and highly-occluded spots may still cause artifacts.

6.8 Conclusion and Future Work

In this chapter, we have presented *Montage4D*, an interactive and real-time solution to blend multiple video textures onto dynamic meshes with nearly indiscernible view transitions. We improve on previous *Holoportation* renderer by adopting view-dependent rendering, seam identification, geodesic distance fields based diffusion, and smooth transition using temporal texture fields. Our technique offers sharper images than previous interactive texturing algorithms, allowing users to observe fine facial expressions for immersive telepresence and communication.

In the future, we would like to integrate the *Montage4D* texturing pipeline with the cloud-based scene acquisition server. By incorporating the user's view directions, the acquisition server could progressively synthesize a compact view-dependent video texture atlas directly on the client side, thus greatly reducing the bandwidth requirement. Although we have figured out many problems with the screen-based optical flow approach, such as surface specularity and poor color calibration, we would like to investigate adaptive and efficient optical flow algorithms over the mesh surface [177]. We envision our algorithm to be useful for virtual and augmented reality applications, such as remote business meetings, medical training, and live social events.

Chapter 7: Conclusion and Future Directions

In this dissertation, we have presented our research to fuse multimedia data, including text, images, 360° panoramas, and multiview videos, into dynamic virtual environments.

In *Social Street View* [1], we present the first real-time system that blends geotagged social media with immersive maps. We have developed techniques to carefully lay out and display social media on virtual billboards by a judicious combination of depth maps, normal maps, and maximal Poisson sampling. We further validate the efficiency of such mixed-reality visualizations for saliency coverage in the immersive maps. To obtain an overview of the topics in geotagged social media, we present *TopicFields* to facilitate deep learning models for fusing and visualizing multiple scalar fields on the map.

To avoid placing information over salient objects such as pedestrians and building entrances, we devise a spherical saliency residual model to measure the visual saliency in 360° videos on the GPU. Our model outperforms the classic Itti *et al.*'s model by 5× to 13× in timings and remains consistent in challenging cases such as horizontal clipping and spherical rotations. We further develop a spatiotemporal model for virtual cinematography to ensure large saliency coverage while reducing the camera movement jitter.

The successor of *Social Street View*, *Geollery*, progressively reconstructs a mirrored world and fuses social media, panoramas, 3D buildings, and virtual avatars in real time. We qualitatively evaluate both systems in a formal user study with semi-structured interviews. The user responses have emphasized the significance of interactive capabilities and revealed several applications: travel planning, the local discovery of food and events, virtual meetings, and social gatherings.

Furthermore, we present *Video Fields* [90] to validate the concept of fusing multi-view videos with 3D environments in real time. Its successor, *Montage4D* [220], offers a practical solution towards real-time, seamless fusion of multiview video textures. We use geodesics on meshes, view-dependent rendering, and temporal smoothing to mitigate spatial occlusion seams. Our algorithms run in real time and significantly improve the original *Holoportation* approach. Furthermore, we present our preliminary exploration towards real-time stylization and relighting to empower the Holoportation users to interactively stylize live 3D content. Our pipeline could be further accelerated with kernel foveated rendering [221].

Currently, weak content is identified as one of the major global market restraints for the adoption of VR and AR technology. We believe that our solutions may inspire a wide range of applications including mixed-reality social platforms, immersive surveillance, recurrence of past events, telepresence for business meetings, family gatherings, and remote education.

In future, we envision a system that fuses a variety of multimedia data to create a vivid mirrored world, in which it fuses past events (geotagged images, text messages, and videos) with the present (3D buildings with immersive street views, live surveillance videos, and the user's telepresence), and looks into the future by learning the trajectory of pedestrians and cars, clustering the topics and emotions of social media, and measuring the spatiotemporal saliency of the real-time information in the world. Such a system may eventually change the way we communicate with each other and consume the information throughout the world.

List of Peer-reviewed Publications

[1] **Ruofei Du**, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4D: Interactive Seamless Fusion of Multiview Video Textures. *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, pp. 124–133, 2018.

[2] **Ruofei Du**, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4D: Real-time Seamless Fusion and Stylization of Multiview Video Textures. *Journal of Computer Graphics Techniques*, 1(8), pp. 1–39, 2019.

[3] **Ruofei Du**, David Li, and Amitabh Varshney. Geollery: Designing an Interactive Mirrored World with Geo-tagged Social Media. *ACM CHI Conference on Human Factors in Computing Systems*, 2019.

[4] **Ruofei Du**, David Li, and Amitabh Varshney. Experiencing a Mirrored World With Geotagged Social Media in Geollery. *Extended Abstracts of ACM CHI Conference on Human Factors in Computing Systems*, 2019.

[5] **Ruofei Du**, Eric Lee, and Amitabh Varshney. Tracking-Tolerent Visual Cryptography. *The 26th IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[6] **Ruofei Du**, David Li, and Amitabh Varshney. Interactive Fusion of 360° Images for a Mirrored World. *The 26th IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[7] **Ruofei Du** and Amitabh Varshney. Spherical Harmonics for Saliency Computation and Virtual Cinematography in 360° Videos. [**Best Student Poster Award**] at *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D), 2018. Under Review, IEEE Transactions on Visualization and Computer Graphics*, 2018.

[8] **Ruofei Du** and Amitabh Varshney Social Street View: Blending Immersive Street Views with Geo-Tagged Social Media. [**Best Paper Award**] In *Proceedings of the 21st International Conference on Web3D Technology*, pp. 77–85, 2016.

[9] Xiaoxu Meng, **Ruofei Du**, Matthias Zwicker, and Amitabh Varshney. Kernel Foveated Rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(5), pp. 1–20, 2018.

[10] Changqing Zou, Qian Yu, **Ruofei Du**, Haoran Mo, Yi-Zhe Song, Tao Xiang, Chengying Gao, Baoquan Chen, and Hao Zhang. SketchyScene: Richly-Annotated Scene Sketches. In *European Conference of Computer Vision (ECCV 2018)*, pp. 421–436.

[11] **Ruofei Du**, Sujal Bista, and Amitabh Varshney. Video Fields: Fusing Multiple Surveillance Videos into a Dynamic Virtual Environment. *Proceedings of the 21st International Conference on Web3D Technology*, pp. 165–172, 2016.

[12] **Ruofei Du** and Liang He. VRSurus: Enhancing Interactivity and Tangibility of Puppets in Virtual Reality *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pp. 2454–2461, 2016.

[13] **Ruofei Du**, Kent Wills, Max Potasznik, and Jon Froehlich. AtmoSPHERE: Representing Space and Movement Using Sand Traces in an Interactive Zen Garden *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 1627–1632, 2016.

[14] Lee Stearns, **Ruofei Du**, Uran Oh, Catherine Jou, Leah Findlater, David A. Ross, Rama Chellappa, and Jon E. Froehlich Evaluating Haptic and Auditory Directional Guidance to Assist Blind People in Reading Printed Text Using Finger-Mounted Cameras *ACM Transactions on Accessible Computing (TACCESS)*, 9(1):1, 2014.

[15] Changqing Zou, Haoran Mo, **Ruofei Du**, Xing Wu, Chengying Gao, and Hongbo Fu. LUCCS: Language-based User-customized Colorization of Scene Sketches. *arXiv preprint*, arXiv:1808.10544, *Under Review. ACM Transaction on Graphics*, 2018.

[16] Leah Findlater, Lee Stearns, **Ruofei Du**, Uran Oh, David Ross, Rama Chellappa, and Jon E. Froehlich Supporting Everyday Activities for Persons with Visual Impairments Through Computer Vision-Augmented Touch *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS 2015)*, pp. 383–384, 2015.

[17] Lee Stearns, **Ruofei Du**, Uran Oh, Yumeng Wang, Leah Findlater, Rama Chellappa, and Jon E. Froehlich The Design and Preliminary Evaluation of a Finger-Mounted Camera and Feedback System to Enable Reading of Printed Text for the Blind *European Conference on Computer Vision (ECCV), Workshop on Assistive Computer Vision and Robotics*, pp. 615–631, 2014.

References

[1] Ruofei Du and Amitabh Varshney. Social street view: Blending immersive street views with geo-tagged social media. In *Proceedings of the 21st International Conference on Web3D Technology*, Web3D, pages 77–85. ACM, July 2016. doi:10.1145/2945292.2945294.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv Preprint ArXiv:1301.3781*, 3. URL `https://arxiv.org/pdf/1301.3781`.

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 6.

[4] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: an Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. doi:10.1145/1656274.1656278.

[5] Sarah J Andrabi, Michael K Reiter, and Cynthia Sturton. Usability of Augmented Reality for Revealing Secret Messages to Users But Not Their Devices. In *SOUPS*, volume 2015, pages 89–102, 5.

[6] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. Fast Exact and Approximate Geodesics on Meshes. *ACM Transactions on Graphics (TOG)*, 24(3):553–560, 2005. doi:10.1145/1186822.1073228.

[7] Andreas M. Kaplan and Michael Haenlein. Users of the World, Unite! the Challenges and Opportunities of Social Media. *Business Horizons*, 53(1): 59–68, 0. ISSN 00076813. doi:10.1016/j.bushor.2009.09.003. URL `http://linkinghub.elsevier.com/retrieve/pii/S0007681309001232`.

[8] Benjamin E Teitler, Michael D Lieberman, Daniele Panozzo, Jagan Sankaranarayanan, Hanan Samet, and Jon Sperling. NewsStand: a New View on

News. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '08*, pages 1–10. ACM Press, 8.

[9] Pavel Serdyukov, Vanessa Murdock, and Roelof Van Zwol. Placing Flickr Photos on a Map. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 484–491, 2009. doi:10.1145/1571941.1572025.

[10] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google Street View: Capturing the World at Street Level. *Computer*, 43(6):32–38, 0. ISSN 0018-9162. doi:10.1109/MC.2010.170.

[11] A Torii, M Havlena, and T Pajdla. From Google Street View to 3D City Models. *Proceedings of 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2188–2195, 9. doi:10.1109/ICCVW.2009.5457551. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5457551`.

[12] Hanan Samet, Marco D Adelfio, Brendan C Fruin, Michael D Lieberman, and Jagan Sankaranarayanan. PhotoStand: a Map Query Interface for a Database of News Photos. *Proceedings of the VLDB Endowment*, 6(12):1350–1353, 2013. doi:2536274.

[13] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. TwitterStand: News in Tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 42. ACM Press, 9. ISBN 9781605586496. doi:10.1145/1653771.1653781. URL `http://portal.acm.org/citation.cfm?doid=1653771.1653781`.

[14] Blaine Bell, Steven Feiner, and Tobias Höllerer. View Management for Virtual and Augmented Reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology - UIST '01*, pages 101–110, 2001. doi:10.1145/502348.502363.

[15] Robert Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney. Social Snapshot: a System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications*, 31(1):74–84, 2011. doi:10.1109/MCG.2010.107.

[16] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics (TOG)*, 25(3): 835–846, 2006. doi:10.1145/1179352.1141964.

[17] Bc Russell and R Martin-Brualla. 3D Wikipedia: Using Online Text to Automatically Label and Navigate Reconstructed Geometry. *ACM*

Transactions on Graphics (TOG), 32(6):1–10, 3. ISSN 07300301. doi:10.1145/2508363.2508425.

[18] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the World From Internet Photo Collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. doi:10.1007/s11263.

[19] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Sietz, and Rick Szeliski. Building Rome in a Day. In *2009 IEEE 12th International Conference on Computer Vision*, CVPR, pages 72–79. IEEE, September 2009. doi:10.1145/2001269.2001293.

[20] Akihiko Torii, Michal Havlena, and Tomáš Pajdla. From Google Street View to 3D City Models. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 2188–2195. IEEE, 2009. doi:10.1109/ICCVW.2009.5457551.

[21] Jianxiong Xiao, Tian Fang, Peng Zhao, Maxime Lhuillier, and Long Quan. Image-Based Street-Side City Modeling. *ACM Transactions on Graphics (TOG)*, 28(5):114:1–114:12, 2009. doi:10.1145/1661412.1618460.

[22] Abdullah Bulbul and Rozenn Dahyot. Social Media Based 3D Visual Popularity. *Computers & Graphics*, 63:28–36, 2017. doi:10.1016/j.cag.2017.01.005.

[23] Hannu Kukka, Minna Pakanen, Mahmoud Badri, and Timo Ojala. Immersive Street-Level Social Media in the 3D Virtual City: Anticipated User Experience and Conceptual Development. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW, pages 2422–2435. ACM, 2017. doi:10.1145/2998181.2998341.

[24] Mahmoud Badri, Minna Pakanen, Paula Alavesa, Hannu Kukka, and Timo Ojala. Design, Development, and Usability Evaluation of a System for Adding and Editing Social Media Banners in the Immersive Street-Level 3D Virtual City. In *2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pages 102–108, Sept 2017. doi:10.1109/VS-GAMES.2017.8056577.

[25] Toru Ishida. Digital City Kyoto. *Communications of the ACM*, 45(7):76–81, 2002. ISSN 0001-0782. doi:10.1007/1140754_8.

[26] Nick Yee. The Demographics, Motivations, and Derived Experiences of Users of Massively Multi-User Online Graphical Environments. *Presence: Teleoperators and Virtual Environments*, 15(3):309–329, 2006. doi:10.1007/1-4020-3898-_9.

[27] O'Sullivan Carol and Cathy Ennis. Metropolis: Multisensory Simulation of a Populated City. In *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, pages 1–7. IEEE, 2011. doi:10.1109/VS.

[28] Erik Palmquist and Jonathan Shaw. Collaborative City Modeling. *Architecture in Computro*, pages 249–256, 8. doi:10.1007/978-3-642-15690-_9.

[29] Matthias Uden and Alexander Zipf. Open Building Models: Towards a Platform for Crowdsourcing Virtual 3D Cities. In *Progress and New Trends in 3D Geoinformation Sciences*, pages 299–314. Springer, 2013. doi:10.1007/978-3-642-29793-_17.

[30] Marcus Goetz and Alexander Zipf. The Evolution of Geo-Crowdsourcing: Bringing Volunteered Geographic Information to the Third Dimension. In *Crowdsourcing Geographic Knowledge: Volunteered Geographic Information (VGI) in Theory and Practice*, pages 139–159. Springer, 2013. doi:10.1007/978-94-007-4587-_9.

[31] Toni Alatalo, Timo Koskela, Matti Pouke, Paula Alavesa, and Timo Ojala. VirtualOulu: Collaborative, Immersive and Extensible 3D City Model on the Web. In *Proceedings of the 21st International Conference on Web3D Technology*, Web3D, pages 95–103. ACM, 2016. doi:10.1145/2945292.2945305.

[32] Dennis Zielstra and Hartwig H. Hochmair. Positional Accuracy Analysis of Flickr and Panoramio Images for Selected World Regions. *Journal of Spatial Science*, 58(954):251–273, 3. ISSN 1449-8596. doi:10.1080/14498596.2013.801331. URL `http://ezproxy.utwente.nl:2183/doi/full/10.1080/14498596.2013.801331`.

[33] Alan M MacEachren, Francis P Boscoe, Daniel Haug, and Linda W Pickle. Geographic Visualization: Designing Manipulable Maps for Exploring Temporally Varying Georeferenced Statistics. In *Proceedings of IEEE Symposium on Information Visualization*, pages 87–94, 1998. doi:721081.

[34] Alan M MacEachren, Anuj Jaiswal, Anthony C Robinson, Scott Pezanowski, Alexander Savelyev, Prasenjit Mitra, Xiao Zhang, and Justine Blanford. Senseplace2: Geotwitter Analytics Support for Situational Awareness. In *2011 IEEE Conference on Visual Analytics Science and Technology*, VAST, pages 181–190, 2011. doi:10.1109/VAST.2011.6102456.

[35] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S Ebert, and Thomas Ertl. Spatiotemporal Social Media Analytics for Abnormal Event Detection and Examination Using Seasonal-Trend Decomposition. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 143–152. IEEE, 2012. doi:10.1109/VAST.2012.6400557.

[36] Anthony Stefanidis, Andrew Crooks, and Jacek Radzikowski. Harvesting Ambient Geospatial Information From Social Media Feeds. *GeoJournal*, 78(2): 319–338, 2013. doi:10.1007/s10708-011-9438-2.

[37] Ross Maciejewski, Stephen Rudolph, Ryan Hafen, Ahmad Abusalah, Mohamed Yakout, Mourad Ouzzani, William S Cleveland, Shaun J Grannis, and

David S Ebert. A Visual Analytics Approach to Understanding Spatiotemporal Hotspots. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):205–220, 2010. doi:10.1145/2820783.2820817.

[38] Siming Chen, Xiaoru Yuan, Zhenhuang Wang, Cong Guo, Jie Liang, Zuchao Wang, Xiaolong Luke Zhang, and Jiawan Zhang. Interactive Visual Discovering of Movement Patterns From Sparsely Sampled Geo-Tagged Social Media Data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1): 270–279, 2016. doi:10.1109/TVCG.2015.2467619.

[39] Yafeng Lu, Xia Hu, Feng Wang, Shamanth Kumar, Huan Liu, and Ross Maciejewski. Visualizing Social Media Sentiment in Disaster Scenarios. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1211–1215, 05 2015. doi:10.1145/2740908.2741720.

[40] Ming C Hao, Christian Rohrdantz, Halldor Janetzko, Daniel A Keim, Umeshwar Dayal, Lars Erik Haug, Meichun Hsu, and Florian Stoffel. Visual Sentiment Analysis of Customer Feedback Streams Using Geo-Temporal Term Associations. *Information Visualization*, 12(3-4):273–290, 2013. doi:10.1177/1473871613481691.

[41] Arno Scharl, Alexander Hubmann-Haidvogel, Albert Weichselbraun, Gerhard Wohlgenannt, Heinz-Peter Lang, and Marta Sabou. Extraction and Interactive Exploration of Knowledge From Aggregated News and Social Media Content. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 163–168. ACM, 2012. doi:10.1145/2305484.2305511.

[42] Seokyeon Kim, Seongmin Jeong, Insoo Woo, Yun Jang, Ross Maciejewski, and David S Ebert. Data Flow Analysis and Visualization for Spatiotemporal Statistical Data Without Trajectory Information. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1287–1300, March 2018. ISSN 1077-2626. doi:10.1109/TVCG.2017.2666146.

[43] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1079–1088. ACM, 2010. doi:10.1145/1753326.1753486.

[44] Jie Yin, Andrew Lampert, Mark Cameron, Bella Robinson, and Robert Power. Using Social Media to Enhance Emergency Situation Awareness. *IEEE Intelligent Systems*, 27(6):52–59, 2012. doi:2832847.

[45] Myung-Hwa Hwang, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Zhenhua Zhang. Spatiotemporal Transformation of Social Media Geostreams: a Case Study of Twitter for Flu Risk Analysis. In *Proceedings of*

the 4th ACM SIGSPATIAL International Workshop on GeoStreaming, pages 12–21. ACM, 2013. doi:10.1145/2534303.2534310.

[46] Justin Cranshaw, Raz Schwartz, Jason I. Hong, and Norman M. Sadeh. The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City. In John G. Breslin, Nicole B. Ellison, James G. Shanahan, and Zeynep Tufekci, editors, *ICWSM*. The AAAI Press, 2.

[47] Andrea Vaccari, Mauro Martino, Francisca Rojas, and Carlo Ratti. Pulse of the City: Visualizing Urban Dynamics of Special Events. In *Proceedings of the 20th International Conference on Computer Graphics and Vision*, GraphiCon, pages 1–10, 2010.

[48] Chaolun Xia, Raz Schwartz, Ke Xie, Adam Krebs, Andrew Langdon, Jeremy Ting, and Mor Naaman. CityBeat: Real-Time Social Media Visualization of Hyper-Local City Data. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 167–170. ACM, 2014. doi:10.1145/2567948.2577020.

[49] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. Travel Route Recommendation Using Geotags in Photo Sharing Sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 579–588, 2010. doi:10.1145/1871437.1871513.

[50] Alexandre Devaux and Nicolas Paparoditis. Increasing Interactivity in Street View Web Navigation Systems. In *Proceedings of the International Conference on Multimedia*, pages 903–906. ACM Press, 0. doi:10.1145/1873951.1874109.

[51] Thommen Korah and Yun-Ta Tsai. Urban Canvas: Unfreezing Street-View Imagery With Semantically Compressed LIDAR Pointclouds. In *Proceedings of 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 271–272. IEEE, 1. doi:10.1109/ISMAR.2011.6143897.

[52] Jianxiong Xiao and L Quan. Multiple View Semantic Segmentation for Street View Images. In *2009 IEEE 12th International Conference on Computer Vision*, pages 686–693. IEEE, 9. ISBN 978-1-4244-4420-5. doi:10.1007/978-3-319-10602-_34. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459249http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459249`.

[53] Angelo Nodari, Marco Vanetti, and Ignazio Gallo. Digital Privacy: Replacing Pedestrians From Google Street View Images. In *2012 21st International Conference OnPattern Recognition (ICPR)*, pages 2889–2893, 2. doi:10.1109/CVPRW.2010.5543255.

[54] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. Tohme: Detecting Curb Ramps in Google Street View Using Crowdsourcing, Computer Vision, and Machine Learning. In *Proceedings of the 27th*

Annual ACM Symposium on User Interface Software and Technology - UIST '14, pages 189–204. ACM, 4. doi:10.1145/2642918.2647403.

[55] Rattapoom Tuchinda, Pedro Szekely, Craig a Knoblock, and Marina Rey. Building Mashups by Example. In *The 13th International Conference on Intelligent User Interfaces*, pages 139–148. ACM, 8. ISBN 9781595939876. doi:10.1145/1378773.1378792.

[56] C. C. Robusto. The Cosine-Haversine Formula. *The American Mathematical Monthly*, 64(1):38–40, 7.

[57] Peter Shirley, Kelvin Sung, and William Brown. A Ray Tracing Framework for Global Illumination Systems. In *Proceedings of Graphics Interface '91*, pages 117–128, 1991. doi:10.1145/1198555.1198751.

[58] Michael McCool and Eugene Fiume. Hierarchical Poisson Disk Sampling Distributions. In *Proceedings of Graphics Interface '92*, volume 92, pages 94–105, 1992. doi:10.1145/1640443.1640451.

[59] Robert L Cook. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986. doi:10.1145/7529.8927.

[60] Cheuk Yiu Ip, M. Adil Yalçin, David Luebke, and Amitabh Varshney. Pixelpie: Maximal poisson-disk sampling with rasterization. In *Proceedings of the 5th High-Performance Graphics Conference*, HPG '13, pages 17–26, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2135-8. doi:10.1145/2492045.2492047. URL `http://doi.acm.org/10.1145/2492045.2492047`.

[61] L. Itti, C. Koch, and E. Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998. ISSN 0162-8828. doi:10.1109/34.730558.

[62] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-Based Visual Saliency. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, pages 545–552, Vancouver, British Columbia, Canada, 2006. doi:2976525.

[63] Youngmin Kim, Amitabh Varshney, David W Jacobs, and François Guimbretière. Mesh Saliency and Human Eye Fixations. *ACM Transactions on Applied Perception (TAP)*, 7(2):12, 2010. doi:10.1145/1670671.1670676.

[64] Cheuk Yiu Ip and Amitabh Varshney. Saliency-Assisted Navigation of Very Large Landscape Images. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):1737–1746, 2011. doi:10.1109/TVCG.2011.231.

[65] Xiaodi Hou, Jonathan Harel, and Christof Koch. Image Signature: Highlighting Sparse Salient Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):194–201, 1. ISSN 1939-3539. doi:10.1109/TPAMI.2011.146. URL `http://www.ncbi.nlm.nih.gov/pubmed/21788665`.

[66] Alonzo C. Addison. Virtual Heritage: Technology in the Service of Culture. In *Proceedings of the 2001 Conference on Virtual Reality, Archeology, and Cultural Heritage*, pages 343–380. ACM Press, 1. doi:584993.

[67] Jonas Lukasczyk, Ross Maciejewski, Christoph Garth, and Hans Hagen. Understanding Hotspots: a Topological Visual Analytics Approach. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 36, 2015. doi:10.1145/2820783.2820817.

[68] Ben Shneiderman. The Eyes Have It: a Task by Data Type Taxonomy for Information Visualizations. In *The Craft of Information Visualization*, pages 364–371. 10.1109/VL.1996.545307, 2003. doi:834354.

[69] Qiaomu Shen, Wei Zeng, Yu Ye, Stefan Müller Arisona, Simon Schubiger, Remo Burkhard, and Huamin Qu. StreetVizor: Visual Exploration of Human-Scale Urban Forms Based on Street Views. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1004–1013, 2018.

[70] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. ISSN 1532-4435.

[71] Robert Paige and Robert E Tarjan. Three Partition Refinement Algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987. doi:10.1137/0216062.

[72] George A Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63 (2):81, 1956.

[73] Christopher G Healey. Choosing Effective Colours for Data Visualization. In *Proceedings of the 7th Conference on Visualization'96*, pages 263–ff, 1996. doi:245597.

[74] Anthony A Apodaca, Larry Gritz, and Ronen Barzel. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 2000.

[75] David L Williamson and Philip J Rasch. Two-Dimensional Semi-Lagrangian Transport With Shape-Preserving Interpolation. *Monthly Weather Review*, 117(1):102–129, 9. doi:10.1137/0911039.

[76] Wade Roush. Second Earth. *Technology Review*, page 38, 7.

[77] David Gelernter. *Mirror Worlds: Or: the Day Software Puts the Universe in a Shoebox... How It Will Happen and What It Will Mean.* Oxford University Press, 1993. doi:120456.

[78] Ilaria Liccardi, Alfie Abdul-Rahman, and Min Chen. I Know Where You Live: Inferring Details of People's Lives by Visualizing Publicly Shared Location Data. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI, pages 1–12. ACM, 2016. doi:10.1145/3137616.3137617.

[79] Jose M Such, Joel Porter, Sören Preibusch, and Adam Joinson. Photo Privacy Conflicts in Social Media: a Large-Scale Empirical Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI, pages 3821–3832, 2017. doi:10.1145/3025453.3025668.

[80] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 3. doi:944937.

[81] A Donald Keedwell and József Dénes. *Latin Squares and Their Applications.* North-Holland, second edition edition, 2015. ISBN 978-0-444-63555-6. doi:https://doi.org/10.1016/B978-0-444-63555-6.50003-9.

[82] Matthew Carrasco and Andruid Kerne. Queer Visibility: Supporting LGBT+ Selective Visibility on Social Media. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI, page 250, 2018. doi:10.1145/3173574.3173824.

[83] Haley MacLeod, Cynthia L Bennett, Meredith Ringel Morris, and Edward Cutrell. Understanding Blind People's Experiences With Computer-Generated Captions of Social Media Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5988–5999. ACM, 2017. doi:10.1145/3025453.3025814.

[84] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: a Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *ArXiv Preprint ArXiv:1511.00561*, 5. doi:10.1109/TPAMI.2016.2644615.

[85] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-Level Accuracy With 50x Fewer Parameters and <1MB Model Size. *CoRR*, abs/1602.07360, 2016. URL `http://arxiv.org/abs/1602.07360`.

[86] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic Image Segmentation With Deep Convolutional Nets, Atrous Convolution, and Fully Connected Crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. doi:10.1109/TPAMI.2017.2699184.

[87] Arturo Flores and Serge Belongie. Removing Pedestrians From Google Street View Images. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 53–58. IEEE, 2010. doi:10.1109/CVPRW.2010.5543255.

[88] Angelo Nodari, Marco Vanetti, and Ignazio Gallo. Digital Privacy: Replacing Pedestrians From Google Street View Images. In *2012 21st International Conference on Pattern Recognition (ICPR)*, pages 2889–2893, 2012. doi:10.1109/CVPRW.2010.5543255.

[89] Lydia Manikonda and Munmun De Choudhury. Modeling and Understanding Visual Attributes of Mental Health Disclosures in Social Media. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI, pages 170–181, 2017. doi:10.1145/3025453.3025932.

[90] Ruofei Du, Sujal Bista, and Amitabh Varshney. Video fields: Fusing multiple surveillance videos into a dynamic virtual environment. In *Proceedings of the 21st International Conference on Web3D Technology*, Web3D, pages 165–172. ACM, July 2016. doi:10.1145/2945292.2945299.

[91] Robert Deloatch, Brian P Bailey, Alex Kirlik, and Craig Zilles. I Need Your Encouragement!: Requesting Supportive Comments on Social Media Reduces Test Anxiety. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI, pages 736–747, 2017. doi:10.1145/3025453.3025709.

[92] Ian P Howard and Brian J Rogers. *Binocular Vision and Stereopsis*. Oxford University Press, 1995. doi:10.1007/978-981-287-846-_25.

[93] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. Viewport-Adaptive Navigable 360-Degree Video Delivery. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, 2017. doi:10.1145/3123266.3123372.

[94] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, 2018. doi:10.1109/TVCG.2018.2793599.

[95] Taehyun Rhee, Lohit Petikam, Benjamin Allen, and Andrew Chalmers. Mr360: Mixed Reality Rendering for 360 Panoramic Videos. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1379–1388, 2017. doi:10.1109/TVCG.2017.2657178.

[96] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2Vid: Automatic Cinematography for Watching 360 Videos. In *Asian Conference on Computer Vision*, pages 154–171, 2016.

[97] Yu-Chuan Su and Kristen Grauman. Making 360 Video Watchable in 2d: Learning Videography for Click Free Viewing. *ArXiv Preprint*, 7.

[98] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 Pilot: Learning a Deep Agent for Piloting Through 360 Sports Video. In *CVPR*, page 3, 7.

[99] Amy Pavel, Björn Hartmann, and Maneesh Agrawala. Shot Orientation Controls for Interactive Cinematography With 360 Video. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 289–297, 2017. doi:10.1145/3126594.3126636.

[100] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh Saliency. *ACM Transactions on Graphics (TOG)*, 24(3):659–666, 2005. doi:10.1145/1186822.1073244.

[101] Youngmin Kim and Amitabh Varshney. Saliency-Guided Enhancement for Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):925–932, 2006. doi:10.1109/TVCG.2006.174.

[102] Nianyi Li, Jinwei Ye, Yu Ji, Haibin Ling, and Jingyi Yu. Saliency Detection on Light Field. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2806–2813, 2014. doi:10.1145/3107956.

[103] Xiaodi Hou and Liqing Zhang. Saliency Detection: a Spectral Residual Approach. In *2007. CVPR'07. IEEE Conference OnComputer Vision and Pattern Recognition*, pages 1–8, 2007. doi:10.1145/1631272.1631370.

[104] Chenlei Guo and Liming Zhang. A Novel Multiresolution Spatiotemporal Saliency Detection Model and Its Applications in Image and Video Compression. *IEEE Transactions on Image Processing*, 19(1):185–198, 2010. doi:1821438.

[105] John K Tsotsos, Scan M Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis, and Fernando Nuflo. Modeling Visual Attention Via Selective Tuning. *Artificial Intelligence*, 78(1):507–545, 1995. doi:10.1007/978-1-4615-0111-_22.

[106] Atrde Oliva, Antonio Torralba, Monica S Castelhano, and John M Henderson. Top-Down Control of Visual Attention in Object Detection. In *Proceedings on 2003 International Conference on Image Processing*, volume 1, pages 1–4, 2003. doi:10.1109/ICIP.2003.1246946.

[107] Dirk Walther and Christof Koch. Modeling Attention to Salient Proto-Objects. *Neural Networks*, 19(9):1395–1407, 2006. doi:10.1007/978-3-642-00582-_13.

[108] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-Aware Saliency Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1915–1926, 2012. doi:10.1109/TPAMI.2011.272.

[109] Yangqing Jia and Mei Han. Category-Independent Object-Level Saliency Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1761–1768, 2013. doi:10.1016/j.patcog.2015.07.005.

[110] Risheng Liu, Junjie Cao, Zhouchen Lin, and Shiguang Shan. Adaptive Partial Differential Equation Learning for Visual Saliency Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3866–3873, 2014. doi:10.1109/CVPR.2014.494.

[111] Qi Zhao and Christof Koch. Learning Saliency-Based Visual Attention: a Review. *Signal Processing*, 93(6):1401–1407, 2013. doi:10.1016/j.sigpro.2012.06.014.

[112] Eleonora Vig, Michael Dorr, and David Cox. Large-Scale Optimization of Hierarchical Features for Saliency Prediction in Natural Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2798–2805, 2014. doi:10.1109/CVPR.2014.358.

[113] Guanbin Li and Yizhou Yu. Visual Saliency Based on Multiscale Deep Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5455–5463, 2015.

[114] Lijun Wang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Deep Networks for Saliency Detection Via Local Estimation and Global Search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3183–3192, 2015. doi:10.1109/CVPR.2015.7298938.

[115] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency Detection by Multi-Context Deep Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1265–1274, 2015. doi:10.1109/CVPR.2015.7298731.

[116] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. Shallow and Deep Convolutional Networks for Saliency Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 598–606. IEEE, 2016. doi:10.1109/CVPR.2016.71.

[117] Ziheng Zhang, Yanyu Xu, Jingyi Yu, and Shenghua Gao. Saliency Detection in 360 Videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 488–503, 8.

[118] Neil DB Bruce and John K Tsotsos. Saliency, Attention, and Visual Search: an Information Theoretic Approach. *Journal of Vision*, 9(3):5–5, 2009. doi:1627279.

[119] Wei Wang, Yizhou Wang, Qingming Huang, and Wen Gao. Measuring Visual Saliency by Site Entropy Rate. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2368–2375, 2010. doi:10.1016/j.image.2016.03.003.

[120] Ronen Basri and David W Jacobs. Lambertian Reflectance and Linear Subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25 (2):218–233, 2003. doi:10.1109/TPAMI.2003.1177153.

[121] Robin Green. Spherical Harmonic Lighting: the Gritty Details. In *Archives of the Game Developers Conference*, volume 56, page 4, 2003.

[122] Sujal Bista, Jiachen Zhuo, Rao P Gullapalli, and Amitabh Varshney. Visualization of Brain Microstructure Through Spherical Harmonics Illumination of High Fidelity Spatio-Angular Fields. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2516–2525, 2014. doi:10.1109/TVCG.2014.2346411.

[123] Thushara D Abhayapala and Darren B Ward. Theory and Design of High Order Sound Field Microphones Using Spherical Microphone Array. In *2002 IEEE International Conference OnAcoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages II–1949, 2002. doi:10.1121/1.4978660.

[124] Mark A Poletti. Three-Dimensional Surround Sound Systems Based on Spherical Harmonics. *Journal of the Audio Engineering Society*, 53(11):1004–1025, 2005. doi:10.1121/1.3506352.

[125] Dejan V Vranic, Dietmar Saupe, and Jörg Richter. Tools for 3D-Object Retrieval: Karhunen-Loeve Transform and Spherical Harmonics. In *IEEE Fourth Workshop on Multimedia Signal Processing*, pages 293–298, 2001. doi:10.1007/3-540-45404-_52.

[126] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003. doi:10.1007/s00138.

[127] Perumaal Shanmugam and Okan Arikan. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pages 73–80, 2007. doi:10.1145/1230100.1230113.

[128] Olivier Génevaux, Frédéric Larue, and Jean-Michel Dischler. Interactive Refraction on Complex Static Geometry Using Spherical Harmonics. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games (I3D)*, pages 145–152. ACM, 2006. doi:10.1145/1111411.1111438.

[129] Markus Billeter, Erik Sintorn, and Ulf Assarsson. Real-Time Multiple Scattering Using Light Propagation Volumes. In *Proceedings of the 2012 Symposium on Interactive 3D Graphics (I3D)*, pages 119–126. ACM, 2012. doi:10.1145/1730804.1730821.

[130] Jaakko Lehtinen and Jan Kautz. Matrix Radiance Transfer. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics (I3D)*, pages 59–64. ACM, 2003. doi:10.1145/641480.641495.

[131] Guy E Blelloch. Scans As Primitive Parallel Operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989. doi:10.1109/12.42122.

[132] Peter Alfeld, Marian Neamtu, and Larry L Schumaker. Fitting Scattered Data on Sphere-Like Surfaces Using Spherical Splines. *Journal of Computational and Applied Mathematics*, 73(1-2):5–43, 1996. doi:10.1016/0377.

[133] Jules Bloomenthal. Calculation of Reference Frames Along a Space Curve. *Graphics Gems*, 1:567–571, 0. doi:10.1145/1330511.1330513.

[134] Brent M Dennis and Christopher G Healey. Assisted Navigation for Large Information Spaces. In *Proceedings of the Conference on Visualization '02*, pages 419–426. IEEE Computer Society, 2002. doi:10.1007/978-0-387-35175-_7.

[135] Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. Hovercam: Interactive 3D Navigation for Proximal Object Inspection. In *Roceedings of the 2005 Symposium on Interactive 3D Graphics and Games (I3D)*, pages 73–80. ACM, 2005. doi:10.1145/1053427.1053439.

[136] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera Control in Computer Graphics. *Computer Graphics Forum*, 27(8):2197–2218, 2008. doi:10.1145/1665817.1665820.

[137] Thomas Oskam, Alexander Hornung, Huw Bowles, Kenny Mitchell, and Markus H Gross. Oscam-Optimized Stereoscopic Camera Control for Interactive 3d. *ACM Transaction on Graphics (ToG)*, 30(6):189–1, 2011. doi:10.1145/2024156.2024223.

[138] Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. An Interactive Tool for Designing Quadrotor Camera Shots. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015. doi:10.1145/2816795.2818106.

[139] Wenguan Wang, Jianbing Shen, Yizhou Yu, and Kwan-Liu Ma. Stereoscopic Thumbnail Creation Via Efficient Stereo Saliency Detection. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):2014–2027, 2017. doi:10.1145/2483977.2483979.

[140] Miao Wang, Jun-Bang Liang, Song-Hai Zhang, Shao-Ping Lu, Ariel Shamir, and Shi-Min Hu. Hyper-Lapse From Multiple Spatially-Overlapping Videos. *IEEE Transactions on Image Processing*, 27(4):1735–1747, 2018. doi:10.1109/TIP.2017.2749143.

[141] Chris Stauffer and W Eric L Grimson. Adaptive Background Mixture Models for Real-Time Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 246–252, 1999. doi:10.1007/978-1-4615-0913-_11.

[142] Youngmin Kim and Amitabh Varshney. Persuading Visual Attention Through Geometry. *IEEE Transactions on Visualization and Computer Graphics*, 14 (4):772–782, 2008. doi:10.1109/TVCG.2007.70624.

[143] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 3D Puppetry: a Kinect-Based Interface for 3D Animation. In *Proceedings of the 25th Annual Symposium on User Interface Software and Technology (UIST)*, pages 423–434, 2012. doi:10.1145/2380116.2380170.

[144] Connelly Barnes, David E Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. Video Puppetry: a Performative Interface for Cutout Animation. *ACM Transactions on Graphics (TOG)*, 27(5):124, 2008. doi:10.1145/1457515.1409077.

[145] Lee Stearns, Ruofei Du, Uran Oh, Yumeng Wang, Leah Findlater, Rama Chellappa, and Jon E Froehlich. The design and preliminary evaluation of a finger-mounted camera and feedback system to enable reading of printed text for the blind. In *European Conference on Computer Vision (ECCV), Workshop on Assistive Computer Vision and Robotics*, pages 615–631. Springer, 2014.

[146] Lee Stearns, Ruofei Du, Uran Oh, Catherine Jou, Leah Findlater, David A. Ross, and Jon E. Froehlich. Evaluating haptic and auditory directional guidance to assist blind people in reading printed text using finger-mounted cameras. *ACM Transactions on Accessible Computing (TACCESS)*, 2016.

[147] Leah Findlater, Lee Stearns, Ruofei Du, Uran Oh, David Ross, Rama Chellappa, and Jon E Froehlich. Supporting everyday activities for persons with visual impairments through computer vision-augmented touch. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS 2015)*, pages 383–384. ACM, 2015.

[148] Claudio Marforio, Nikolaos Karapanos, Claudio Soriente, Kari Kostiainen, and Srdjan Cˇapkun. Smartphones As Practical and Secure Location Verification Tokens for Payments. In *The Network and Distributed System Security Symposium (NDSS)*, 4.

[149] Dongtao Liu, Eduardo Cuervo, Valentin Pistol, Ryan Scudellari, and Landon P Cox. Screenpass: Secure Password Entry on Touchscreen Devices. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, pages 291–304, 2013. doi:10.1145/2470654.2481331.

[150] Zakir Durumeric, James Kasten, David Adrian, J Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, et al. The Matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 475–488, 2014. doi:10.1145/2663716.2663755.

[151] Moni Naor and Adi Shamir. Visual Cryptography. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 1–12, 1994. doi:10.1109/ICRITO.2016.7784984.

[152] Mark A Livingston, Joseph L Gabbard, J Edward Swan II, Ciara M Sibley, and Jane H Barrow. Basic Perception in Head-Worn Augmented Reality Displays. In *Human Factors in Augmented Reality Environments*, pages 35–65. Springer, 2013. doi:10.1007/978-1-4614-4205-_3.

[153] Carlo Blundo, Alfredo De Santis, and Moni Naor. Visual Cryptography for Grey Level Images. *Information Processing Letters*, 75(6):255–259, 2000. doi:10.1016/S0020.

[154] Zhi Zhou, Gonzalo R Arce, and Giovanni Di Crescenzo. Halftone Visual Cryptography. *IEEE Transactions on Image Processing*, 15(8):2441–2453, 2006. doi:10.1109/TIP.2006.875249.

[155] Young-Chang Hou. Visual Cryptography for Color Images. *Pattern Recognition*, 36(7):1619–1629, 2003. doi:10.1109/COMPTELIX.2017.8003979.

[156] Bin Liu, Ralph R Martin, Ji-Wu Huang, and Shi-Min Hu. Structure Aware Visual Cryptography. *Computer Graphics Forum*, 33(7):141–150, 2014. doi:10.1111/cgf.12482.

[157] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 559–568. ACM, 2011. doi:10.1109/ISMAR.2011.6092378.

[158] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[159] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-Quality Streamable Free-Viewpoint Video. *ACM Transactions on Graphics (TOG)*, 34(4):69, 2015. doi:10.1145/2766945.

[160] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Philip A.Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. Holoportation: Virtual 3D Teleportation in Real-Time. In *Proceedings of the*

29th Annual Symposium on User Interface Software and Technology (UIST), pages 741–754, 2016. doi:10.1145/2984511.2984517.

[161] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4D: Real-Time Performance Capture of Challenging Scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016. doi:10.1145/2897824.2925969.

[162] David Bommes and Leif Kobbelt. Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes. In *Computer Cision, Graphics and Visualization Workshop*, volume 7, pages 151–160. VMV, 2007. doi:10.1.1.88.2462.

[163] H Fuchs and U Neumann. A Vision of Telepresence for Medical Consultation and Other Applications. In *Sixth International Symposium of Robotics Research*, pages 555–571, 1993. doi:10.1023/A.

[164] Henry Fuchs, Gary Bishop, Kevin Arthur, Leonard McMillan, Ruzena Bajcsy, Sang Lee, Hany Farid, and Takeo Kanade. Virtual Space Teleconferencing Using a Sea of Cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery (MRCAS)*, volume 26, 1994. doi:10.1145/1026776.1026790.

[165] Takeo Kanade, Peter Rander, and PJ Narayanan. Virtualized Reality: Constructing Virtual Worlds From Real Scenes. *IEEE Multimedia*, 4(1):34–47, 1997. doi:10.1109/93.580394.

[166] PJ Narayanan, Peter W Rander, and Takeo Kanade. Constructing Virtual Worlds Using Dense Stereo. In *Sixth International Conference on Computer Vision (ICCV)*, pages 3–10, 1998. doi:10.1109/ICCV.1998.710694.

[167] Herman Towles, Wei-Chao Chen, Ruigang Yang, Sang-Uok Kum, Henry Fuchs Nikhil Kelshikar, Jane Mulligan, Kostas Daniilidis, Henry Fuchs, Carolina Chapel Hill, Nikhil Kelshikar Jane Mulligan, et al. 3D tele-collaboration over Internet2. In *International Workshop on Immersive Telepresence*, 2002. doi:10.1109/ICDCS.2008.47.

[168] Bastian Goldluecke and Marcus Magnor. Space-Time Isosurface Evolution for Temporally Coherent 3D Reconstruction. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–350, 2004. doi:10.1109/CVPR.2004.1315053.

[169] Yasutaka Furukawa and Jean Ponce. Dense 3D Motion Capture From Synchronized Video Streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. doi:10.1109/CVPR.2009.5206868.

[170] Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance Capture From Sparse Multi-View Video. *ACM Transactions on Graphics (TOG)*, 27(3):98, 2008. doi:10.1145/1399504.1360697.

[171] Benjamin Lok. Online Model Reconstruction for Interactive Virtual Environments. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D)*, pages 69–72. ACM, 2001. doi:10.1145/364338.364364.

[172] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated Mesh Animation From Multi-View Silhouettes. *ACM Transactions on Graphics (TOG)*, 27(3):97, 2008. doi:10.1145/1399504.1360696.

[173] Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. Probabilistic Deformable Surface Tracking From Multiple Videos. In *ECCV'10 Proceedings of the 11th European Conference on Computer Vision: Part IV*, pages 326–339. Springer, 2010. doi:10.1007/978-3-642-15561-_24.

[174] Rob Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney. Social Snapshot: a System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications*, 31(1):74–84, 2011. doi:10.1109/MCG.2010.107.

[175] Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. Video-Based Characters: Creating New Human Performances From a Multi-View Video Database. *ACM Transactions on Graphics (TOG)*, 30(4):32, 2011. doi:10.1145/1964921.1964927.

[176] Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. Interactive Animation of 4D Performance Capture. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(5):762–773, 2013. doi:10.1145/2159616.2159633.

[177] Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. Motion Graphs for Unstructured Textured Meshes. *ACM Transactions on Graphics (TOG)*, 35(4):108, 2016. doi:10.1145/2897824.2925967.

[178] Fabian Prada, Misha Zazhdan, Ming Chuang, Collet Alvaro, and Hugues Hoppe. Spatiotemporal Atlas Parameterization for Evolving Meshes. *ACM Transactions on Graphics (TOG)*, 36(4):58, 2017. doi:10.1145/3072959.3073679.

[179] Adnane Boukhayma and Edmond Boyer. Surface Motion Capture Animation Synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2018. doi:10.1109/TVCG.2018.2831233.

[180] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-Quality Video View Interpolation Using a Layered

Representation. *ACM Transactions on Graphics (TOG)*, 23(3):600–608, 2004. doi:10.1145/1015706.1015766.

[181] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *2011 10th IEEE International Symposium OnMixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. doi:10.1109/ISMAR.2011.6092378.

[182] Mao Ye and Ruigang Yang. Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2345–2352, 2014. doi:10.1109/CVPR.2014.301.

[183] Qing Zhang, Bo Fu, Mao Ye, and Ruigang Yang. Quality Dynamic Human Body Modeling Using a Single Low-Cost Depth Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 676–683. IEEE, 2014. doi:10.1109/CVPR.2014.92.

[184] Weipeng Xu, Mathieu Salzmann, Yongtian Wang, and Yue Liu. Deformable 3D Fusion: From Partial Dynamic 3D Observations to Complete 4D Models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2183–2191. IEEE, 2015. doi:10.1109/ICCV.2015.252.

[185] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. Robust Non-Rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3083–3091. IEEE, 2015. doi:10.1109/ICCV.2015.353.

[186] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017. doi:10.1145/3083722.

[187] Tao Yu, Kaiwen Guo, Feng Xu, Yuan Dong, Zhaoqi Su, Jianhui Zhao, Jianguo Li, Qionghai Dai, and Yebin Liu. BodyFusion: Real-Time Capture of Human Motion and Surface Geometry Using a Single Depth Camera. In *The IEEE International Conference on Computer Vision (ICCV)*. ACM, October 2017. doi:10.1109/ICCV.2017.104.

[188] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-Time Volumetric Performance Capture. *ACM Transactions on Graphics (TOG)*, 36(6):246, 2017. doi:10.1145/3130800.3130801.

[189] Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. UltraStereo: Efficient Learning-Based Matching for Active Stereo Systems. In *2017 IEEE Conference OnComputer Vision and Pattern Recognition (CVPR)*, pages 6535–6544. IEEE, 2017. doi:10.1109/CVPR.2017.692.

[190] Sean Ryan Fanello, Julien Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip Davidson, and Shahram Izadi. Low Compute and Fully Parallel Computer Vision With HashMatch. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3894–3903. IEEE, 2017. doi:10.1109/ICCV.2017.418.

[191] Paul Debevec, Steven Gortler, Leonard McMillan, Richard Szeliski, and Chris Bregler. Image-Based Modeling and Rendering. *SIGGRAPH 98 Course Notes for Course*, 15, 1998. doi:10.1109/VR.2003.1191174.

[192] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient View-Dependent Image-Based Rendering With Projective Texture-Mapping. In *Rendering Techniques' 98*, pages 105–116. Eurographics, 1998. doi:10.1007/978-3-7091-6453-_10.

[193] Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. Seamless Montage for Texturing Models. *Computer Graphics Forum*, 29(2): 479–486, 2010. doi:10.1111/j.1467-8659.2009.01617.x.

[194] Cédric Allène, Jean-Philippe Pons, and Renaud Keriven. Seamless Image-Based Texture Atlases Using Multi-Band Blending. In *19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008. doi:10.1109/ICPR.2008.4761913.

[195] Zsolt Janko and Jean-Philippe Pons. Spatio-Temporal Image-Based Texture Atlases for Dynamic 3-D Models. In *IEEE 12th International Conference on Computer Vision Workshops*, pages 1646–1653. IEEE, 2009. doi:10.1109/ICCVW.2009.5457481.

[196] Victor Lempitsky and Denis Ivanov. Seamless Mosaicing of Image-Based Texture Maps. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007. doi:10.1109/CVPR.2007.383078.

[197] Kun Zhou, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. TextureMontage: Seamless Texturing of Arbitrary Surfaces From Multiple Images. *ACM Transactions on Graphics (TOG)*, 24(3): 1148–1155, 2005. doi:10.1145/1073204.1073325.

[198] Qian-Yi Zhou and Vladlen Koltun. Color Map Optimization for 3D Reconstruction With Consumer Depth Cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014. doi:10.1145/2601097.2601134.

[199] Karthik S Narayan and Pieter Abbeel. Optimized Color Models for High-Quality 3D Scanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2503–2510, 2015. doi:10.1109/IROS.2015.7353717.

[200] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum*, 27(2):409–418, 2008. doi:10.1111/j.1467-8659.2008.01138.x.

[201] Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 4d Video Textures for Interactive Character Appearance. *Computer Graphics Forum*, 33(2):371–380, 2014. doi:10.1145/2775292.2775297.

[202] Marco Volino, Dan Casas, John P Collomosse, and Adrian Hilton. Optimal Representation of Multi-View Video. In *British Machine Vision Conference 2014*, 2014. doi:10.1109/ICC.2017.7996612.

[203] Joseph SB Mitchell. Geometric Shortest Paths and Network Optimization. *Handbook of Computational Geometry*, 334:633–702, 2000. doi:10.1007/978-1-4613-0303-_10.

[204] Eitan Grinspun, Mathieu Desbrun, Konrad Polthier, Peter Schröder, and Ari Stern. Discrete Differential Geometry: an Applied Introduction. *ACM SIGGRAPH Course*, 7, 2006. doi:10.1145/3245634.

[205] Fernando do Goes, Mathieu Desbrun, and Yiying Tong. Vector Field Processing on Triangle Meshes. In *SIGGRAPH Asia 2015 Courses*, page 17, 2015. doi:10.1145/2818143.2818167.

[206] Joseph SB Mitchell, David M Mount, and Christos H Papadimitriou. The Discrete Geodesic Problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. doi:10.1145/1559755.1559761.

[207] Yipeng Qin, Xiaoguang Han, Hongchuan Yu, Yizhou Yu, and Jianjun Zhang. Fast and Exact Discrete Geodesic Computation Based on Triangle-Oriented Wavefront Propagation. *ACM Transactions on Graphics*, 35(4), 6. doi:10.1145/2897824.2925930.

[208] Sanjiv Kapoor. Efficient Computation of Geodesic Shortest Paths. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 770–779, 1999. doi:10.1145/301250.301449.

[209] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximating Weighted Shortest Paths on Polyhedral Surfaces. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pages 274–283, 1997. doi:10.1145/262839.263101.

[210] Takashi Kanai and Hiromasa Suzuki. Approximate Shortest Path on a Polyhedral Surface and Its Applications. *Computer-Aided Design*, 33(11):801–811, 2001. doi:10.1145/1044731.1044733.

[211] Dimas Martınez, Luiz Velho, and Paulo Cezar Carvalho. Geodesic Paths on Triangular Meshes. In *Proceedings of Computer Graphics and Image Processing*, 2004. doi:10.1109/SIBGRA.2004.1352963.

[212] Vibhav Vineet, Jonathan Warrell, and Philip HS Torr. Filter-Based Mean-Field Inference for Random Fields With Higher-Order Terms and Product Label-Spaces. *International Journal of Computer Vision*, 110(3):290–307, 2014. doi:10.1007/s11263.

[213] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi:10.1109/TIP.2003.819861.

[214] Martin Eisemann and Marcus A Magnor. Filtered Blending: a New, Minimal Reconstruction Filter for Ghosting-Free Projective Texturing With Multiple Images. In *Proc. Vision, Modeling and Visualization (VMV)*, pages 119–126. Eurographics, Nov 2007.

[215] Adrian Secord. Weighted Voronoi Stippling. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering*, pages 37–43. ACM, 2002. doi:10.1145/508530.508537.

[216] Oscar Meruvia Pastor, Bert Freudenberg, and Thomas Strothotte. Real-Time Animated Stippling. *IEEE Computer Graphics and Applications*, 23(4):62–68, 2003. doi:10.1109/MCG.2003.1210866.

[217] Dongyeon Kim, Minjung Son, Yunjin Lee, Henry Kang, and Seungyong Lee. Feature-Guided Image Stippling. *Computer Graphics Forum*, 27(4):1209–1216, 2008. doi:10.1111/j.1467.

[218] Abhimitra Meka, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live Intrinsic Video. *ACM Transactions on Graphics (TOG)*, 35 (4):109, 2016. doi:10.1145/2897824.2925907.

[219] Abhimitra Meka, Maxim Maximov, Michael Zollhöfer, Avishek Chatterjee, Christian Richardt, and Christian Theobalt. Live Intrinsic Material Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[220] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4d: Interactive seamless fusion of multiview video textures. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, I3D, pages 124–133. ACM, May 2018. doi:10.1145/3190834.3190843.

[221] Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques (I3D)*, 1(5):1–20, May 2018. doi:10.1145/3203199.