Understanding Semantics and Geometry of Scenes

by

Mohammed Suhail

B.Tech., Indian Institute of Technology, Guwahati, 2017
M.Sc., University of British Columbia, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

in
THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

The University Of British Columbia
(Vancouver)
April 2024
© Mohammed Suhail 2024
The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

**Understanding Semantics and Geometry of Scenes**

submitted by Mohammed Suhail in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

**Examinining Committee:**
Leonid Sigal, Professor, Department of Computer Science, UBC  
*Supervisor*

Helge Rhodin, Assistant Professor, Department of Computer Science, UBC  
*Supervisory Committee Member*

Kwang Moo Yi, Assistant Professor, Department of Computer Science, UBC  
*Supervisory Committee Member*

Jeff Clune, Assistant Professor, Department of Computer Science, UBC  
*University Examiner*

Purang Abolmaesumi, Professor, Department of Electrical and Computer Engineering, UBC  
*University Examiner*

Hao (Richard) Zhang, Professor, Department of Computing Science, Simon Fraser University  
*External Examiner*
Abstract

In this dissertation, we present new approaches for structured scene understanding from images and videos. Structured scene understanding finds numerous applications, including in robotics and autonomous vehicles, as well as in 3D content creation and video editing. The focus of this research is on three specific tasks: scene graph generation, novel view synthesis, and layered scene representation. Scene graph generation involves creating a graph structure that represents the objects and their relationships in a scene. Generating a scene graph from an image demands a comprehensive comprehension of constituent objects and their associations. Our exploration delves into integrating the often overlooked structure of the output space into the reasoning framework. Additionally, we extend beyond bounding box granularity by leveraging pixel-level masks to ground objects when such annotations are absent in scene graph datasets. Novel view synthesis involves generating new views of a scene from input images. Achieving this demands a deep comprehension of the scene’s underlying geometry to ensure the rendering of pixels aligns seamlessly with the scene’s structure. Within this dissertation, our exploration centers on methods capable of accurately rendering scenes, particularly when dealing with non-Lambertian surfaces. Moreover, we address the challenge of developing view-synthesis techniques capable of generating new scene perspectives without necessitating training for each scene. Layered scene representation involves decomposing a scene into different semantically meaningful layers. In our pursuit of this task, we confront the constraints inherent in existing methods when handling videos with parallax effects resulting from homography-based modeling. To address this, our exploration focuses on a methodology aimed at learning a three-dimensional (3D) layered representation. This approach aims to surpass these limitations and facilitate a more comprehensive scene decomposition. The main contributions of this thesis thus include the exploration and advancement of these tasks.
Lay Summary

This dissertation presents new methods to understand images and videos better by focusing on three main tasks: creating scene graphs, generating new scene views, and segmenting scenes into layers. Scene graphs serve as maps revealing objects in a scene and their connections, demanding a comprehensive understanding of scene elements and their relationships. This research enhances graph accuracy by inferring missing details and reasoning about their structure. Generating new scene views involves creating fresh perspectives from existing images, a challenge requiring a profound grasp of scene geometry. The thesis focuses on refining techniques, especially for non-traditional surfaces and ensuring applicability across diverse scenes. Segmenting scenes into layers aims to dissect scenes into understandable components, particularly addressing complex camera movements in videos. This approach strives for a more precise three-dimensional comprehension, refining scene breakdown accuracy. Overall, this thesis advances these tasks, contributing to a more nuanced understanding of scenes.
Preface

The chapters in this thesis consist of original works of Mohammed Suhail under the guidance of Leonid Sigal, Behjat Siddiquie, Carlos Esteves, Ameesh Makadia and Forrester Cole. This dissertation is based on the following publications.

  In this work, I developed the idea of using energy-based modelling for scene graph generation. I implemented the method, performed the experiments and wrote the paper. Abhay Mittal, Behjat Siddiquie and Leonid Sigal provided advice during project development and helped with paper writing. Gerard Medioni and Jayan Eledath provided feedback.

  The central idea for this work was developed during discussions with the other two authors. Siddesh Khandelwal implemented the segmentation-related components of the method and I implemented the scene-graph-related components. We both performed experiments and contributed equally to the writing. Leonid Sigal provided valuable feedback and suggestions during project development and helped write the paper.

  The core idea of incorporating a geometric bias into rendering light fields was developed during discussions with Carlos Esteves. I proposed the architecture design, implemented the method, performed the experiments and wrote the paper. Carlos Esteves, Ameesh Makadia and Leonid Sigal provided valuable feedback and helped write the paper.

The idea for this paper emerged from the work done in Chapter 5. I implemented the model and performed the necessary experiments. Carlos Esteves, Ameesh Makadia and Leonid Sigal provided valuable feedback and helped write the paper.


Forrester Cole proposed working on layered decomposition for casual videos. I proposed the algorithm, implemented the method, performed experiments and wrote the paper. All other authors provided feedback and helped with paper writing.

* Denotes equal contribution
# Contents

Abstract ................................................................. iii

Lay Summary ............................................................ iv

Preface ................................................................. v

Contents ............................................................... vii

List of Tables .......................................................... x

List of Figures .......................................................... xi

1 Introduction .......................................................... 1
  1.1 Scene graph generation ........................................... 2
  1.2 Novel View Synthesis ............................................. 3
  1.3 Layered Scene Representation .................................. 4
  1.4 Thesis Overview and Contributions .............................. 5

2 Related Work .......................................................... 7
  2.1 Scene Graph Generation ........................................... 7
  2.2 Novel View Synthesis ............................................. 8
  2.3 Layered Scene Representation .................................. 10

3 Energy-Based Learning for Scene Graph Generation .......... 12
  3.1 Approach .......................................................... 15
    3.1.1 Scene Graph Generation ................................... 15
    3.1.2 Energy Based Modeling ..................................... 16
    3.1.3 Energy Models for Scene Graphs Generation .............. 17
    3.1.4 Energy Model Architecture ................................ 19
  3.2 Experiments ........................................................ 20
    3.2.1 Scene Graph Generation Models ............................ 21
    3.2.2 Evaluation ................................................... 21
    3.2.3 Implementation Details ....................................... 22
## Contents

3.3 Experimental Results .................................................. 23
3.4 Conclusion and Discussion .............................................. 27

4 Segmentation-grounded Scene Graph Generation ....................... 28
4.1 Approach ................................................................. 30
  4.1.1 Notation .............................................................. 30
  4.1.2 Scene Graph Generation ........................................... 31
  4.1.3 Segmentation Mask Transfer ....................................... 32
  4.1.4 Grounding Nodes to Segmentation Masks .......................... 33
  4.1.5 Grounding Edges to Segmentation Masks .......................... 33
  4.1.6 Refining Segmentation Masks ...................................... 34
  4.1.7 Training ............................................................. 35
4.2 Experiments ............................................................... 36
  4.2.1 Scene Graph Generation Model .................................... 37
  4.2.2 Evaluation .......................................................... 37
  4.2.3 Implementation Details ............................................ 38
4.3 Results ................................................................. 38
4.4 Discussion ............................................................... 42

5 Light Field Neural Rendering ............................................ 43
5.1 Approach ................................................................. 45
  5.1.1 Light field parametrization ......................................... 46
  5.1.2 Model overview ..................................................... 46
  5.1.3 Network architectures ............................................. 47
  5.1.4 Loss ................................................................. 50
5.2 Experiments ............................................................... 50
  5.2.1 Implementation details ............................................. 51
  5.2.2 Results ............................................................. 51
  5.2.3 Ablation studies ................................................... 53
  5.2.4 Interpreting the model ............................................. 54
5.3 Limitations ............................................................... 56
5.4 Discussion ............................................................... 57

6 Generalizable Patch-Based Neural Rendering .......................... 59
6.1 Approach ................................................................. 61
  6.1.1 Light field representation .......................................... 61
  6.1.2 Patch extraction .................................................... 62
  6.1.3 Patch embedding and positional encoding .......................... 63
  6.1.4 Canonicalized ray representation ................................... 64
  6.1.5 Rendering network ................................................ 64
6.2 Experiments ................................................. 67
  6.2.1 Implementation Details .......................... 67
  6.2.2 Results ............................................. 67
6.3 Discussion ................................................ 70

7 Associating Object and their Effects in Unconstrained Monocular Video 72
  7.1 Method .................................................. 75
    7.1.1 Layer Decomposition Networks .................. 75
    7.1.2 Differentiable Rendering .......................... 76
    7.1.3 Losses ............................................. 76
    7.1.4 Frame Selection ................................... 80
    7.1.5 Detail Transfer .................................... 80
  7.2 Experiments ............................................. 80
    7.2.1 Training Details .................................. 80
    7.2.2 Layer Decomposition Results .................... 81
    7.2.3 Object Removal .................................... 82
    7.2.4 Depth-based Applications ........................ 83
    7.2.5 Ablations .......................................... 84
  7.3 Discussion .............................................. 86

8 Conclusion ................................................ 87

Bibliography .................................................. 89
# List of Tables

3.1 Quantitative Results ................................................. 20
3.2 Zero-shot Recall .................................................. 23
3.3 Few-shot Recall@20 .................................................. 24
3.4 Sentence to Graph Retrieval ....................................... 25
3.5 Ablation ............................................................... 25

4.1 Scene Graph Prediction on Visual Genome ...................... 36
4.2 Segmentation Refinement on MSCOCO ............................ 36
4.3 Zero-Shot Recall on Visual Genome ............................... 40
4.4 Ablation ............................................................... 41

5.1 Results on RFF dataset .............................................. 50
5.2 Results on Shiny Dataset ............................................ 52
5.3 Results on Blender Dataset ......................................... 52
5.4 Ablation on RFF ...................................................... 53

6.1 Results for setting 1 .................................................. 69
6.2 Results for setting 2 .................................................. 71
6.3 Ablations ............................................................... 71
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Scene Graph</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Novel View Synthesis</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Layered Scene Representation</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>Scene Graph Generation</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Model Overview of the Energy-based Learning</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Qualitative Results</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Segmentation-grounded scene graph generation</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Model Architecture for Segmentation Grounded Scene Graph Generation</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>Qualitative Results</td>
<td>39</td>
</tr>
<tr>
<td>5.1</td>
<td>Novel view synthesis</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>Model Overview</td>
<td>45</td>
</tr>
<tr>
<td>5.3</td>
<td>Qualitative Comparison</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Correspondence Distribution</td>
<td>54</td>
</tr>
<tr>
<td>5.5</td>
<td>Disparity Map</td>
<td>55</td>
</tr>
<tr>
<td>5.6</td>
<td>Epipolar-plane images (EPI)</td>
<td>56</td>
</tr>
<tr>
<td>5.7</td>
<td>Failure Cases</td>
<td>57</td>
</tr>
<tr>
<td>6.1</td>
<td>Motivation overview</td>
<td>60</td>
</tr>
<tr>
<td>6.2</td>
<td>Model Overview</td>
<td>62</td>
</tr>
<tr>
<td>6.3</td>
<td>Qualitative results on RFF</td>
<td>68</td>
</tr>
<tr>
<td>6.4</td>
<td>Qualitative results on Shiny</td>
<td>70</td>
</tr>
<tr>
<td>7.1</td>
<td>Layer decomposition under strong camera parallax</td>
<td>72</td>
</tr>
<tr>
<td>7.2</td>
<td>Existing methods fail on scenes with parallax</td>
<td>73</td>
</tr>
<tr>
<td>7.3</td>
<td>Model Overview</td>
<td>74</td>
</tr>
<tr>
<td>7.4</td>
<td>Qualitative layer decomposition results on DAVIS</td>
<td>77</td>
</tr>
<tr>
<td>7.5</td>
<td>Object removal</td>
<td>81</td>
</tr>
<tr>
<td>7.6</td>
<td>Camera stabilization</td>
<td>82</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.7</td>
<td>Synthetic defocus</td>
<td>83</td>
</tr>
<tr>
<td>7.8</td>
<td>Frame Selection Ablation</td>
<td>84</td>
</tr>
<tr>
<td>7.9</td>
<td>Projection Loss Ablation</td>
<td>85</td>
</tr>
<tr>
<td>7.10</td>
<td>Detail Transfer Ablation</td>
<td>85</td>
</tr>
<tr>
<td>7.11</td>
<td>Limitation: object occlusions</td>
<td>86</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Human beings possess an inherent ability to effortlessly recognize and differentiate objects within a visual scene, comprehend their three-dimensional geometry, and infer the relationships between them. This ability allows us to understand and make sense of our surroundings. The goal of this thesis is to develop effective computational methods and models that achieve a similar level of scene understanding, enabling them to understand the semantics and geometry of environments from images or videos. The semantics of scenes encompass the underlying meaning and conceptual associations embedded within them. Every scene carries a distinct narrative, composed of objects, people, and their interactions, which collectively shape our understanding of the environment. From the bustling streets of a city to the serene landscapes of nature, the semantics provide us with context, enabling us to recognize and assign meaning to different elements within our visual field. On the other hand, geometry refers to the spatial structure, layout, and arrangement of objects and their relationships in three-dimensional space. The geometry of a scene encompasses the physical boundaries, depths, and distances that define visual appearance and perspective. Understanding geometry is crucial for perceiving the layout, scale, and spatial organization, allowing us to navigate through our surroundings and make informed judgments about object relationships and interactions.

In this thesis, we explore the problem of understanding the semantics and geometry of scenes through the lens of three different tasks: scene graph generation, novel view synthesis and layered scene representation. Scene graph generation involves creating an image-grounded graph structure that represents the objects and their relationships in a scene. Novel view synthesis involves generating new views of a scene from a set of reference images. Achieving this task necessitates the model’s comprehension of the scene’s underlying geometry and its fidelity in preserving this structure while rendering the desired target view. Layered scene representation involves decomposing a scene into different layers that contain objects of interest along with their associated effects (e.g., shadow). Our exploration into these tasks is aimed at advancing the broader objective of enhancing scene understanding capabilities.

In the subsequent chapter, we delve into the background of each task, offering
Chapter 1. Introduction

Figure 1.1: **Scene Graph.** An example image (left) with its corresponding scene graph (right). The scene graph represents objects as nodes and the relation between them as edges. Each object node is grounded in the image using a bounding box.

a comprehensive understanding. Following that, we outline and detail the contributions made within this thesis toward these specific tasks.

1.1 Scene graph generation

Scene Graph Generation has emerged as a vital task in the field of computer vision and artificial intelligence, enabling machines to comprehend, analyze and represent visual scenes with a high level of detail and rigor. A scene graph, as shown in Figure 1.1, represents the objects present in an image along with their relationships, providing a structured representation of the visual content of the scene. Recently, scene graph generation has experienced significant advancements, driven by the development of neural network architectures, and the increasing demand in various applications, such as image captioning [34, 150], visual question answering [42], and scene generation [39, 46].

The primary goal of scene graph generation is to construct a graph-based representation, that captures the objects present as nodes and their relationships as edges, from an image. Each node in the graph corresponds to an object, characterized by its class label and spatial location. Edges represent relationships between objects, indicating their interactions or spatial arrangements. By incorporating contextual information about objects and their relationships, scene graphs facilitate higher-level reasoning and inference about the scene’s content.

Scene graph generation methods leverage the power of neural networks to automatically learn rich and discriminative representations from raw image data. Convolution-based networks are employed as feature extractors, capturing low-
Figure 1.2: **Novel View Synthesis.** Given a set of observations of a scene (black) the goal of novel view synthesis is to generate an image from an unseen viewpoint (green) that accurately reconstructs the underlying scene.

level visual cues and object-level information, while recurrent [156] or graph neural networks [147] are utilized to model object relationships and generate the scene graph structure. These models have demonstrated superior performance compared to traditional methods, achieving state-of-the-art results on benchmark datasets.

Scene graph generation, however, still faces several challenges. One of the primary challenges is structure in the output space that needs to be respected. Scene graphs are inherently structured, but most methods treat the objects and relations as independent entities during inference or when learning. In addition to losing structure-based reasoning and consistency, such methods can be adversely affected by an imbalance in the number of training labels leading to biased predictions at test-time [128]. Another challenge lies in grounding the objects to images. Scene graph datasets generally use bounding box annotation to ground objects to images. However, as shown in various visual and visual-lingual tasks [36, 41, 53], a more granular pixel-level ground proves more valuable in real-world applications. In this dissertation, we will explore methods to address these challenges to improve the performance and usability of scene graph generation methods.

### 1.2 Novel View Synthesis

Novel view synthesis is a dynamic and vibrant field of research within computer vision and graphics. It involves generating new views of a scene from previously unseen viewpoints, by leveraging an existing set of reference images of that scene (Figure 1.2). This task has gained immense interest due to its wide range of applications in fields such as virtual reality and 3D content generation. By synthesizing
new views, novel view synthesis enables immersive experiences and enhances the realism of virtual environments.

Novel view synthesis, however, is a highly challenging task that requires a deep understanding of the underlying 3D structure of the scene. Generating new pixels that are consistent with this structure poses significant difficulties. Designing an effective system for novel view synthesis involves two fundamental components: choosing an appropriate representation of the scene and developing a rendering method to convert this representation into a visually coherent image.

Classical methods for view synthesis have explored various representations such as light fields [63] and lumigraphs [33] to achieve a faithful reproduction of the scene. These methods, however, relied on dense sampling of the scene which limited their applicability. Recently, significant progress in view synthesis has been achieved by coupling volumetric, or surface, renderers with neural networks that learn an implicit or explicit representation of the underlying scene from a sparse set of image samples. While these methods have brought us tantalizingly close to the capability of creating photo-realistic images, several challenges remain [129]. For instance, these methods are challenged when the underlying scene contains complex non-Lambertian effects (e.g., reflections or refractions). Additionally, developing a generalizable model that can render novel views of previously unseen scenes without requiring additional training or fine-tuning is a crucial objective. Such a model would greatly enhance the practicality and usability of novel view synthesis, enabling seamless adaption to new scenes and environments.

In this dissertation, we aim to address these limitations and contribute to the advancement of novel view synthesis. Our focus will be on exploring innovative methods to reconstruct scenes with complex view-dependent effects and developing a generalizable model for rendering novel views. By tackling these challenges, we aspire to push the boundaries of novel view synthesis and pave the way for more realistic and immersive virtual experiences.

1.3 Layered Scene Representation

In computer vision and computer graphics, the representation of complex scenes is a fundamental challenge. Capturing the rich structure and depth of real-world scenes and objects requires techniques that can handle various sources of information, such as images, videos, depth maps, and 3D models. One approach that has gained significant attention and proven to be effective is layered scene representation. A layered scene representation organizes and structures the objects and elements in a three-dimensional scene. It involves dividing the scene into a background and multiple foreground layers, with each foreground layer representing a
Chapter 1. Introduction

5

Background Foreground RGB Foreground Alpha Input

Figure 1.3: Layered Scene Representation. Given an input RGB and mask, a layered scene representation decomposes the scene into a background and foreground layers where the foreground consists of the object represented by the input mask (snowboarder) along with its associated effects (shadow).

specific object of interest. This concept of layers is inspired by how the human visual system processes and understands complex scenes, where objects and surfaces at different depths are mentally separated and organized [81].

Layered scene representation has found applications in several domains. In computer graphics, it enables realistic rendering and image synthesis by accurately modelling the interactions of light with different layers and surfaces. In robotics, layered scene representation is crucial for object detection, scene understanding, and navigation tasks [146]. It also plays a significant role in augmented reality systems, where virtual objects need to be seamlessly integrated into the real world [12]. A more challenging variant of the problem is decomposing into layers where the foreground layer, in addition to the object, also has its associated effects such as shadows, reflections, etc. Figure 1.3 show an example of such a decomposition of a scene. In this dissertation, we will explore how to extract such a layered representation from an unconstrained monocular video.

1.4 Thesis Overview and Contributions

The main contribution of this thesis is the exploration of three different tasks that facilitate understanding the semantics * and the geometry of scenes captured either as images or videos. In proposing solutions for these tasks we focus on modeling the core problem-specific constraints and incorporating them explicitly, or implicitly, into the corresponding methods; be it structure † and label coherence of the graph or Epipolar geometric consistency. This systematic approach allows us to address prominent challenges and shortcomings.

*Semantic understanding refers to the interpretation and comprehension of the meaning embedded within scenes.
†Structure here refers to the organized arrangement of elements encompassing information about the relations and interactions between them.
Chapter 1. Introduction

Chapters 3 and 4 investigate the task of scene graph generation. In Chapter 3 we explore an energy-based learning framework for training scene graph generation models. Such a formulation allows us to efficiently reason about the structure of the scene graphs in the output space. We find that the proposed framework can learn efficiently from a small number of labels and helps alleviate issues arising from the long-tailed distribution of relation labels. In Chapter 4 we enhance the scene graphs by grounding objects to pixel-level segmentations instead of bounding boxes. The central focus is to investigate if scene graph generation can benefit from a segmentation-level grounding and equivalently can segmentation predictions be improved using information available in the scene graph. An important trait of the two methods presented in the corresponding chapters is that they develop methodologies that are agnostic to the underlying scene graph generation architecture and can thereby be easily integrated with future developments.

Chapters 5 and 6 explore the task for novel view synthesis. Recent works in novel view synthesis have shown impressive performance by encoding the 3D structure in a neural network and using volumetric rendering to generate images from unseen viewpoints. In Chapter 5 we ask the question “Can we learn a rendering network that can predict the color of a ray without volumetric rendering?” We investigate the use of light field, a function that maps a 4D representation of a ray to a radiance value, to represent a scene. Our findings show that such an approach not only improves the fidelity of the rendering, but is more suitable when the scene contains complex view-dependant effects such as reflections and refractions. In Chapter 6 we move further toward an image-based rendering approach investigating if novel views of a scene can be generated using just patches from the input images. We find that such modelling allows for the faithful rendering of new views on unseen scenes using a single pretrained model.

In Chapter 7, to combine the semantic and geometric understanding, we investigate the task of learning a layered scene representation. The goal here is to extract semantically meaningful layers along with their associated effects and 3D structure from an unconstrained input video. We demonstrate the merits of such a representation through use in applications such as camera stabilization, object removal and synthetic defocus.

The methods discussed in this thesis are aimed at better understanding the semantics and geometry of scenes at varying levels of granularity. Scene graph generation methods help in semantic reasoning by extracting a 2D structured representation at the granularity of objects. Novel view synthesis methods discussed in this thesis, implicitly reason about the 3D geometry of the scene from sampled views at a pixel-level granularity. Finally, the layered scene representation requires grouping entities into semantically meaningful layers (object-level) while reasoning about the 3D structure (pixel-level).
Chapter 2

Related Work

In this chapter, we discuss prior work related to scene graph generation, novel view synthesis and layered scene representation.

2.1 Scene Graph Generation

Scene graphs are graphical representation that summarize the object and their relations in an image (see Figure 1.1). The nodes in a scene graph represent objects along with their corresponding bounding box information. The relation between these objects are encoded in the directed edges of the graph. The goal of scene graph generation is to summarize an input image by identifying participant objects and their interactions in the form of a scene graph.

Scene graph generation has attracted much attention in the vision community [57, 65, 70, 88, 128, 144, 147, 154–156]. Scene graph generation methods can be roughly categorized into two: one-stage and two-stage methods. The two-stage approach consist of first detecting the object in the scene using a pre-trained network followed by determination of relation between the identified entities. The less common approach of one-stage generation involves jointly inferring the objects and relations ([88]). Due to the dependence of relation identification on objects, the two-stage approach has gained more traction, and we will primarily focus on such methods in this thesis.

A general two-stage scene graph generation pipeline works as follows: Given an image, the model first generates object proposal using a Region Proposal Network (RPN). These initial bounding boxes are used to instantiate the features of nodes in an intermediate graph. The features for the edges connecting the nodes are obtained using the union of the bounding boxes of the object nodes it connects. With this feature graph in place, various method are employed to aggregate contextual information and refine the features before predicting the object and relation labels. A cross-entropy based loss is used for each object and relation prediction to penalize wrong predictions and drive the learning process. Several works have explored the use of different architectures for contextual aggregation as well as novel loss formulations to improve scene graph generation, some of which will be discussed below.
Xu et al. [144] proposed to use an iterative message passing model using RNN to refine the feature of nodes and edges prior to classification. In Neural-Motif [156], the authors identified recurring structures prevalent in scene graphs and proposed an LSTM-based network along with a frequency-based bias to improve performance. The frequency bias helps identify the number of occurrences of relation triplets in a scene graph. By using this bias, a simple baseline can be constructed to predict the most frequent relationships, which alone was shown to produce reasonable results. Due to its success, several follow-up works incorporated this bias into their modeling framework. However, the frequency bias also resulted in subsequent scene graph generation methods producing biased scene graphs, where the models often ignore rare or granular relationships in favor of generic relationships due to the long-tail nature of relationship annotations in scene graph datasets. This issue will be discussed in further detail in Chapter 3. GraphRCNN [147], proposed a novel method that consisted of a relation proposal network as well as an attention-based Graph Neural Network (GNN) to capture contextual information. The purpose of the relation proposal network is to reduce the possible number of relation given a set of object nodes instead of having to consider all possible combinations pairs. FactorizableNet [66], improved the efficiency of scene graph generation by decomposing the representation into sub-graph based on clustering to tackle the problem of quadratic object combinations. SGTR [64] proposed a transformer based end-to-end framework that build on top of the DETR [15] architecture. While some methods have proposed architectural improvements, others have suggested novel training frameworks to enhance the quality of predicted scene graphs. Tang et al. [128] proposed the use of a causal inference framework to debias the prediction of the model obtained from biased training. They showed that applying their method on various baseline allowed to accurately capture relations with relatively fewer annotations. Knyazev et al. [57], similarly proposes a graph density-aware loss to address the relation annotation imbalance to improve performance on zero-shot and few-shot compositions of objects and relations. In this thesis, we will explore an energy-based formulation for training scene graph generation models (Chapter 3) as well as propose a method to achieve granularity beyond bounding box localization in the form of segmentation mask for existing scene graph dataset where such annotation are absent (Chapter 4).

### 2.2 Novel View Synthesis

Novel view synthesis entails the generation of images from previously unseen viewpoints by leveraging observations of a scene (see Figure 1.2). Methods for novel view synthesis can be broadly categorized based on their reliance on geo-
metric information, encompassing approaches rendering novel views without geometric and method that rely on geometry implicitly or explicitly.

Methods that achieve novel view synthesis without relying on explicit geometry adopt a direct learning approach to grasp the underlying plenoptic function. The *plenoptic function* [1], a multidimensional construct, models the intensity of light rays emitted by cameras from every conceivable location and angle, forming the bedrock of these techniques. In its most generic form, the plenoptic function is a 7D function, mapping a 3D location, 2D view direction, wavelength (scalar), and time (scalar) to the radiance of the ray passing through the specified location in the given direction at the time of observation. However, for static scenes with fixed lighting conditions, the plenoptic function’s dimensions can be reduced to 5, depending solely on the location and viewing direction. Moreover, in free space, the radiance observed along the ray remains constant, eliminating an additional degree of freedom and resulting in a 4D function. Light field [63] and lumigraph rendering [33] employ such 4D function to synthesize novel views of a scene. With the advent of neural networks, learning based methods [13, 48, 117, 119, 139] explored the use of light field representation as input to neural networks to render new views. These methods, however, either require dense input sampling [48], have a limited range of motion [119] or are limited to simple scenes [117].

Geometry-based methods in novel view synthesis can be classified into two categories: those that aggregate multiple input views to synthesize a novel view and those that learn an implicit representation of the underlying scene through the weights of a neural network. Chen and Williams [19] uses dense optical flow between two images to reconstruct images from arbitrary viewpoints. Seitz *et al.* [109] introduced view morphing that interpolates new views by pre-warping the images and linearly combining them followed by an post-warping stage. Recently, neural network based approaches have been employed to learn a representation of the scene. These methods can be further classified into methods that employ an *explicit* representation and methods that learn an *implicit* representation encoded into the weights of the neural network. *Explicit* representation-based methods use differentiable rendering to learn 3D representations such as point clouds [4, 104, 140], meshes [130] or voxels [74, 115] or 3D gaussians [52] for the scene. *Implicit* representation-based methods represent scenes using continuous coordinate-based functions such as signed distance fields [5, 16, 32, 44, 145, 151] or occupancy fields [83, 90]. Scene Representation Networks [116] use a differentiable ray marching algorithm along with a continuous function that maps coordinates to features. NeRF [85] achieves photo-realistic rendering by learning a function that maps points along a ray to color and opacity followed by volumetric rendering. NeX [138] is a multiplane image-based scene representation that addresses NeRF’s difficulty to model large view dependent effects. DVGo [122] proposed using a
Chapter 2. Related Work

2.1 Implicit Representation Methods

Density voxel grid to represent scene geometry and a feature voxel grid along with a neural network to present the color and view dependant effect. Plenoxels [29] proposed using a sparse grid structure with along with spherical harmonics to represent the scene. Instant-NGP [86] proposed a multi-resolution hash encoding that is decoded using a small MLP to obtain the geometry and appearance. The implicit representation based methods discussed so far require over-fitting to a single scene for accurate reconstruction. Another avenue of research involves training models on a large dataset of scenes and leveraging them to render novel views of previously unseen scenes without the need for additional training. Stereo Radiance Fields [21] introduced method inspired by classical stereo algorithm. Their approach project 3D points to reference views and process feature from these views in pairs to obtain the final rendering. PixelNeRF [153] conditions a NeRF [85] on deep convolutional visual features of the reference views, enabling generalization to new scenes; however, it uses absolute positions and directions as inputs to the NeRF, which generalizes poorly across scenes. Similarly, IBRNet [136] also uses deep features and NeRF-like volume rendering, but it learns to blend colours from neighboring views for each point along a ray. IBRNet uses the difference between view directions as MLP inputs; while this is superior to absolute coordinates, the relative view directions still depend on a global reference frame which is scene-specific. MVSNeRF [17] constructs a cost volume from deep visual features.

Aforementioned works, both in the over-fitting and generalization setup, have bought us closer to photorealistic view synthesis. However, several challenges still remain. In Chapter 5, we investigate the limitations of current methods in representing scenes with complex Lambertian effects and propose an approach to tackle this challenge. In Chapter 6, we delve into the generalization setup and introduce a method inspired by classical image-based rendering techniques. Our approach renders new views solely based on patches from reference views, which enhances the generalization quality and eliminates the need for fine-tuning, ensuring faithful reconstruction.

2.3 Layered Scene Representation

A layered scene representation decomposes a frame into a set of semantically meaningful layers (see fig. 1.3). There is a large body of work on decomposing videos into layers based on appearance and motion cues [11, 28, 47, 50, 51, 60, 99, 100, 110, 135]. Layered video representations have proven useful for a variety of applications, including view synthesis for static scenes [120, 161], reflection removal [2, 3], segmentation [31], game deconstruction [118], and text-based video editing [7]. A recent work, Omnimatte [77, 78], aims to decompose a video
into layers grouping objects with their correlated effects, enabling users to perform editing tasks like object removal and retiming. However, as discussed in Section 7, these methods rely on a constrained background model, limiting their applicability to panning camera motions only. Subsequent works [49, 152] relax this assumption by using general image warps instead of homographies, but they still struggle with significant parallax. Moreover, their focus is on learning a per-frame mapping to global sprites to enable consistent edits such as style transfer, rather than addressing object removal or video stabilization. In Chapter 7, we explore the task of layered decomposition from a causal monocular video and present a method that achieves accurate decomposition in videos where other methods face challenges.
Chapter 3

Energy-Based Learning for Scene Graph Generation

This chapter presents a new energy-based framework for training scene graph generation models, allowing for structural reasoning in the output space.

As discussed in Section 2.1, a typical scene graph generation model comprises of the object detection network, which extracts object regions and corresponding features, and a message passing network with nodes initialized with these region features and edges accounting for the potential relations among them. The features are refined, through context aggregation, and then classified to produce both object (node) and relation (edge) labels. These networks are often trained end-to-end by minimizing individual cross-entropy losses on both sets of labels. A major drawback of such an approach is that the quality of prediction (loss, e.g., encoded by cross-entropy) is simply proportional to the number of correctly predicted labels and ignores the rich structure of the scene graph output space (e.g., correlation or exclusion among object and relation label sets). In addition, the imbalance in the number of training samples for the relations results in dominant relations being heavily favored, leading to biased relation prediction at test time [128].

Figure 3.1 (b) illustrates the scene graph generated by a model [127] trained using the cross-entropy loss. Both the aforementioned drawbacks are apparent in the output. First, the model predicts a relation <man, riding, wave>. A simple examination of the rest of the scene graph reveals that such a relationship is impossible given that the man is on a rock and holding a surfboard. Second, the model tends to make generic relation predictions such as <man, on, rock> as opposed to more informative alternatives, e.g., <man, standing on, rock>.

The origin of these issues can be identified by examining the likelihood term. Cross-entropy based training treats objects (O) and relationships (R) in a scene graph as independent entities. This amounts to factorizing the likelihood of a scene graph (SG), given an image (I), as the product of the likelihoods for the individual objects and relations:

$$\log p(SG|I) = \sum_{i \in O} \log p(o_i|I) + \sum_{j \in R} \log p(r_j|I).$$  \hfill (3.1)
Eq. (3.1) brings to light the underlying cause of the problem highlighted above. First, during loss computation, the loss for each relation term is independent of the relations predicted in the rest of the scene graph. Thus an incorrect relation such as \(<\text{man, riding, wave}>\) is penalized the same as \(<\text{man, behind, wave}>\) irrespective of the other relations (\(<\text{man, on, rock}>\)). However, using common sense reasoning, we can determine that \(<\text{man, riding, wave}>\) is highly improbable given \(<\text{man, carrying, surfboard}>\) and should be penalized heavily as opposed to a likely, albeit incorrect, relation behind. Second, due to the summation over individual relation terms, the model, in order to minimize the loss, is incentivized to predict relations which are more common in training data.

While prior works have tried to address the issue of biased predictions [70, 128] in the context of scene graph generation, little progress has been made towards structured learning of scene graphs. In this chapter, we address both of these issues by proposing a novel generic loss formulation that incorporates the structure of scene graphs into the learning framework using an energy-based learning framework. This energy-based framework relies on a graph message-passing algorithm for energy computation, that is learned to model the joint conditional density of a scene graph, given an image. Such a formulation transforms the problem from maximizing sum of the individual likelihood terms to that of directly maximizing the joint likelihood of the objects and relations. Furthermore, this added structure acts as an inductive bias for the learning, allowing the model to efficiently learn relationship statistics from less data.

The proposed learning framework is general and hence can be used to train any off-the-shelf scene graph generation model. We experiment with various state-of-the-art models and demonstrate that our energy-based formulation achieves significant improvements in the performance over the corresponding models trained using the standard cross-entropy based formulation. We also demonstrate the enhanced generalization capability of models trained using our framework by evaluating zero shot relation retrieval performance. Finally, we demonstrate the ability of our energy-based framework to learn from lesser amounts of training data by demonstrating an improvement in relative performance when evaluating on few-shot relation triplets. Figure 3.1 (c), shows the scene graph generated by the proposed method. The generated scene graph is more granular, predicting relations such as \(<\text{man, standing on, rock}>\) as opposed to the biased and generic variant \(<\text{man, on, rock}>\). The model is also able to preclude improbable relations (e.g., \(<\text{man, riding, wave}>\)) and instead predicts in front of between man and wave.

Our main contribution is a novel energy-based framework for scene graph generation that allows for direct incorporation of structure into the learning. We also propose a novel message-passing algorithm that is used for computing the energy
Figure 3.1: **Scene Graph Generation**: Figure shows scene graphs generated by a VCTree [127] model trained using conventional cross-entropy loss (purple) and our proposed energy-based framework (green). We make two crucial observations. First, the model trained using cross-entropy loss is incapable of consistent structural reasoning (*riding* is not possible given the rest of the graph). Second, the trained model tends to be biased, favoring more frequent relations (e.g., *on*). Our proposed energy-based framework is designed, and able, to address these shortcomings.

of scene graph configurations. This message-passing algorithm is generic and can be used for other applications such as learning graph embeddings. Finally, we demonstrate the efficacy of our proposed framework by applying it to multiple state-of-the-art models and evaluating performance on two benchmark datasets - Visual Genome [59] and GQA [42] - where we consistently outperform the cross-entropy based counterparts by up to 21% on Visual Genome and 27% on GQA.
Figure 3.2: Model Overview of the Energy-based Learning. The region in light blue corresponds to most traditional scene graph generation pipelines. The proposed energy-based learning framework is highlighted in light green. We initialize the image graph with the extracted object proposal features as the node states. We instantiate the scene graph with predictions from traditional pipeline (or ground truth annotation). The image graph and scene graph are fed into the energy model where they undergo state refinement using a Gated Graph Neural Network and a novel Edged Graph Neural Network, respectively. We then obtain vector representations of each graph using pooling layers. The representations are concatenated and passed as input to a multi-layer perceptron which predicts the energy of the joint input (image) - output (scene graph) configuration. The loss is computed from the energy values of the ground truth and predicted configuration.

3.1 Approach

We first provide an overview of current approaches for scene graph generation based on the standard cross-entropy loss, followed by a description of our proposed energy-based learning framework along with the architecture used for energy computation.

3.1.1 Scene Graph Generation

Scene graph generation methods typically adopt a two-stage framework. In the first stage, given an image $I$, bounding boxes are obtained using a standard object detector such as Faster R-CNN [97]. Features corresponding to these regions are extracted using RoIAlign along with an initial distribution over object labels. In the next stage, these detections are used as inputs to predict the scene graphs.
The bounding-box features along with the object label and the spatial coordinates of the bounding boxes are used to initialize a set of node features. These features are refined using architectures such as LSTM [156], TreeLSTM [127] or Graph Attention Networks [147], to incorporate contextual information. Object labels, \( O \), are then obtained by classifying the refined features. Relationship labels, \( R \), are obtained by extracting features from union of object bounding boxes, followed by state refinement using BiLSTMs [156] or BiTreeLSTMs [127] and subsequent classification.

These models are trained using standard cross-entropy loss on object and relation labels. Each object and relationship is considered in isolation when computing individual losses, which are then summed up to obtain the loss for the given image. Such a loss formulation ignores the fact that objects and relations in a scene graph are interdependent. Intuitively, incorporating such dependencies into the learning procedure should lead to an improvement in performance. However, it is not clear how one can exploit the rich structure in the output space. Most methods ([127, 144, 147, 156]) attempt to find a way around this by employing message-passing algorithms in the input space that allow for aggregation of context information. However, this does not explicitly consider structure in the output space; neither in predictions nor in the loss function used for learning. We propose a novel energy-based learning framework that allows scene graph generation models to be trained using a “loss” that explicitly incorporates structure in the output space.

### 3.1.2 Energy Based Modeling

Energy-based models [62] encode dependencies between variables by assigning a scalar energy value to an input configuration. Given a data point \( x \in \mathcal{X} \) with corresponding label \( y \in \mathcal{Y} \), let \( E_\theta(x, y) \in \mathbb{R} \) be a joint energy function. While energy models map inputs to unnormalized densities, we can define a probability distribution via the Boltzmann distribution, \( p_\theta(x, y) = \frac{\exp(-E_\theta(x, y))}{Z_\theta} \), where \( Z(\theta) = \int \exp(-E_\theta(x, y)) \) is referred to as the normalization constant or partition function. Computing the normalization constant, \( Z_\theta \) for most parameterizations of the energy function is intractable. Therefore, learning the parameters \( \theta \) using methods such as maximum likelihood is not straightforward. Most methods address this problem by rewriting the derivative of the log likelihood as

\[
\nabla_\theta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(x', y')}[\nabla_\theta E_\theta(x, y')] - \nabla_\theta E_\theta(x, y),
\]

where the expectation is approximately estimated using MCMC methods that sample from the data distribution.

Unlike most prior works that train energy models for generative modelling, our focus is scene graph generation, a discriminative task. For such a task, we are only concerned with the relative energies of the various label configurations given
an input $x$. Training with a carefully crafted loss function circumvents the need for estimating the partition function or computing expectations. Therefore we can parametrize the energy function using an arbitrary neural network architecture. For a more detailed discussion on energy loss formulation refer to Section 2 in [61].

### 3.1.3 Energy Models for Scene Graphs Generation

We now describe our proposed energy-based learning framework for scene graph generation. In our formulation of the energy function, the data space $\mathcal{X}$ is the set of images $\mathcal{X} \in \mathbb{R}^{W \times H \times 3}$ and the label space $\mathcal{Y}$ is the set of scene graphs $SG$. The scene graph, $SG$, is defined by a tuple $(O, R)$, where $O \in \mathbb{R}^{n \times d}$ is the set of object labels and $R \in \mathbb{R}^{n \times n \times d'}$ is the set of relationship labels; $n$ is the number of objects in an image; $d$ and $d'$ is the total number of possible object and relation labels in the dataset.

A simple implementation of the joint energy function, would take an encoding of the image, $\mathcal{I}$, and a scene graph, $SG$, and produce a scalar energy value. However, there are a few challenges with this. First, a simple global CNN-based encoding of the image may fail to highlight, potentially small, regions relevant for scene graph prediction. Second, scene graph representation is variable in length ($n$ is not fixed) and high dimensional. The second challenge can be addressed by pooling object $O$ and relations $R$ across $n$ and $n \times n$ dimensions respectively. We propose a more sophisticated and effective scene graph refinement and gated pooling formulation in Section 3.1.4. To facilitate the former challenge of image encoding, we extract a graph based representation from the image. This representation is henceforth referred to as an image graph ($\mathcal{G}_I$). The nodes of the image graph are instantiated using features extracted from the object bounding boxes.

Given a scene graph generation model $M$ and an image $\mathcal{I}$, we predict a scene graph, $G_{SG}^0$ and compute the image graph $\mathcal{G}_I$. The scene graph along with the image is provided as input to the energy model ($E_\theta$) to compute the energy corresponding to the predicted configuration. Similarly, we compute the energy of the ground truth configuration using the ground truth scene graph ($G_{SG}^+$) and image graph ($\mathcal{G}_I^+$) constructed from ground truth bounding boxes. These two energy values are then used to compute the energy loss

$$L_e = E_\theta(G_{\mathcal{I}}^+, G_{SG}^+) - \min_{G_{SG} \in SG} E_\theta(G_{\mathcal{I}}, G_{SG}).$$

(3.2)

Computing this loss requires solving an optimization problem to find a scene graph configuration that minimizes the energy value (second term in Eq.(3.2)). We use Stochastic Gradient Langevine Dynamics (SGLD) to find such a scene graph configuration.
SGLD is a computational technique widely utilized in Bayesian optimization, notably with its close association with Markov Chain Monte Carlo (MCMC) methods. While traditional MCMC methods, like Metropolis-Hastings, propose candidate samples based on the entire dataset, SGLD leverages stochastic gradients computed from subsets of the data, resembling the mini-batch approach in stochastic optimization. SGLD incorporates Langevin dynamics, a process inspired by statistical mechanics, to simulate the behaviour of particles undergoing diffusion in a thermodynamic system. This diffusion process introduces a noise term facilitating an effective exploration of the parameter space while maintaining convergence towards the target distribution.

Starting from \( G_{SG}^0 \) we use SGLD [137] which approximately solves the optimization problem iteratively:

\[
O^{\tau+1} = O^\tau - \frac{\lambda}{2} \nabla_O E_\theta(G_T, G_{SG}^\tau) + \epsilon^\tau,
\]
\[
R^{\tau+1} = R^\tau - \frac{\lambda}{2} \nabla_R E_\theta(G_{II}, G_{SG}^\tau) + \epsilon^\tau
\]  \hspace{1cm} (3.3)

where \( O^\tau \) and \( R^\tau \) are the node and edge states in the scene graph and \( \epsilon^\tau \) is sampled from a normal distribution \( \mathcal{N}(0, \lambda) \).

Conceptually, the predicted \( G_{SG}^0 \) is used as an “initialization” to arrive at the low energy configuration through a series of steps defined by Eq.(3.3); each step similar to regular gradient descent with an added Gaussian noise. Differentiation through the optimization path guides parameter learning of the model \( M \) that generates \( G_{SG}^0 \) in the first place.

When training using the above loss, we observe that the energy values get arbitrarily large in magnitude leading to gradient overflow. We address this problem by adding an \( L2 \) regularization loss on the energy values:

\[
\mathcal{L}_r = E_\theta(G_T^+, G_{SG}^+)^2 + E_\theta(G_{II}, G_{SG})^2.
\]  \hspace{1cm} (3.4)

Finally, since the space of scene graphs is very high dimensional, we need to restrict the search space of the energy model in order to stabilize the learning. This is done by incorporating the task loss used by the underlying scene graph generation model, \( \mathcal{L}_t \) on the predicted output as an added regularization on the initial prediction. The total loss for training the scene graph generator and the the energy model is given by:

\[
\mathcal{L}_{total} = \lambda_e \mathcal{L}_e + \lambda_r \mathcal{L}_r + \lambda_t \mathcal{L}_t,
\]  \hspace{1cm} (3.5)

where \( \lambda_e, \lambda_r \) and \( \lambda_t \) are the relative weights.
3.1.4 Energy Model Architecture

Given an image graph \((G_I)\) and a scene graph \((G_{SG})\), the energy model first refines the state representations using graph neural networks. We use a novel Edged Graph Neural Networks (EGNN) and Graph Neural Network [67] on the scene graph and image graph respectively to incorporate contextual information. This is followed by applying a pooling layer on each graph to obtain a vector representation summarizing the graph states. Finally, these two vectors are fed into a multi layer perceptron (MLP) to compute the energy value corresponding to the predicted scene graph configuration. We then repeat these operations for the ground truth scene graph and the input image. The energy model can be parametrized as:

\[
E_\theta(G_I, G_{SG}) = \text{MLP} \left[ f(\text{EGNN}(G_{SG})); g(\text{GNN}(G_I)) \right],
\]

where \(f\) and \(g\) are pooling functions.

**Notation.** We use \(n_i\) to represent the features of the \(i^{th}\) node, which for \(G_{SG}\) corresponds to the \(i^{th}\) object, initialized to the corresponding \(i^{th}\) row of matrix \(O\). For \(G_I\) \(n_i\) corresponds to the \(i^{th}\) image region, initialized by the RoIAlign image features. We use \(e_{i\rightarrow j}\) to represent the feature of the directed edge from node \(i\) to \(j\), initialized to the \((i, j)^{th}\) column of \(R\). \(N_i\) denotes the neighbours of node \(i\).

### Edge Graph Neural Network

To allow for the direct application of convolution operations on graph representations accommodating edge features such as scene graphs, we propose a variant of graph message passing algorithm. For each node \(n_i\), we aggregate the message from neighbouring nodes and edges by:

\[
n_i^t = \alpha W_{nn} \left( \sum_{j \in N_i} n_j^{t-1} \right) + (1 - \alpha) W_{en} \left( \sum_{j \in N_i} e_{j \rightarrow i}^{t-1} \right),
\]

where \(W_{nn}\) and \(W_{en}\) are the kernel matrices for node-to-node and node-to-edge communication and \(0 \leq \alpha \leq 1\) is a hyper-parameter that controls the contribution of messages from edges and nodes. Similarly, the message passing for edges are given by:

\[
d_{i \rightarrow j} = W_{ee} [ n_i^{t-1} \parallel n_j^{t-1} ],
\]

where \(W_{ee}\) is the kernel matrix for node-to-edge communication. Note that the message passing for edges is direction aware i.e. \(d_{i \rightarrow j} \neq d_{j \rightarrow i}\). This is crucial
Table 3.1: **Quantitative Results.** We compare the proposed energy-based loss formulation against traditional cross-entropy loss using various state-of-the-art models. We report the mean Recall@K [127] under all three experimental setting.

as the relationship between two nodes change depending on direction of the edge for example `<cat, has, tail>` and `<tail, on, cat>`. The incoming messages are combined with the states using a gating mechanism [67].

### Pooling Layer

We use gated pooling layers to generate vector representations of the two graphs. The pooling operation is given by

\[
N = \sum_{k} f_{gate}(n_k) \odot n_k
\]  

\[
E = \sum_{ij} g_{gate}(e_{i\rightarrow j}) \odot e_{i\rightarrow j}
\]

where \(f_{gate}\) and \(g_{gate}\) are gating functions that map node and edge states to a scalar and \(\odot\) represents element-wise multiplication. These two vectors are then passed through a linear layer, after concatenation, to obtain the final vector representation of the graph. In the image graph \(G_I\), since there are no edge features, we use the pooled node vector \(N\) as the vector representation of the graph.

### 3.2 Experiments

We present experimental results on two datasets: the Visual Genome dataset [59] and the GQA dataset [42].
Visual Genome: We use the pre-processed version of the dataset from [144]. The dataset consists of 108k images and contains 150 object categories and 50 predicate categories. We use the original split with 70% of the images in the training set and the remaining 30% in the test set, with 5k images from the training set held out for validation [156].

GQA: The GQA dataset [42] is also constructed from images in the Visual Genome dataset. Starting from the scene graph annotations provided in Visual Genome, a normalization process is applied. This normalization process augments object and relation annotations and prunes inaccurate or unnatural relations. The resulting dataset contains a total of 1704 object categories, 311 relation categories. We use the same 70−30 split for the train and the test set, with 5k images in the validation set. Compared to Visual Genome, the GQA dataset has denser graphs with a larger number of object and relation categories.

3.2.1 Scene Graph Generation Models

The energy-based training introduced in this chapter is generic and does not make any assumptions on the underlying scene graph generation model. This allows freedom in choosing the model architecture for image-to-scene graph mapping. On the Visual Genome dataset, we experiment with VCTree [127], Neural Motifs [156] and Iterative Message Passing [144]. We also experiment with VCTree-TDE [128], where the inference involves counterfactual reasoning. On the GQA dataset, we experiment with Transformers [133], instead of VCTree, as the larger number of object classes in the GQA dataset leads to a considerably larger memory requirement in VCTree. The ability to experiment with different models demonstrates the versatility of our approach.

3.2.2 Evaluation

Relationship Recall (RR). We use the mean Recall@K (mR@K) metric [127] to evaluate the performance of the scene graph generation models. We report the mean Recall@K instead of the Regular Recall@K (R@K) due to data imbalance that leads to reporting bias as pointed out in recent works [128]. The evaluation is performed under three settings, (1) Predicate Classification (PredCls): Predict the relationship labels, given the image, object bounding boxes and object labels. (2) Scene Graph Classification (SGCls): Predict the object and predicate labels, given the image and bounding boxes and (3) Scene Graph Detection (SGDet): Predict the scene graph from the image.

Zero-Shot Recall (zsR@K). Introduced in [76], zsR@K evaluates the ability to
identify subject-predicate-object relation triplets that were not observed during training. We compute zsR@K for 3 settings: PredCls, SGCls and SGDet.

**Few-Shot Recall.** We introduce the few-shot Recall@K (fsR@K) metric that reports the Recall@K for relation triplets that occur a certain number of times in the training set. Unlike the conventional few-shot metric, we use a range of values to generate the few-shot triplet splits. Thus instead of splitting the triplets into 1-shot, 2-shot, etc., we split them into groups of 1−5-shot, 6−10-shot, etc. with triplets in a $k_1 - k_2$-shot occurring between $k_1$ and $k_2$ times.

**Sentence-to-Graph Retrieval (S2GR):** Introduced in [128], S2GR was designed to address the inability of RR (Relationship Recall) and zsR (Zero-Shot Recall) to capture graph-level coherence. In S2GR, the scene graph predicted for an image (obtained in the SGDet setting) is used as a semantic representation of the image. The task then is to retrieve images using their graph representation, with the image caption as a query. Note that the retrieval task is based solely on the detected scene graph and no other visual information. As a consequence, any bias in the scene graph generation will result in a decrease in the S2GR performance. Similar to [128], we report Recall@20/100 on 1k/5k gallery.

### 3.2.3 Implementation Details

**Detector.** We pre-train a Faster R-CNN [97] with ResNeXt-101-FPN [82, 143] backbone. We obtain the weights of the pre-trained detector for Visual Genome from [126]. For GQA, we pre-train the object detector using standard Faster-RCNN settings. The object detector has 28 mAP on Visual Genome and 10 mAP on GQA.

**Scene Graph Generator.** The baseline models, trained with standard cross-entropy loss, as well as our proposed framework, are trained using an identical setup. We use an SGD optimizer with an initial learning rate of $10^{-2}$. For the Visual Genome models, we incorporate frequency bias [156] into the training and inference. We do not use the frequency bias on GQA dataset, due to high memory requirements.

**Energy Model.** In the sampling step of SGLD (Eq. 3.3), we set the number of iterations ($\tau$) to 20 and the number of message passing iterations in EGN and GNN to 3. We use a step size of 1 and clip the gradients within $[-0.01, 0.01]$. After every gradient update step, we normalize the node and edge states of the scene graph to the range of $[0, 1]$.

**Sentence-to-Graph Retrieval.** We use the same formulation for S2GR as previous work [128]. The problem is formulated as a graph matching problem where the image caption is converted to graph structure using [59]. The scene graphs and
Table 3.2: Zero-shot Recall. The zero shot recall performance comparison of model trained using cross-entropy (CE) and energy-based loss (EB) on the Visual Genome (VG) and GQA dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Method</th>
<th>PredCls</th>
<th>SGCls</th>
<th>SGDet</th>
</tr>
</thead>
<tbody>
<tr>
<td>VG</td>
<td>PredCls</td>
<td>CE</td>
<td>1.43/4</td>
<td>.39/1.2</td>
<td>.19/.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>2.25/5.36</td>
<td>.87/1.87</td>
<td>.21/.54</td>
</tr>
<tr>
<td></td>
<td>SGCls</td>
<td>CE</td>
<td>1.28/3.56</td>
<td>.39/1.23</td>
<td>0/0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>2.07/4.87</td>
<td>.52/1.25</td>
<td>.11/.23</td>
</tr>
<tr>
<td></td>
<td>SGDet</td>
<td>CE</td>
<td>12.17/17.66</td>
<td>2.09/3.3</td>
<td>.14/.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>12.6/18.6</td>
<td>2.29/3.7</td>
<td>.16/.43</td>
</tr>
<tr>
<td>GQA</td>
<td>PredCls</td>
<td>CE</td>
<td>8.98/14.52</td>
<td>3.16/4.97</td>
<td>1.47/2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>9.58/15.14</td>
<td>4.18/6.38</td>
<td>1.62/2.68</td>
</tr>
<tr>
<td></td>
<td>SGCls</td>
<td>CE</td>
<td>17.74/30.61</td>
<td>1.27/2.16</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>19.47/33.45</td>
<td>1.49/2.48</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SGDet</td>
<td>CE</td>
<td>15.58/27.6</td>
<td>1.02/1.88</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EB</td>
<td>16.65/27.77</td>
<td>1.1/1.98</td>
<td>-</td>
</tr>
</tbody>
</table>

3.3 Experimental Results

Quantitative Results: Table 3.1 compares performance of various state-of-the-art methods trained using cross-entropy and our energy-based loss on two datasets, Visual Genome and GQA. We observe that training using the proposed energy loss leads to a consistent improvement in the mean Recall in all three tasks, for all the models. For the VCTree model, we obtain a relative improvement of 12.7%, 22.3% and 5.6% on mR@20 for PredCls, SGCls and SGDet respectively. We obtain relative improvements of similar magnitudes with the Motif, IMP and VCTree-TDE.
Chapter 3. Energy-Based Learning for Scene Graph Generation

Table 3.3: **Few-shot Recall@20.** Table compares the few short recall performance of a VCTree [127] model trained using cross-entropy and energy-based loss.

<table>
<thead>
<tr>
<th>$k_1 - k_2$ shot</th>
<th>1-5</th>
<th>6-10</th>
<th>11-15</th>
<th>16-20</th>
<th>20-25</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.E.</td>
<td>16.9</td>
<td>24.41</td>
<td>27.73</td>
<td>31.52</td>
<td>32.31</td>
</tr>
<tr>
<td>E.B.M.</td>
<td><strong>18.55</strong></td>
<td><strong>25.22</strong></td>
<td><strong>28.1</strong></td>
<td><strong>32.05</strong></td>
<td><strong>32.57</strong></td>
</tr>
</tbody>
</table>

models. On the GQA dataset, we present results of three models, Transformer, Motif and IMP. Similar to the Visual Genome dataset, we observe a consistent improvement in the mean Recall metric under PredCls and SGCls, with each of the models. We omit experiments on SGDet task due to low mAP of the underlying object detector. The proposed method leads to a relative improvement of 8.5% and 25.92% in the mR@20 metric for the PredCls and SGCls task when using the Transformer model for scene graph generation.

The absolute performance on the GQA dataset is lower, compared to Visual Genome dataset due to larger number of object and relationship classes. Additionally, due to memory constraints, we omit the highly effective [156] frequency prior on the GQA dataset.

**Zero-Shot Recall:** Table 3.2 reports zero-shot recall (zsR@20 and zsR@50) for all models. Similar to mR@K, we note consistent improvement in the zero-shot recall. We attribute this behaviour to our energy-based structure-aware framework, which facilitates the learning of models that are capable of performing global scene graph reasoning.

**Few-shot Recall:** In Section 3, we hypothesized that the energy-based learning bakes an inductive bias into the learning, thereby allowing the models to learn from fewer data. To test this hypothesis, we measure the few-shot Recall@20 for the VCTree model. We train a scene graph detection model using the proposed energy-based model as well as the standard cross-entropy loss. The result, as shown in Table 3.3, demonstrates that our training framework is able to provide a significant boost in performance in few-shot scenarios when less data is available. This shows that the energy formulation provides a data-efficient method for learning scene graph generation.

**Sentence-to-Graph Retrieval:** Table 3.4 compares the results of sentence-to-graph retrieval experiments. We use VCTree and Motif as our scene graph generation architecture. For each model, we observe relative improvements ranging from 5%-23% in the retrieval recall compared to the scene graphs generated using
Sentence to Graph Retrieval

<table>
<thead>
<tr>
<th>Gallery Size</th>
<th>1000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@20</td>
<td>R@50</td>
</tr>
<tr>
<td>VCTree</td>
<td>CE</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>EBM</td>
<td>17.2</td>
</tr>
<tr>
<td>Motif</td>
<td>CE</td>
<td>15.5</td>
</tr>
<tr>
<td></td>
<td>EBM</td>
<td>19.2</td>
</tr>
</tbody>
</table>

Table 3.4: **Sentence to Graph Retrieval.** We compare the scene graph retrieval performance on gallery of 1000 and 5000 images.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>mR@20</th>
<th>mR@50</th>
<th>mR@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-steps</td>
<td>14.18</td>
<td>18.14</td>
<td>19.66</td>
</tr>
<tr>
<td>20-steps</td>
<td>14.2</td>
<td>18.19</td>
<td>19.72</td>
</tr>
<tr>
<td>40-steps</td>
<td>14.37</td>
<td>18.18</td>
<td>19.81</td>
</tr>
<tr>
<td>60-steps</td>
<td>14.41</td>
<td>18.23</td>
<td>19.79</td>
</tr>
<tr>
<td>120-steps</td>
<td>14.59</td>
<td>19.29</td>
<td>20</td>
</tr>
<tr>
<td>No-Image-Graph</td>
<td>14.19</td>
<td>18.05</td>
<td>19.55</td>
</tr>
</tbody>
</table>

Table 3.5: **Ablation.** We experiment with the number of optimization steps ($\tau$) needed to estimate the energy loss and the effect of excluding image information in the energy model. All numbers were obtained using a VCTree [127] model.

This improvement can be attributed to the more coherent and informative scene graphs generated by our models.

**Ablation Studies:** We investigated the impact of optimization in Eq.(3.2) on the effectiveness of the energy model. We experimented with a different number of optimization steps while training a predicate classification model using VCTree. Note that increasing the number of iterations corresponds to more precise minima. We also study the effect of removing the image graph input to the energy model to determine the effectiveness of modelling the energy density over the joint space of the scene and image graph. Table 3.5 summarizes the mean recall@K for both of these experiments. We find that with an increase in the number of optimization steps in the energy loss, the mean recall almost consistently increases. Intuitively,
this is expected as a larger number of optimization steps means that we have a better chance of convergence when trying to find the minima in Eq.(3.2). This increase, however, comes with the added computation overhead and increase in training time. Similarly, we note that by removing the image information from the joint modelling, there is a drop in performance as the model is now forced to learn from only the scene graph labels.

**Qualitative Results:** We visualize the qualitative results obtained from a VCTree model trained using the proposed energy-based framework as well as cross-entropy loss in Figure 3.3. The top two rows show results from the regular relation retrieval task. We observe that the models trained using our proposed framework can consistently generate instructive relationships such as mounted on, parked on, walking on, standing on as opposed to the less informative and biased variant on generated by the baseline model. Similarly, in the top-left image, the energy-based training generates spatially informative relations such as <cat, in front of, door> instead of <cat, near, door> and <cat, looking at, dog> as opposed <dog, near, cat>. The bottom row shows the results of zero-shot relation retrieval. In the first image, due to the triplets of elephants with glasses not being present in the training data, the baseline model predicts <women, wearing, glasses> whereas our method generates the accurate prediction <elephant, has, glasses>.
3.4 Conclusion and Discussion

We present a novel model-agnostic energy-based learning framework for training scene graph generation models. Unlike cross-entropy-based training, the proposed method embraces structure in the output space allowing the model to perform structure-aware learning. We show that scene graph generation models can benefit from the proposed training framework by performing experiments on the Visual Genome and GQA datasets. We observe significant improvement in performance as compared to traditional cross-entropy-based training. We also exhibit the generality and efficiency of our model through experiments in zero-shot and few-shot relationship settings. Finally, the proposed method does not make any assumptions about the underlying generation model and can be easily used with any model.

While the proposed energy-based modelling leads to an improvement in performance, it comes with an added computational overhead that slows down the training (up to 4x) and increases the memory requirement (up to 2x). As opposed to a cross-entropy-based method that only requires a single forward pass during each step of training, our method has to perform multiple iterations of Stochastic Gradient Langevine Dynamics (SGLD) to estimate the energy loss term. An alternative formulation, in light of recent developments in diffusion models, would be to use a diffusion loss to find stable scene graph configurations with low energy. This would require modelling the diffusion process in graph domain, which can be a challenging.
Chapter 4

Segmenation-grounded Scene Graph Generation

In the previous chapter, we presented a new method that can be applied to any existing scene graph generation method to address the lack of structure-based reasoning capability. In this chapter, we explore another limitation of an existing method. Scene graph generation methods ground nodes (objects) and edges (relations) to (rectangular) bounding boxes produced by the object proposal mechanism directly (e.g., pre-trained as part of R-CNN) or by taking a union of bounding boxes of objects involved in a relation. A more granular and accurate pixel-level grounding would naturally be more valuable. This has been shown to be the case in other visual and visual-lingual tasks (e.g., referring expression comprehension [41, 72], video segmentation with referring expression [53] and instance segmentation with Mask-RCNN [36]). In addition, grounding to segmentations could improve the overall performance of the scene graph generation by focusing node and edge features on irregular regions corresponding to objects or interface between objects, constituting an interaction. The method address in this chapter is aimed at achieving this level of granularity in the generated scene graphs.

Pixel-level grounding, however, comes with a number of unique challenges. The foremost of which is that traditional scene graph datasets, such as Visual Genome [59], do not come with instance-level segmentation annotations. This makes it impossible to employ a traditional fully supervised approach. Further, even if we were to collect segmentation annotations, doing so for a large set of object types typically involved in Scene Graph predictions would be prohibitively expensive*. To address this, we propose a transfer and multi-task learning formulation that uses an external dataset (e.g., MS COCO [68]) to provide segmentation annotations for some categories; while leveraging standard scene graph dataset (e.g., Visual Genome [59]) to provide graph and bounding box annotations for the target task.

On a technical level, for a target object that lacks segmentation annotations, its mask is expressed as a weighted linear combination over categories for which an-

*As per [9], labelling one image in VOC [26] takes 239.7 seconds.
**Chapter 4. Segmentation-grounded Scene Graph Generation**

Figure 4.1: **Segmentation-grounded scene graph generation.** The image on top is the output of an existing scene graph generation method [127]. The bottom image is the output of augmenting our approach to [127]. The effective grounding of objects to pixel-level regions within the image leads to better relation predictions.

notations are present in an external dataset. This transfer is realised by leveraging the linguistic similarities between the target object label and these supervised categories, thus enabling grounding objects to segmentation masks without introducing any annotation cost. For a pair of objects that share a relation, our approach additionally employs a Gaussian masking mechanism to assign this relation to a pixel-level region within the image. Through joint optimization over the tasks of scene graph and segmentation generation, our approach achieves simultaneous improvements over both tasks. Our proposed method is end-to-end trainable and can be easily integrated into any existing scene graph generation method (e.g., [127, 156]).

Our foremost contribution is that we propose the first, to our knowledge, a framework for pixel-level segmentation-grounded scene graph generation/prediction, which can be integrated with any existing scene graph generation method. For objects, these groundings are realised via segmentation masks, which are computed through a lingual-similarity-based zero-shot transfer mechanism over categories in an auxiliary dataset. To effectively ground relations at a pixel level, we additionally propose a novel Gaussian masking mechanism over segmentation masks. Finally, we demonstrate the flexibility and efficacy of our approach by augmenting it to existing scene graph architectures and evaluating performance on the Visual Genome [59] benchmark dataset, where we consistently outperform baselines by up to 12% on relation prediction.
Figure 4.2: **Model Architecture.** For an image, the object detector provides a set of bounding boxes, and for each box, additionally generates instance-level segmentations via a zero-shot transfer mechanism. These inferred segmentation masks are incorporated into the nodes and edges of the underlying graph, before passing it into an existing scene graph prediction architecture like [127, 156]. The inferred segmentation masks are additionally refined by leveraging the global context captured by the context aggregation step of the scene graph prediction method. The proposed method is end-to-end trainable and can be augmented to any existing scene graph method.

### 4.1 Approach

We propose a novel multi-task learning framework that leverages instance-level segmentation annotations, obtained via a zero-shot transfer mechanism, to effectively generate pixel-level groundings for the objects within a scene graph. Our approach, highlighted in Figure 4.2, builds on existing scene graph generation methods but is agnostic to the underlying architecture and can be easily integrated with existing state-of-the-art approaches.

#### 4.1.1 Notation

Let $\mathcal{D}^g = \{(x^g_i, G^g_i)\}$ denote the dataset containing graph-level annotations $G^g_i$ for each image $x^g_i$. We represent the scene graph annotation $G^g_i$ as a tuple of object and relations, $G^g_i = (O^g_i, R^g_i)$, where $O^g_i \in \mathbb{R}^{n_i \times d_g}$ represents object labels and $R^g_i \in \mathbb{R}^{n_i \times n_i \times d'_g}$ represents relationship labels; $n_i$ is the number of objects in an image $x^g_i$; $d_g$ and $d'_g$ are the total number of possible object and relation labels, respectively, in the dataset.

In addition, we assume availability of the dataset $\mathcal{D}^m = \{(x^m_i, M^m_i)\}$, where each image $x^m_i$ has corresponding instance-level segmentation annotations $M^m_i$. Finally, $d_m$ are the total number of possible object labels in $\mathcal{D}^m$.

As is the case with existing scene graph datasets like Visual Genome [59],
$D^g$ does not contain any instance-level segmentation masks. Also, $D^m$ can be any dataset (like MS COCO [68]). Note, that in general, the images in the two datasets, $D^g$ and $D^m$, are disjoint and the object classes in the two datasets may have minimal overlap (e.g., MS COCO provides segmentations for 80, while Visual Genome provides object bounding boxes for 150 object categories\(^1\)).

For brevity, we drop subscript $i$ for the rest of the chapter.

### 4.1.2 Scene Graph Generation

Given an image $x^g \in D^g$, a typical scene graph model defines the distribution over the scene graph $G^g$ as follows,

$$
Pr(G^g|x^g) = Pr(B^g|x^g) \cdot Pr(O^g|B^g, x^g) \cdot Pr(R^g|O^g, B^g, x^g) \tag{4.1}
$$

The bounding box network $Pr(B^g|x^g)$ extracts a set of boxes $B^g = \{b^g_1, \ldots, b^g_n\}$ corresponding to regions of interest. This can be achieved using standard object detectors such as Faster R-CNN [97] or Detectron [141]. Specifically, these detectors are pretrained on $D^g$ with the objective to generate accurate bounding boxes $B^b$ and object probabilities $L^g = \{l^g_1, \ldots, l^g_n\}$ for an input image $x^g$. Note that this only requires access to the object (node) annotations in $G^g$.

The object network $Pr(O^g|B^g, x^g)$, for each bounding box $b^g_j \in B^g$, utilizes feature representation $z^g_j$, where $z^g_j$ is computed as $\text{RoIAlign}(x^g, b^g_j)$, which extracts features from the area within the image corresponding to the bounding box $b^g_j$. These features, alongside object label probabilities $l^g_j$, are fed into a context aggregation layer such as Bi-directional LSTM [156], Tree-LSTM [127], or Graph Attention Network [147], to obtain refined features $z^{g,g}_j$. These refined features are used to obtain the object labels $O^g$ for the nodes within the graph $G^g$. Similarly, for the relation network $Pr(R^g|O^g, B^g, x^g)$, features corresponding to union of object bounding boxes are refined using message passing layers and subsequently classified to produce predictions for relations.

Existing models ground the objects in the scene graph to rectangular regions in the image. While bounding boxes provides an approximate estimate of the object locations, having a more granular pixel-level grounding achievable through segmentation masks is much more desirable. A major challenge is the lack of segmentation annotation in scene graph datasets like Visual Genome [59]. Furthermore, manually labelling segmentation masks for such large datasets is both time-consuming and expensive. As a solution, we derive segmentation masks via a zero-shot transfer mechanism from a segmentation head trained on an external dataset $D^m$ (e.g., MSCOCO [68]). This inferred segmentation mask is then used as

---

\(^1\)Visual Genome has a $\sim 47\%$ image overlap with MS-COCO. However, they have different object categories and annotations. We make no use of this implicit image overlap in our formulation.
additional input to the object and relation networks to generate better scene graphs. Our approach factorizes the distribution over $G^g$ as,

$$
\Pr(G^g|x^g) = \Pr(B^g|x^g) \cdot \Pr(M^g|x^g) \cdot \Pr(O^g|B^g, M^g, x^g) \\
\cdot \Pr(R^g|O^g, B^g, M^g, x^g)
$$

(4.2)

where $M^g = \{m^g_1, \ldots, m^g_n\}$ are the inferred segmentation masks corresponding to the bounding boxes $B^g$. Such a factorization enables grounding scene graphs to segment masks and affords easy integration to existing architectures.

### 4.1.3 Segmentation Mask Transfer

For each image $x^g \in D^g$, we derive segmentation masks $M^g$ using annotations learned over classes in an external dataset $D^m$. To facilitate this, like described in Section 4.1.2, we pretrain a standard object detector (like Faster R-CNN [97]) on the scene graph dataset $D^g$. However, instead of training the detector just on images in $D^g$, we additionally jointly learn a segmentation head $f_M$ on images in $D^m$. Note that when training the object detector jointly on images in $D^g$ and $D^m$, the same backbone and proposal generators are used, thus reducing the memory overhead.

For an image $x^g \in D^g$, let $z^g_j$ be the feature representation for a bounding box $b^g_j \in B^g$. Let, $m^g_j = f_M(z^g_j)$,

where $m^g_j \in \mathbb{R}^{d_m \times m \times m}$, $d_m$ represents the number of classes in $D^m$, and $m$ is the spatial resolution of the mask. Per class segmentation masks $m^g_j \in \mathbb{R}^{d_g \times m \times m}$ are then derived from $m^g_j$ using a zero-shot transfer mechanism. Let $S \in \mathbb{R}^{d_g \times d_m}$ be a matrix that captures linguistic similarities between classes in $D^g$ and $D^m$. For a pair of classes $c_g \in [1, d_g]$, $c_m \in [1, d_m]$, the element $S_{c_g, c_m}$ is defined as,

$$
S_{c_g, c_m} = g^\top_{c_g} g_{c_m}
$$

(4.3)

where $g_{c_g}$ and $g_{c_m}$ are 300-dimensional GloVe [91] vector embeddings for classes $c_g$ and $c_m$ respectively. $m^g_j$ is then obtained as a linear combination over $m^g_j$ as follows,

$$
\hat{m}^g_j = S^\top \bar{m}^g_j
$$

(4.4)

Note that such a transfer doesn’t require any additional labelling cost as we rely on a publicly available dataset $D^m$.

\[\text{Note that } d_g \gg d_m \text{ in our case.}\]

\[\text{For class names that contain multiple words, individual GloVe word embeddings are averaged.}\]
4.1.4 Grounding Nodes to Segmentation Masks

As mentioned in Equation 4.2, we incorporate the inferred segmentation masks in the object network \( \Pr(O^g|B^g, M^g, x^g) \) to ground objects in \( D^g \) to pixel-level regions within the image.

Specifically, for a particular image \( x^g \), the model \( \Pr(B^g|x^g) \) outputs a set of bounding boxes \( B^g \). For each bounding box \( b^g_j \in B^g \), it additionally computes a feature representation \( z^g_j \) and object label probabilities \( l^g_j \in \mathbb{R}^{d_g+1} \). Following the procedure described in Section 4.1.3, per-class segmentation masks \( m^g_j \) are inferred for each bounding box \( b^g_j \). We define a segmentation aware representation \( \hat{z}^g_j \) as,

\[
\hat{z}^g_j = f_N([z^g_j, m^g_j]) \tag{4.5}
\]

where \( f_N \) is a learned network and \([\ldots]\) represents concatenation. Contrary to existing methods like [127, 156] that use the segmentation agnostic representation \( z^g_j \), we feed \( \hat{z}^g_j \) and \( l^g_j \) as inputs to the object network \( \Pr(O^g|B^g, M^g, x^g) \).

4.1.5 Grounding Edges to Segmentation Masks

To facilitate better relation prediction, we leverage the inferred segmentation masks in the relation network \( \Pr(R^g|O^g, B^g, M^g, x^g) \). Specifically, for a pair of objects, we utilize a Gaussian masking mechanism to identify relation identifying pixel-level regions within an image.

Given a pair of bounding boxes \( (b^g_j, b^g_{j'}) \in B^g \) that contain a possible edge and their corresponding object label probabilities \( (l^g_j, l^g_{j'}) \), their respective segmentation masks \( (m^g_j, m^g_{j'}) \) are computed via the procedure described in Section 4.1.3. We define \( z^g_{j,j'} \) as the segmentation agnostic feature representation representing the union of boxes \( (b^g_j, b^g_{j'}) \), which is computed as \( \text{RoIAlign}(x^g, b^g_j \cup b^g_{j'}) \).

Contrary to existing works that rely on this coarse rectangular union box, our approach additionally incorporates an intersection of the segmentation masks \( (m^g_j, m^g_{j'}) \) to provide more granular information. To this end, we define the union segmentation mask \( m^g_{j,j'} \) as,

\[
m^g_{j,j'} = (K_j \ast m^g_j) \odot (K_{j'} \ast m^g_{j'}) \tag{4.6}
\]

where \( \ast \) is the convolution operation, and \( \odot \) computes an element-wise product. \( K_j, K_{j'} \) are \( \delta \times \delta \) sized Gaussian smoothing spatial convolutional filters parameterized by variances \( \sigma_x^2, \sigma_y^2 \) and correlation \( \rho_{x,y} \).

\(^1\text{RoIAlign}(x^g, b^g_j \cup b^g_{j'})\) computes the convex hull of the union of the two boxes.
These parameters are obtained by learning a transformation over the object label probabilities \( l^g_j \). Specifically, \( \sigma^2_x, \sigma^2_y, \rho_{x,y} = f_N (l^g_j) \), where \( f_N \) is a learned network. \( K_j' \) is computed analogously using \( l^g_j' \). The attended union segmentation mask \( m^g_{j,j'} \) affords the computation of a segmentation aware representation \( \hat{z}^g_{j,j'} \) as follows,

\[
\hat{z}^g_{j,j'} = f_E ([z^g_{j,j'}, m^g_{j,j'}])
\]

(4.7)

where \( f_E \) is a learned network. \( \hat{z}^g_{j,j'} \) is then used as an input to the relation network \( Pr (R^g | O^g, B^g, M^g, x^g) \).

### 4.1.6 Refining Segmentation Masks

As described previously, our proposed approach incorporates segmentation masks to improve relation prediction. However, we posit that the tasks of segmentation and relation prediction are indelibly connected, wherein an improvement in one leads to an improvement in the other.

To this end, for each object \( b^g_j \in B^g \), in addition to predicting the object labels \( O^g \), we learn a segmentation refinement head \( f_{M'} \) to refine the inferred segmentation masks \( m^g_j \). However, as the scene graph dataset \( D^g \) does not contain any instance-level segmentation annotations, training \( f_{M'} \) in a traditionally supervised manner is challenging.

To alleviate this issue, we again leverage the auxiliary dataset \( D^m \), which contains segmentation annotations. For an image \( x^m \in D^m \) we compute the bounding boxes \( B^m \). Note that this does not require any additional training as the object detector is jointly trained using both \( D^g \) and \( D^m \) as described in Section 4.1.3.

For a bounding box \( b^m_j \in B^m \), the corresponding per class masks are computed as, \( m^m_j = f_M (z^m_j) \), where \( z^m_j \) is the feature representation for \( b^m_j \), and \( f_M \) is the segmentation head defined in Section 4.1.3. The refined mask \( \hat{m}^m_j \) is then computed as,

\[
\hat{m}^m_j = m^m_j + f_{M'} (z^{o,m}_j)
\]

(4.8)

where \( z^{o,m}_j \) is the representation computed by the context aggregation layer within the object network \( Pr (O^m | B^m, M^m, x^m) \). Note that this network is identical to the one defined in Equation 4.2. The segmentation refinement head \( f_{M'} \) is a zero-initialized network that learns a residual update over the mask \( m^m_j \). As ground-truth segmentation annotations are available for all objects \( B^m \), \( f_{M'} \) is trained using a pixel-level cross-entropy loss.
Chapter 4. Segmentation-grounded Scene Graph Generation

\( f_{M'} \) is trained alongside the scene graph generation model, and the refined masks are used during inference to improve relation prediction performance. Specifically, for a particular image \( x^g \in D^g \), we follow the model described in Equation 4.2 to generate predictions. However, instead of directly using the inferred masks obtained using the zero-shot formulation in Section 4.1.3, we additionally refine it using \( f_{M'} \). For a particular mask \( m^g_j \) corresponding to a bounding box \( b^g_j \), we compute \( \hat{m}^g_j \) as,

\[
\hat{m}^g_j = m^g_j + f_{M'}(z^{o,g}_j)
\]

where \( z^{o,g}_j \) is the representation computed by the context aggregation layer. The refined mask is used in the object and relation networks as described in Sections 4.1.4 and 4.1.5.

4.1.7 Training

Our proposed approach is trained in two stages. The first stage involves pre-training the object detector to enable bounding box proposal generation for a given image. Given datasets \( D^g \) and \( D^m \), the object detector is jointly trained to minimize the following objective,

\[
L^{obj} = L^{rcnn} + L^{seg}
\]

where \( L^{rcnn} \) is the Faster R-CNN [97] objective, and \( L^{seg} \) is the pixel-level binary cross entropy loss [36] applied over segmentation masks. Note that images in \( D^g \) do not contribute to \( L^{seg} \) due to lack of segmentation annotations.

The second stage of training involves training the scene graph generation network to accurately identify relations between pairs of objects. Given datasets \( D^g \) and \( D^m \), the scene graph generation network is jointly trained to minimize the following objective,

\[
L = L^{sg} + L^{seg}
\]

where \( L^{sg} \) depends on the architecture of the underlying scene graph method our approach is augmented to. For example, in the case of MOTIF [156], \( L^{sg} \) consists of two cross-entropy losses, one to refine the object categorization obtained from the pretrained detector, and the other to aide with accurate relation prediction. \( L^{seg} \) is identical to the segmentation loss described in Equation 4.10, and is used to learn the refinement network \( f_{M'} \) (Section 4.1.6). As images in \( D^m \) do not contain scene graph annotations, they only contribute to \( L^{seg} \). Similarly, images in \( D^m \) only affect \( L^{seg} \).
**Chapter 4. Segmentation-grounded Scene Graph Generation**

### Table 4.1: Scene Graph Prediction on Visual Genome.

Mean Recall (mR) is reported for three tasks, across two detector backbones. Our approach is augmented to and contrasted against MOTIF [156] and VCTree [127]. † denotes our re-implementation of the methods.

<table>
<thead>
<tr>
<th>Detector Method</th>
<th>Predicate Classification</th>
<th>Scene Graph Classification</th>
<th>Scene Graph Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMP [144]</td>
<td>mR@20 9.8 mR@50 10.5 mR@100 7.5</td>
<td>- 5.8 6.0</td>
<td>- 3.8 4.8</td>
</tr>
<tr>
<td>MOTIF† [156]</td>
<td>- 10.8 14.0 15.3 6.5 7.7 8.2 4.2 5.7 6.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCTree† [127]</td>
<td>- 14.0 17.9 19.4 8.2 10.1 10.8 5.2 6.9 8.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Detector Method

- **IMP [144]**
  - VGG-16 [114]
  - mR@20 9.8, mR@50 10.5, mR@100 7.5
- **MOTIF† [156]**
  - VGG-16 [114]
  - mR@20 10.8, mR@50 14.0, mR@100 15.3
- **VCTree† [127]**
  - VGG-16 [114]
  - mR@20 14.0, mR@50 17.9, mR@100 19.4
- **MOTIF† [156]**
  - VGG-16 [114]
  - Baseline 13.7, Seg-Grounded 14.6
- **VCTree† [127]**
  - VGG-16 [114]
  - Baseline 14.4, Seg-Grounded 14.5

#### Detector Method

- **ResNeXt-101-FPN [82, 143]**
  - Baseline 13.7, Seg-Grounded 15.0

### Table 4.2: Segmentation Refinement on MSCOCO.

Standard COCO precision metrics are reported across three tasks and two detector backbones. Task formulation is identical to Table 4.1. ‘No Refine’ is the baseline where the segmentation masks are obtained from the pre-trained detector. As ground truth masks are unavailable in Visual Genome, evaluation on MSCOCO serves as a proxy.

<table>
<thead>
<tr>
<th>Detector Method</th>
<th>Predicate Classification</th>
<th>Scene Graph Classification</th>
<th>Scene Graph Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMP [144]</td>
<td>mR@20 9.8 mR@50 10.5 mR@100 7.5</td>
<td>- 5.8 6.0</td>
<td>- 3.8 4.8</td>
</tr>
<tr>
<td>MOTIF† [156]</td>
<td>- 10.8 14.0 15.3 6.5 7.7 8.2 4.2 5.7 6.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCTree† [127]</td>
<td>- 14.0 17.9 19.4 8.2 10.1 10.8 5.2 6.9 8.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Detector Method

- **IMP [144]**
  - VGG-16 [114]
  - mR@20 9.8, mR@50 10.5, mR@100 7.5
- **MOTIF† [156]**
  - VGG-16 [114]
  - mR@20 10.8, mR@50 14.0, mR@100 15.3
- **VCTree† [127]**
  - VGG-16 [114]
  - mR@20 14.0, mR@50 17.9, mR@100 19.4
- **MOTIF† [156]**
  - VGG-16 [114]
  - Baseline 13.7, Seg-Grounded 14.6
- **VCTree† [127]**
  - VGG-16 [114]
  - Baseline 14.4, Seg-Grounded 14.5

#### Detector Method

- **ResNeXt-101-FPN [82, 143]**
  - Baseline 13.7, Seg-Grounded 15.0

### 4.2 Experiments

We perform experiments using two datasets: the Visual Genome Dataset [59] and the COCO dataset [68].

#### Visual Genome.

For training and evaluating the scene graph generation performance, we use the Visual Genome dataset [59]. We use the widely adopted prepossessed version of Visual Genome from [144]. This subset contains 108k images across 150 object categories and 50 relation labels. Images with more that 40 object bounding boxes are filtered out from the test set due to memory constraints.

#### MS-COCO

For training and evaluating the segmentation masks, we use the MSCOCO 2017 dataset [68], which contains 123k images, split into 118k training and 5k validation images, across 80 categories. As the ground-truth annotations for the test set are not available, as is common practice, results are reported on the validation
set.

Note that our approach is agnostic to the choice of the auxiliary dataset \( D^m \). The choice of using MS-COCO is motivated by its popularity in the community. As MS-COCO has images in common with Visual Genome, there is a possibility of information leakage across the two datasets. However, due to the differences in annotation types, such leakage is not observed in practice. For simplicity, this image overlap is not removed when computing the results described in Section 4.3.

4.2.1 Scene Graph Generation Model

Our proposed framework is generic and can be easily integrated with various scene graph generation models. We experiment with two scene graph architectures, namely MOTIF [156] and VCTree [127].

In MOTIF [156], the object and relation networks (Equation 4.1) are each instantiated by bidirectional LSTMs [40]. For an image \( x^g_j \in D^g \), the extracted bounding boxes \( B^g \) are arranged based on their \( x \)-coordinate position, and passed through the bidirectional LSTM networks. Instead of assuming a linear ordering between the objects, VCTree [127] generates a dynamic binary tree, with the aim of explicitly encoding the parallel and hierarchical relationships between objects. The object and relation networks are instantiated as bidirectional TreeLSTMs [124]. When augmenting our approach with MOTIF [156] and VCTree [127], we identically replicate the object and relation networks proposed in the respective works.

4.2.2 Evaluation

Relationship Recall (RR). To measure the performance of models we use the mean Recall \( @K \) (mR@K) metric introduced in [20, 127]. The mean Recall metric calculates the recall for predicate label independently across all images and then averages the result. We report the mean Recall instead of the conventional Regular Recall(R@K) due to the long-tail nature of relation labelling in Visual Genome that leads to reporting bias [128]. Mean Recall reduces the influence of dominant relationships such as on and has, and gives equal weight to all the labels in the dataset.

Zero-Shot Recall (zsR@K). Introduced in [76], zsR@K computes the Recall@K for subject-predicate-object triplets that are not present in the training data.

These evaluation metrics are computed for three different sub-tasks: 1) Predicate Classification (PredCls): predict the relation labels given the ground truth object bounding boxes and labels; 2) Scene Graph Classification (SGCls): predict the object and relation labels given the ground truth object bounding boxes; 3)
Scene Graph Detection (SGDet): given an image, predict the entire scene graph.

**Segmentation Precision.** As the Visual Genome dataset [59] does not contain any instance-level segmentation annotations, as a proxy we use the MSCOCO dataset [68] to measure the performance of the segmentation refinement procedure described in Section 4.1.6. To make the evaluation similar to scene graph generation, we analogously define three sub-tasks to measure the improvement in the quality of segmentation masks. These sub-tasks, namely (PredCls), (SGCls), and (SGDet), are identical to the ones defined earlier. For each of these sub-tasks, the standard evaluation metrics on COCO are reported [36].

### 4.2.3 Implementation Details

**Detector.** For our detector architecture, we use the two-stage Faster-RCNN [97] framework. To demonstrate the flexibility of our approach, we experiment with two different backbones within the Faster-RCNN framework: 1) VGG-16 [114] pre-trained on the ImageNet [105] dataset, and 2) ResNeXt-101-FPN [82, 143] backbone pre-trained on the MSCOCO [68] dataset. We first fine-tune the detector jointly on the Visual Genome and MSCOCO datasets, refining the classifiers and regressors, and simultaneously learning a segmentation network on images in MSCOCO. When training the scene graph generators, the detector parameters are freezed. Note that for the baselines, the detector is fine-tuned only on the Visual Genome, and hence no segmentation is learned.

**Scene Graph Models.** For training the scene graph models we use an SGD optimizer with an initial learning rate of $10^{-2}$. Following prior works, we integrate the frequency bias [156] into the training and inference procedure. During inference, in SGDet task, we filter pairs of objects that do not have any bounding box overlap for relation prediction.

### 4.3 Results

**Relationship Recall.** We report the mean Recall values comparing the baseline and proposed method in Table 4.1. To ensure a fair comparison, we additionally report the numbers obtained via re-implementing the baselines. Note that in case of MOTIF [156], our re-implementation provides significantly higher performance compared to the reported numbers in [156]. For both the MOTIF [156] and VC-Tree [127], irrespective of the backbone architecture, we observe a consistent improvement in the recall rate across all three tasks when incorporating our proposed approach.

For MOTIF [156], we observe an improvement of 7.0% on average at mR@20, 50 and 100 across all settings and backbones. Specifically, on the VGG back-
Chapter 4. Segmentation-grounded Scene Graph Generation

Figure 4.3: **Qualitative Results.** Visualizations of scene graphs generated by using VCTree [127] (in purple) and our approach augmented to VCTree (in green). The left two images contrast performance on relation retrieval. The right two images contrast performance on zero-shot relation retrieval, with the zero-shot triplet shown in yellow. Our approach additionally generates pixel-level object groundings.

bone [114], we obtain a relative improvement of 6.5%, 5.3%, and 7.7% on mr@20 across the three tasks. Similarly, for the ResNeXt-101-FPN [82, 143] backbone, we observe relative improvements of 2.8%, 11.2%, and 10.3% on mr@20. Similarly for VCTree [127], an average improvement of 12.6% is observed across tasks and backbones. We attribute the performance improvement to the ability of our model to effectively ground objects and relations to pixel-level regions, thus providing more discriminative features.

**Zero-Shot Recall.** We report the Zero-Shot Recall value zsR@20 and zsR@100 in Table 4.3. We observe an consistent improvement in zero-shot recall when using the proposed scene graph generation framework. Our method outperforms the baselines by an average of 94.5% and 97.9% on MOTIF and VCTree respectively.

**Segmentation Accuracy.** As segmentation annotations are not present in Visual Genome [59], we evaluate our proposed segmentation refinement on the MSCOCO dataset [68]. This provides a suitable proxy, wherein segmentation improvements on the MSCOCO dataset can be translated, to some degree, on Visual Genome. We report the standard COCO evaluation metrics, namely AP (averaged over IoU thresholds), AP$_{50}$, AP$_{75}$, and AP$_S$, AP$_M$, AP$_L$ (AP at different scales), on three different scene graph evaluation tasks in Table 4.2. ‘No Refine’ acts as a strong baseline, wherein the segmentation masks are generated from the pretrained detector. It is evident that our proposed segmentation refinement improves on mask quality across tasks and detector backbones. As the ground truth bounding boxes
Chapter 4. Segmentation-grounded Scene Graph Generation

<table>
<thead>
<tr>
<th>Model</th>
<th>Detector</th>
<th>Method</th>
<th>PredCls zsr@20/100</th>
<th>SGCls zsr@20/100</th>
<th>SGDet zsr@20/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOTIF†</td>
<td>VGG-16 [114]</td>
<td>BL</td>
<td>1.7/6.7</td>
<td>0.2/1.1</td>
<td>0.0/0.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG</td>
<td><strong>3.2/9.3</strong></td>
<td><strong>0.4/1.6</strong></td>
<td><strong>0.1/0.5</strong></td>
</tr>
<tr>
<td></td>
<td>ResNeXt-101-FPN [82, 143]</td>
<td>BL</td>
<td>1.9/7.2</td>
<td>0.3/1.2</td>
<td>0.0/0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG</td>
<td><strong>4.1/10.5</strong></td>
<td><strong>0.8/2.5</strong></td>
<td><strong>0.1/1.0</strong></td>
</tr>
<tr>
<td>VCTree†</td>
<td>VGG-16 [114]</td>
<td>BL</td>
<td>1.8/7.3</td>
<td>0.6/1.8</td>
<td>0.1/0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG</td>
<td><strong>3.5/10.2</strong></td>
<td><strong>0.7/2.4</strong></td>
<td><strong>0.3/0.9</strong></td>
</tr>
<tr>
<td></td>
<td>ResNeXt-101-FPN [82, 143]</td>
<td>BL</td>
<td>1.8/7.1</td>
<td>0.4/1.2</td>
<td>0.1/0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG</td>
<td><strong>4.3/10.6</strong></td>
<td><strong>0.8/2.5</strong></td>
<td><strong>0.3/1.5</strong></td>
</tr>
</tbody>
</table>

Table 4.3: Zero-Shot Recall on Visual Genome. Results are reported for three tasks across two detector backbones. Our approach is augmented to and contrasted against MOTIF [156] and VCTree [127]. † denotes our re-implementation of the methods.

and labels are available for the Predicate Classification task, the observed improvement here is the largest (34.6% higher AP on VGG). Analogously, the observed improvement on the Scene Graph Generation (SGDet) task is the lowest (7.3% higher AP on VGG) as any errors made by the pretrained detector are forwarded to the scene graph network. Note that no noticeable improvement is observed for the SGDet task when using the ResNeXt-101-FPN [82, 143] backbone. We believe this is a direct consequence of the backbone using feature pyramid networks (FPNs) [69] to extract features. As FPNs effectively capture global context using lateral connections, the detector provides much richer object representations. This makes the context aggregation in the scene graph network redundant, making refining segmentation masks harder.

Ablation. We conduct an ablation study over the various components in our model using VCTree. All models are trained with the ResNeXt-101-FPN [82, 143] backbone. The results on the SGCls task are shown in Table 4.4. ‘Base’ is defined as the vanilla VCTree model learned over the detector trained only on the Visual Genome dataset. To understand the effect our joint detector pre-training has on the overall performance, we define ‘Joint’ as the VCTree model learned over this jointly pre-trained detector. It can be seen that just the joint pre-training of the detector provides considerable improvements (5% on mR@20).

We incrementally add components of our proposed approach to the ‘Joint’ detector to better highlight their importance. ‘Joint + OG’ is defined as the model that uses the jointly trained detector and the object grounding mechanism described in
Table 4.4: **Ablation.** Mean Recall (mr) and Zero-shot Recall (zsr) are reported. VCTree [127] is the base architecture for all methods. Please refer Section 4.3 for model definitions.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>mR@20</th>
<th>mR@50</th>
<th>mR@100</th>
<th>zsr@20/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>8.1</td>
<td>9.9</td>
<td>10.6</td>
<td>0.3/1.2</td>
</tr>
<tr>
<td>Joint</td>
<td>8.5</td>
<td>10.5</td>
<td>11.1</td>
<td>0.4/1.5</td>
</tr>
<tr>
<td>Joint + OG</td>
<td>9.0</td>
<td>11.1</td>
<td>11.8</td>
<td>0.6/2.1</td>
</tr>
<tr>
<td>Joint + OG + EG&lt;sub&gt;avg&lt;/sub&gt;</td>
<td>9.1</td>
<td>11.4</td>
<td>12.2</td>
<td>0.8/2.4</td>
</tr>
<tr>
<td>Joint + OG + EG&lt;sub&gt;union&lt;/sub&gt;</td>
<td>9.1</td>
<td>11.3</td>
<td>12.2</td>
<td>0.7/2.4</td>
</tr>
<tr>
<td>Joint + OG + EG&lt;sub&gt;Gaussian&lt;/sub&gt;</td>
<td>9.3</td>
<td>11.5</td>
<td>12.2</td>
<td>0.7/2.3</td>
</tr>
<tr>
<td>Final Model</td>
<td><strong>9.4</strong></td>
<td><strong>11.6</strong></td>
<td><strong>12.3</strong></td>
<td><strong>0.8/2.5</strong></td>
</tr>
</tbody>
</table>

Section 4.1.4. Similarly, ‘Joint + OG + EG<sub>x</sub>’ describes the model that additionally uses our proposed relation grounding mechanism defined in Section 4.1.5. The subscript <sub>x</sub> in EG<sub>x</sub> refers to the type of masking mechanism used to combine the segmentation masks for a pair of objects. We experiment with averaging (<sub>avg</sub>), taking the logical or (<sub>union</sub>), and the proposed Gaussian masking (<sub>Gaussian</sub>). Finally, our complete model with the additional segmentation mask refinement (Section 4.1.6) is denoted as ‘Final Model’. From Table 4.4 it can be seen that using both object and relation grounding helps with performance, and using a Gaussian masking mechanism is superior to other alternatives. Additionally, fine-tuning the segmentation masks not only helps improve their quality but also provides better scene graph generation performance.

**Qualitative Results.** We qualitatively contrast the performance of the VCTree model [127] augmented with our proposed approach against its vanilla counterpart in Figure 4.3. The two images on the left show results from the relation retrieval task. Our approach (in green) predicts more granular and spatially informative relationships standing on and behind, as opposed to the baseline (in purple) which is heavily biased towards the more common and less informative relation on. The two images on the right highlight the ability of our approach to generalize in zero-shot scenarios. As the triplets of cat with sign are absent from the training dataset, the baseline approach (in purple) defaults to predicting incorrect relations of above and in-front-of. On the contrary, our approach accurately predicts the correct relation under.
4.4 Discussion

We present a novel model-agnostic framework for segmentation-grounded scene graph generation. Contrary to traditional scene graph generation frameworks that grounds object in a scene graph to bounding boxes, our proposed methodology allows for a more granular pixel-level grounding, obtained via a zero-shot transfer mechanism. Our proposed framework leverages these groundings to provide significant improvements across various scene graph prediction tasks, irrespective of the architecture it is augmented to. Finally, we highlight the benefits of simultaneously optimizing the tasks of scene graph and segmentation generation, which leads to improved performance on both. A limitation of this work, however, is the reliance on GloVe-based lingual similarity metric to derive segmentation masks for objects absent in the auxiliary dataset. This dependency poses a potential limitation to the method’s applicability, particularly in scenes featuring objects markedly dissimilar from those encompassed within the auxiliary dataset. The method proposed in this chapter was a source of inspiration for the creation of a new scene graph dataset with pixel-level grounding of objects [148]. Future works may explore the use of various kinds of training data such as tags, points etc to learn from a larger number of images and improve scene graph generation.
Chapter 5

Light Field Neural Rendering

In the previous two chapters, we discussed scene graph generation, as a form of object-centric semantic understanding of the scene. First at the level of bounding-boxes, from the perspective on maintaining structure, and second from the perspective of more granular ability to ground concepts into pixel-level masks. However, the 3D geometry of the scene was largely ignored in this process. Therefore, in this and the chapter that follows, we shift to looking at modeling geometric aspects of scene in the context of novel view synthesis.

Synthesizing a novel view given a sparse set of images is a long-standing challenge in computer vision and graphics [19, 110, 113]. Recent advances in 3D neural rendering for view synthesis, in particular, NeRF [85] and its successors [25, 30, 71, 89, 95, 142], have brought us tantalizingly close to the capability of creating photo-realistic images in complex environments. One reason for NeRF’s success is its implicit 5D scene representation which maps a 3D scene point and 2D viewing direction to opacity and color. In principle, such a representation could be perfectly suited to modeling view-dependent effects such as the non-Lambertian reflectance of specular and translucent surfaces. However, without regularization, this formulation permits degenerate solutions due to the inherent ambiguity between 3D surface and radiance, where an incorrect shape (opacity) can be coupled with a high-frequency radiance function to minimize the optimization objective [157]. In practice, NeRF avoids such degenerate solutions through its neural architecture design, where the viewing direction is introduced only in the last layers of the MLP, thereby limiting the expressivity of the radiance function, which effectively translates to a smooth BRDF prior [157]. Thus, NeRF manages to avoid degenerate solutions at the expense of fidelity in non-Lambertian effects (fig. 5.1 highlights this particular limitation of the NeRF model). Photo-realistic synthesis of non-Lambertian effects is one of the few remaining hurdles for neural rendering techniques.

In this chapter, we formulate view synthesis as rendering a sparsely observed light field. The 4D light field [63], which measures the radiance along rays in empty space, is often used for view synthesis [13, 48, 63]. Rendering a novel view from a densely sampled light field can be achieved with signal processing techniques (e.g., interpolation) and without any model of the 3D geometry, but no such
Figure 5.1: **Novel view synthesis.** On top is the target image to be rendered, from the *Lab* scene in the Shiny dataset [138]. Bottom row shows crops of novel views generated by our proposed model, NeX [138], and NeRF [85]. Unlike NeX and NeRF that fail to synthesize refractions on the test tube, our model almost perfectly reconstructs these complex view-dependent effects. We indicate the PSNR of the rendered images within parenthesis (higher is better).

A straightforward method exists with sparse light fields. From sparse images, rendering often utilizes additional 3D geometric constraints, such as predicted depth maps [48, 119], but performance is sensitive to accurate depth estimates which are difficult to obtain for non-Lambertian surfaces.

Motivated by these limitations, we introduce a novel method for rendering a sparse light field. Our neural rendering function operates in the style of image based rendering, where a target ray is synthesized using only observed rays from nearby views. In lieu of explicit 3D information, our transformer based rendering function is trained to fuse rays from nearby views exploiting an additional inductive bias in the form of a multi-view geometric constraint, namely the epipolar geometry. As shown in fig. 5.1, our model is able to faithfully reconstruct the sharp details and lighting effect in the most challenging scene in the Shiny dataset [138].

Our main contribution is the novel light field based neural view synthesis model, capable of photorealistic modeling of non-Lambertian effects (*e.g.*, specularities and translucency). To address the core challenge of sparsity of initial views, we leverage an inductive bias in the form of a multi-view geometric constraint, namely the epipolar geometry, and a transformer-based ray fusion. The resulting model
produces higher fidelity renderings for forward-facing as well as 360° captures, compared to state-of-the-art, achieving up to 5 dB improvement in the most challenging scenes. Further, as a byproduct of our design, we can easily obtain dense correspondences and depth without further modifications, as well as transparent visualization of the rendering process itself. Through ablations, we illustrate the importance of our individual design choices.

5.1 Approach

Figure 5.2: Model Overview. Given a target ray to render, we identify reference views and sample points along the epipolar lines corresponding to the target ray. Features of these epipolar points along with the light field coordinates of the target ray are inputs to the epipolar aggregation. This stage (blue), independently aggregates features along the epipolar lines for each reference view, producing reference view features. The reference view features along with the target ray are passed to the view aggregation stage (green), which combines the reference view features to predict the target ray color.

Our goal is to synthesize novel views of a scene given a collection of input images available during both training and inference. Our design is guided by two key ideas, 1) using the four-dimensional parametrization of the light-field as input enables capturing view-dependent effects with high fidelity, and 2) enforcing constraints from multiple view geometry allows for view synthesis with sparse input views.

These ideas enable faithfully recovering illumination effects as in classical lightfield methods [33, 63], but require only a sparse view-sampling of the scene,
as in geometry-based methods [23, 87] which traditionally struggle reproducing non-Lambertian effects. To implement them, we introduce an epipolar-geometric inductive bias in conjunction with a transformer-based architecture. Our model can render novel views from forward-facing photos as well as 360° scenes captured with cameras on a hemisphere.

In the following sections, we first introduce the light field representation, then an overview of the model, followed by a detailed description of the network architecture.

### 5.1.1 Light field parametrization

Light fields are functions on the space of oriented lines that associate a radiance value to a given ray. In free space, as the radiance along the ray remains constant, the space of rays has four degrees of freedom and can be parametrized by 4D vectors. We consider two distinct parametrizations of light field, the light slab [63] and the two-sphere [14].

**Light slab.** We adopt the light slab parametrization for forward-facing captures. A light slab consists of two parallel planes with their respective 2D coordinate systems \((s, t)\) and \((u, v)\). Rays are then represented as a 4D tuple \(r = (s, t, u, v)\) containing the coordinates of intersections with the two planes in their respective coordinate frames.

**Two-sphere.** For 360° scenes, we use the two-sphere parametrization [14] of the light field. Given a sphere bounding a scene, rays from the camera are represented using the colatitudes and longitudes at the two intersections with the sphere, \(r = (\theta_1, \phi_1, \theta_2, \phi_2)\).

Given a four-dimensional light field ray parametrization \(r\), we learn a neural rendering model \(f\) that maps the rays to radiance values.

To obtain the ray coordinates for a given pixel in homogeneous coordinates \(x \in \mathbb{R}P^2\), from an image taken using a camera with intrinsics \(C\) and pose (extrinsics) \([R \ t]\), we first obtain the ray as a line \(\ell\) in world coordinates parametrized by \(\delta\) as \(\ell(\delta) = -R^\top t + \delta R^\top C^{-1}x\), then solve for \(\delta\) to obtain the intersections \(r\) with either the two planes or the sphere. To render an image, we evaluate the model \(f(r)\) for the rays associated to each target pixel.

### 5.1.2 Model overview

Optimizing a neural rendering model \(f\) that directly maps 4D light field coordinates to color fails to generalize to novel views when trained with a sparse set of input
views (see section 5.2.3 for quantitative evaluation).

To address this challenge, we introduce a model that incorporates a geometric inductive bias in the form of the epipolar constraints.

Given a target camera, we identify a set of neighboring views to be used to enforce multiple-view consistency. During training, this set is constructed by randomly choosing $K$ views from a subset of $N$ closest views. During inference, the closest $K$ are chosen deterministically. We refer to the set of $K$ chosen views as reference views.

Now given a target pixel $x$ to be rendered, we obtain its ray parametrizations $\ell$ and $r$ as described in section 5.1.1, sample a sequence of $P$ points $p_i = \ell(\delta_i)$ along the ray, and project each point to each reference view as $x^j_i = C_j[R_j t_j]p_i$, where $C_j$, $[R_j t_j]$ are the reference view camera intrinsics and extrinsics, respectively, and $1 \leq j \leq K$.

The collection $x^j = \{x^j_i\}_{1 \leq i \leq P}$ consists of points along the epipolar line of the target ray in the $j^\text{th}$ reference view. We refer to $x^j_i$ as epipolar points. To each epipolar point, we associate its ray parametrization as described in section 5.1.1, yielding the collections $r^j = \{r^j_i\}_{1 \leq i \leq P}$.

**Epipolar feature aggregation.** The first stage of our model, represented by the function $f_1$, computes a feature representation per reference view by aggregating features associated to the epipolar points and target ray. We detail what those features are in the following sections. Conceptually, the first stage computes the set of features $\{z^j\}_{1 \leq j \leq K}$ where $z^j = f_1(r, r^j)$. This is loosely related to classical multiple view geometry, where we look for a correspondence to the target ray along the epipolar line. In our case, however, there is no visual representation of the target ray, so the model must learn to match the target ray coordinates with the available reference features, and the output is a feature vector representing the view $j$.

**View feature aggregation.** The second stage, represented by the function $f_2$, predicts the target ray color by aggregating features associated to each reference view, given the target ray representation. Conceptually, the color for pixel $x$ with associated ray $r$ is predicted as $f(r) = f_2(r, \{f_1(r, r^j)\})$. This stage learns to reason about occlusion and illumination effects to combine information from all views and produce the target ray color.

### 5.1.3 Network architectures

One possible approach would be to model $f_1$ and $f_2$ as multi-layer perceptrons (MLP). However, this impedes the model from exploiting readily available relational information that can be extracted from the epipolar points, leading to sub-
optimal performance (section 5.2.3 quantifies this). Since inputs to \( f_1 \) are a sequence of epipolar points, and inputs to \( f_2 \) are a set of reference view features, we propose to use transformers, which excel in sequence and set modeling, to model both epipolar and view feature aggregation.

**Epipolar feature transformer** (\( f_1 \))

This transformer, highlighted in blue in fig. 5.2, combines the features of points along the epipolar line based on the target rays.

The input is a sequence of \( P + 1 \) features, with \( P \) features from epipolar points and one from the target ray. The feature vector for the target ray is its own coordinates \( r \). The feature for an epipolar point \( x_i^j \) is a concatenation of 1) ray coordinates \( r_i^j \), 2) coordinates of \( p_i \), the 3D point along \( r \) projected to \( x_i^j \), 3) a learnable camera embedding \( k_j \), 4) visual features \( v_i^j \) at \( x_i^j \), obtained from a lightweight CNN, and 5) the color \( c_i^j \) at \( x_i^j \).
Assuming the target pixel $x$ matches to an epipolar point $x^j_i$, the corresponding point in the scene can be solved for and will have coordinates of $p_i$. Including it as an epipolar point feature also plays the role of positional encoding, since each point in the epipolar line correspond to some depth value along the query ray. This type of positional encoding is richer than the typical 1D encoding used in sequence modeling [132], and more appropriate for modeling a 3D scene, as demonstrated in section 5.2.3.

We further apply Fourier features [85, 125] positional encoding to facilitate learning of high-frequency functions. This operation is performed by $\gamma_r$ for ray coordinates and $\gamma_p$ for point coordinates, see section 5.2.1 for details. To summarize, each epipolar point $x^j_i$ is represented by a feature

$$y^j_i = [\gamma_r(r^j_i) \parallel \gamma_p(p_i) \parallel k_j \parallel v^j_i \parallel c^j_i],$$

(5.1)

where $\parallel$ denotes concatenation. The epipolar transformer for view $j$ will take as inputs $[\gamma_r(r), \{y^j_i\}_{1 \leq i \leq P}]$. A linear layer first projects the features to the same dimension, then a self-attention transformer is applied to the whole sequence.

We aggregate the $P$ outputs corresponding to the epipolar points ($\tilde{y}^j_i$) to obtain the reference view features. The aggregation is a weighted average, with the weights computed using an attention mechanism similar to the Graph Attention Networks (GAT) [134] as follows,

$$\alpha^j_i = \frac{\exp \left( W_1 \left[ \tilde{r} \parallel \tilde{y}^j_i \right] \right)}{\sum_k \exp \left( W_1 \left[ \tilde{r} \parallel \tilde{y}^j_k \right] \right)},$$

(5.2)

where $\tilde{r}$ are the output features of the target ray, and $W_1$ are learned weights. The first stage is completed by repeating $z^j = f_1(r, r^j) = \sum_{i=1}^P \alpha^j_i \tilde{y}^j_i$ for all views $1 \leq j \leq K$.

**View feature transformer ($f_2$)**

This transformer, highlighted in green in fig. 5.2, takes the target ray and the set of features for each reference view. The input sequence is now $[\gamma_r(r), \{z^j\}_{1 \leq j \leq K}]$, where $z^j$ are the reference view features computed by the first stage, and the output is a single feature vector for the target ray. We use the same self-attention transformer architecture as the epipolar feature aggregator. The transformer output sequence $[\tilde{r}, \{\tilde{z}^j\}_{1 \leq j \leq K}]$ is aggregated with a weighted average using the same idea as the previous section. We compute the weights $\beta^j$ with learnable weights $W_2$.

$$\beta^j = \frac{\exp \left( W_2 \left[ \tilde{r} \parallel \tilde{z}^j \right] \right)}{\sum_k \exp \left( W_2 \left[ \tilde{r} \parallel \tilde{z}^k \right] \right)},$$

(5.3)
### Table 5.1: Results for the real forward-facing (RFF) dataset [84].

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR [dB] ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
<th>Avg. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLFF [84]</td>
<td>24.41</td>
<td>0.863</td>
<td>0.211</td>
<td>0.0656</td>
</tr>
<tr>
<td>NeRF [85]</td>
<td>26.76</td>
<td>0.883</td>
<td>0.246</td>
<td>0.0562</td>
</tr>
<tr>
<td>IBRNet [136]</td>
<td>26.73</td>
<td>0.851</td>
<td>0.175</td>
<td>0.0523</td>
</tr>
<tr>
<td>NeX [138]</td>
<td>27.26</td>
<td>0.904</td>
<td>0.178</td>
<td>0.0473</td>
</tr>
<tr>
<td>Ours</td>
<td>28.26</td>
<td>0.920</td>
<td>0.062</td>
<td>0.0297</td>
</tr>
</tbody>
</table>

then the output of this stage is the target ray feature $\sum_{k=1}^{K} \beta_k z^k$, which is linearly projected and passed by a sigmoid to produce the pixel color prediction $c$.

#### 5.1.4 Loss

During training, we minimize the $L_2$ loss between the observed and predicted colors. We additionally include an auxiliary loss to encourage the attention weights for the epipolar points ($\alpha^j_i$) and reference views ($\beta^j$) to be interpretable, in the sense that high values of $\alpha^j_i$ suggest a valid match to the target ray, while low values of $\beta^j$ might indicate occlusion. This auxiliary loss also leads to more accurate renderings (see section 5.2.3). To compute it, we use the attention weights to combine reference pixel colors and make a second color prediction as

$$c_{aux} = \sum_j \beta^j \left( \sum_i \alpha^j_i c^j_i \right), \quad (5.4)$$

where $c^j_i$ is the color of the epipolar point $x^j_i$. The auxiliary loss is then defined as the $L_2$ loss between $c_{aux}$ and the ground truth. The effect of this loss is two-fold: 1) it incentivizes weights $\alpha^j_i$ to have lower entropy to avoid blurry predictions in the auxiliary branch, and 2) it encourages weights $\beta^j$ to be high for unoccluded views.

### 5.2 Experiments

We show quantitative and qualitative comparisons against state-of-the-art methods for novel view synthesis. We also perform an ablation study to analyze the effectiveness of the components introduced in our method.
5.2.1 Implementation details

Network architecture. We use similar transformer architectures as the ones recently introduced for vision related tasks [24]. Each block consists of a single-headed self-attention layer and an MLP with Gaussian error linear unit (GELU) activation [38]. A residual connection is applied at every block, followed by a LayerNorm (LN) [6]. Each transformer has 8 blocks and the internal feature size is 256. The visual features $v_j^i$ are produced by a single convolutional layer with $5 \times 5$ filters and 32 channels.

Positional encoding. Following prior work [85, 125], we use Fourier features to encode input coordinates to facilitate learning the high-frequency components required for accurate rendering. For the light slab parametrization and the 3D points $p_i$, we positionally-encode each ray coordinate [85] as $\gamma_r(w) = \gamma_p(w) = \{\sin(2^k w)\} \cup \{\cos(2^k w)\}$ for $0 \leq k \leq 4$. For the two-sphere parametrization, we found it beneficial to use a positional encoding based on evaluating the spherical harmonics at the points $(\theta_1, \phi_1)$ and $(\theta_2, \phi_2)$, see the appendix for details. The learnable camera embeddings $k_j$ are 256-dimensional.

Training/inference details. In each training step, we randomly choose a target image and sample a batch of random rays from it. The batch sizes are 4096 for the forward-facing datasets and 8192 for Blender. We train for 250,000 iterations with the Adam optimizer [55] and a linear learning rate decay schedule with 5000 warm-up steps. For inference, we sample contiguous blocks of rays to make a batch. Training on a Blender scene takes around 23 hours on a 32-core TPUv3 slice. Rendering an $800 \times 800$ image then takes around 9.2 seconds.

5.2.2 Results

We compare our method with LLFF [84], NeRF [85], IBRNet [136], NeX [138] and Mip-NeRF [8]. We compare against Mip-NeRF only on the Blender dataset because for forward-facing captures, as noted by Barron et al. [8, Appx D], Mip-NeRF performs on par with NeRF.

Metrics. To measure the performance of our model we use three widely adopted metrics: peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and the learned perceptual image patch similarity (LPIPS) [158]. Following [8], we additionally report the geometric mean of $10^{-\text{PSNR}/10}$, $\sqrt{1-\text{SSIM}}$ and LPIPS, which provides a summary of three metrics for easier comparison. We report the averages of each metric over all the scenes in each dataset. Please refer to the appendix for a scene-wise breakdown of the results.
### Table 5.2: Results for the Shiny dataset from NeX [138].

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
<th>Avg. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF [85]</td>
<td>25.60</td>
<td>0.851</td>
<td>0.259</td>
<td>0.0651</td>
</tr>
<tr>
<td>NeX [138]</td>
<td>26.45</td>
<td>0.890</td>
<td>0.165</td>
<td>0.0499</td>
</tr>
<tr>
<td>IBRNet† [136]</td>
<td>26.50</td>
<td>0.863</td>
<td>0.122</td>
<td>0.0468</td>
</tr>
<tr>
<td>Ours</td>
<td>27.34</td>
<td>0.907</td>
<td>0.045</td>
<td>0.0294</td>
</tr>
</tbody>
</table>

### Table 5.3: Results for the Blender dataset from NeRF [85].

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
<th>Avg. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF [85]</td>
<td>31.01</td>
<td>0.953</td>
<td>0.050</td>
<td>0.0194</td>
</tr>
<tr>
<td>IBRNet [136]</td>
<td>28.14</td>
<td>0.942</td>
<td>0.072</td>
<td>0.0299</td>
</tr>
<tr>
<td>Mip-NeRF [8]</td>
<td>33.09</td>
<td>0.961</td>
<td>0.043</td>
<td>0.0161</td>
</tr>
<tr>
<td>Ours</td>
<td>33.85</td>
<td>0.981</td>
<td>0.024</td>
<td>0.0110</td>
</tr>
</tbody>
</table>

### Real-forward-facing (RFF) dataset

The RFF dataset introduced by Mildenhall et al. [84] consists of 8 forward facing captures of real-world scenes using a smartphone. For our experiments, we use the same resolution and train/test splits as NeRF [85]. Table 5.1 reports the average metrics across all 8 scenes in the RFF dataset. We show qualitative comparisons on the Orchids scene in fig. 5.3. Compared to the baselines, our method retains sharper detailed textures and produces consistent shape boundaries on the leaves and petals.

### Shiny dataset

The RFF dataset mostly consists of diffuse scenes with little view-dependent effects. The Shiny dataset introduced in NeX [138] presents 8 scenes with challenging view-dependent effects, captured by forward-facing cameras. We use the same image resolution and splits as NeX.

We compare our model against NeX, IBRNet and NeRF on the Shiny dataset in table 5.2. We report the average scores across all scenes in the dataset. Our model consistently improves over the state-of-the-art in all metrics. We show qualitative analysis of rendering on a test view from the CD scene in fig. 5.3. Our model is able to reconstruct the interference patterns on the disk and reflections on the bottle with higher level of detail as compared to baselines.

†We fine tune the model at https://github.com/googleinterns/IBRNet on each
Table 5.4: Ablation study on the RFF dataset [84], with 25% the original resolution (504 × 378). Refer to section 5.2.3 for details.

Blender dataset

Our model is capable of rendering novel views of 360° scenes. To evaluate this case, we use the synthetic dataset introduced by Mildenhall et al. [85]. Each scene consists of 800 × 800 resolution images rendered from viewpoints randomly sampled on a hemisphere around the object.

Table 5.3 reports the average performance across all scenes in the Blender dataset. Our model improves over NeRF, IBRNet and Mip-NeRF on all metric and achieves new state-of-the-art results. On the materials scene, which contains reflections on metallic balls, we observe an improvement of around 4 dB on the PSNR metric when compared to Mip-NeRF. We present the full table along with qualitative comparisons in the appendix.

5.2.3 Ablation studies

To validate the effectiveness of different design decisions, we run the following ablation experiments.

Geometric inductive bias. We train a model, called ‘Vanilla-NLF’, that uses an MLP to predict the color of a ray given only its light field representation, without consideration of the scene geometry in form of epipolar constraints.

Transformers vs MLPs. We train variations of our model replacing the transformers with MLPs. In the first variant, we replace each of the epipolar and view transformers with MLPs (‘2-MLP’). The second variant replaces both transformers

scene in Shiny.
by a single MLP that takes as input all epipolar point features from all reference views along with the target ray and directly predicts the color (‘1-MLP’). We detail the architectures in the appendix. We run a sweep over the number of layers for these MLPs and report performance of the best model.

**Model Components.** To probe the efficacy of the different components, we train ablated models 1) without visual features $v_j^i$ (‘No CNN’), 2) without 3D coordinates $p_i$, 3) without the learnable camera embedding $k_j$ (‘No LCE’), 4) replacing the attention based aggregation with mean pooling, and 5) removing the auxiliary loss term.

Table 5.4 reports the ablation results. All models are trained on images from the RFF datasets downsampled to 25% of the original resolution (504 × 378). We use the average metrics across all the scenes for comparison.

### 5.2.4 Interpreting the model

The use of transformers and epipolar geometry in our model permits interpretation of the results via the attention weights. We demonstrate this by extracting correspondences and depth maps. Also, our use of a four-dimensional light field representation enables the construction of epipolar-plane images (EPI) [10], which

![Figure 5.4: Correspondence Distribution.](image)
Figure 5.5: **Disparity Map.** The per-point and per-view attention weights learned by our model can be applied to estimate a disparity map by aggregating the putative depths of each epipolar point on each reference view, for each target ray.

are interpretable reconstructions of the scene geometry.

**Dense correspondence.** We can extract potential correspondences between a target ray and a reference view \( j \) by finding the largest attention weights in \( \{\alpha^j_i\}_{1 \leq i \leq P} \). Figure 5.4 shows the weight distribution of putative correspondences over the epipolar line for four points of interest. For points (1) and (4) we observe unimodal distributions with peaks at the point of correspondence. For point (2) we notice some uncertainty, while for point (3), the distribution is multi-modal with peaks around each blade with the highest peak near the correct correspondence.

**Disparity map.** Since each epipolar point corresponds to the projection of the target ray at a certain depth (the putative depth), we can use the correspondence distributions to estimate a depth map for a target ray. We first extract all the epipolar point \( (\alpha^j_i) \) and reference view \( (\beta^j) \) attention weights as described in section 5.1.3, then compute a weighted average of putative depths, equivalent to applying eq. (5.4) with colors replaced by depths. Figure 5.5 shows an example of a the disparity map obtained for a test view of the Crest scene from the Shiny dataset [138].
Figure 5.6: **Epipolar-plane images (EPI).** Our model represents the 4D light field, so constructing EPIs is natural. Each EPI vertically stacks images along the blue line, while the camera moves parallel to the blue line. Different depths show as lines of different slopes in the EPI, while view-dependent effects show as curves.

**Epipolar-plane images (EPI).** For a 4D light slab representation, we construct the EPI by querying our model with two fixed and two variable coordinates, resulting in a 2D color image. Physically, this corresponds to moving the camera along a 1D trajectory, stacking the images of a line segment parallel to the trajectory. EPIs encode information about specularities and scene geometry, where diffuse points appear as lines and specular points appear as curves. We show the epipolar slices for the **CD** and **Flower** scene in fig. 5.6. The **Flower** scene is predominantly diffuse so we observe lines of varying slopes in the EPI, with slopes inversely proportional to the depth. For the **CD** scene, in addition to lines, we observe curves at regions corresponding to the interference pattern on the disk. This is due to the change in virtual apparent depth of the specular points with change in view point [84, 123].

### 5.3 Limitations

Since our method relies on implicitly finding the correspondences for a target pixel in nearby views, it is challenged by texture-less thin repeating structures. As shown in fig. 5.7, our model produces fuzzy details on the grill-like structure in the **Tools** scene from Shiny and on the wire-mesh on the microphone from Blender. Moreover, our method’s dependence on epipolar geometry necessitates precise camera
poses as input. Incorrect epipolar sampling inevitably results in inaccurate rendering of the target ray, underscoring the criticality of accurate camera pose estimation for the efficacy of our approach. The reliance on epipolar samples also limits their applicability to wide baselines where there is minimal overlap in scene content between reference images.

Transformers are computationally expensive, resulting in slow training and inference times. Our method is around 8 times slower than Mip-NeRF [8] on same hardware. Our model does, however, compare favorably in terms of speed against other transformer-based models. For example, NerFormer [96] takes around 180 s to render an 800 \times 800 image on a single V100 GPU, while our method takes 60 s to 70 s on the same hardware. We notice that our model suffers from overhead of 1) random memory access on device and 2) data transfer between host and device. We believe that it can be made more efficient with some engineering effort.

5.4 Discussion

We present a light field based neural rendering method for novel-view synthesis. Unlike prior volumetric rendering methods, our proposed model can naturally handle real-world illumination effects by learning the light field over a four-dimensional space. To address the dense sampling dependency of light field rendering, we introduced a two-stage framework that incorporates geometric inductive bias in the form of epipolar constraints. Our model leads to significant improvement over previous state-of-the-art models for view synthesis especially for scenes with challenging view-dependent effects. Finally, the design of our model allows for extracting dense correspondences, disparity maps and epipolar-plane images without any additional training. Due to its dependence on transformer-based archi-
tecture, our model is computationally expensive. Since our model requires optimiz-
ing per scene, the computational overhead makes training and inference slower than an MLP-based rendering network. In the next chapter, we introduce a method that can generalize to unseen scenes thus forgoing the need for per-scene training.
Chapter 6

Generalizable Patch-Based Neural Rendering

In the previous chapter we addressed a problem of training an accurate and geometrically consistent novel view synthesis model. However, each scene required a separate model and optimization. In this chapter, we consider the task of training a single novel view synthesis model that is capable of generating novel views of unseen scenes. There are a few notable efforts in this direction [17, 136, 153]. One key idea of these methods is to augment NeRF inputs with deep convolutional features, which include both local and global context. However, these methods still rely on scene-specific inputs such as 3D positions and directions, which are not reliable on unseen scenes. We also hypothesize that using feature extractors that have large receptive fields such as U-Net [102] or Feature Pyramid Networks [69] is harmful when generalizing to scenes visually far from the training distribution.

We propose a different approach that takes only local linear patch embeddings as input, eschewing deep convolutional networks. Moreover, our method does not require the ubiquitous volume rendering from NeRF; it produces the color of a target pixel directly from a set of reference view patches.

We are inspired by both classical and recent works. Classical computer vision tasks such as optical flow and image feature matching for 3D reconstruction were historically dominated by techniques operating on local patches [35, 75, 79, 111]. In fact, for some tasks the classical methods still outperform modern deep learning ones [106]. Another example is COLMAP [107, 108] which is a widely popular method for 3D reconstruction and typically used to generate camera poses (and sometimes depth maps) that are inputs to modern neural rendering.

Our decision to focus on local patches and to avoid convolutional features is supported by the recent success of the Vision Transformers (ViT) [24], which we employ. A second reason to use transformers [133] is that our input is effectively a set of patches, and self-attention is a powerful mechanism to learn from sets without making any assumption about the order of the elements. We show that transformers can effectively replace both the convolutional features and the volume rendering typically employed in the tasks we consider.

Our key contribution is to leverage the structure of the patch collection to build
Figure 6.1: **Motivation overview.** Our goal is to predict the color of a target ray, given only the reference images and camera poses. Consider the patches along each epipolar line, which correspond to samples of increasing depth along the target ray. If there are many matching patches at some depth, there is a high chance that the patch around the target ray also matches. In this example, the matching patches contain the flower, which is where the target ray hits. This motivates our three-stage architecture, that first exchanges information along views at each depth (yellow), then aggregates information along depths for each view (green), and finally aggregates information among reference views to predict the ray color (blue). The figure shows only 2 reference views with 15 sampled patches each, but in practice we use a larger number of views and samples.

Our contributions can be summarized as follows,

- We introduce a model that renders target rays in unseen scenes directly from a collection of patches sampled along epipolar lines of reference views.

- To exploit the structure of the patch collection, we design an architecture with stacked transformers operating over different subsets of the collection such that features are learned, combined, and aggregated in principled ways.

- To improve generalization to unseen scenes, we introduce canonicalized positional encodings of rays, depths, and camera poses such that all inputs to
the model are independent of the scene’s frame of reference.

- Our model outperforms previous baselines in multiple train and evaluation datasets, while using as little as 11% of training data in certain cases.

6.1 Approach

Given a set of scenes with a collection of images and their corresponding camera poses, we aim to learn a generic rendering model that is capable of rendering novel views of a scene without training on it. At the core of our model is a reference-frame-agnostic rendering network that relies only on local patches observed from nearby reference cameras. Figure 6.2 provides a visual overview. We present our approach in the following order: first we introduce light field representations; then we discuss the construction and embedding of reference patches; and finally we detail our transformer-based rendering network that maps a target light field and reference patches to radiance.

6.1.1 Light field representation

The light field characterizes the radiance through points in space. It can be described by a five-dimensional function on \( \mathbb{R}^3 \times S^2 \), mapping each direction through each point to its radiance. In free space, the radiance along a ray remains constant, thus allowing to parametrize the light field as a 4D function [63].

Depending on the camera configuration, different light field representations can be used. For example, for a scene with forward-facing camera configuration, the rays can be parametrized by their intersections with two planes perpendicular to the forward direction, a representation known as the light slab [63]. The entries of the 4D vector are the coordinates of the intersections on each plane’s 2D coordinate system. An alternative representation suitable for bounded scenes observed from all directions is known as the two-sphere [14], and represents rays by their two intersections with a sphere bounding the scene. The work in the previous chapter exploits the camera configuration information of the scene to decide the underlying parametrization. It uses light slab parametrization for forward-facing-scenes and two-sphere parametrization for 360° scenes.

In this chapter, the ray representations are used as positional encoding in the transformers. Since we wish to generalize to new scenes and therefore cannot make assumptions about the camera configurations, we use Plücker coordinates as the choice of parametrization. Given a ray through a point \( o \) (the ray origin) with direction \( v \), the Plücker coordinates can be obtained as \( r = (v, o \times v) \). The representation is six-dimensional, however, it has only four degrees of freedom.
Figure 6.2: **Model Overview.** Our model consists of three stages, with a different transformer per stage. First, patches along epipolar lines are extracted, linearly projected, and arranged in a grid of $K$ reference views by $M$ sampled depths. The first transformer takes a sequence of views and is repeated for each depth, returning another $K \times M$ grid. The second transformer takes a sequence of depths and is repeated for each view; it collapses features along the depth dimension, returning $K$ view features. The third transformer aggregates the $K$ view features. Attention weights extracted from the second and third transformers are used to blend colors over views and epipolar lines and make the final prediction. A canonicalized positional encoding of rays, depths and cameras is appended to the transformer inputs.

since it is defined up to a scale factor and the two vectors that compose it must be orthogonal. The Light Field Networks [117] use the same parametrization but in a different context.

### 6.1.2 Patch extraction

Given a target viewpoint, our method relies on eliciting “local” light field patches to produce the output images. To extract such patches, we first identify a set of reference images that serve as 2D slices of the plenoptic function observed from neighboring viewpoints. While our model is agnostic to the number of reference images, we use a subset of the available input images for patch extraction. Specifically, for a target camera we take a subset of the $N$ closest views. We randomly sub-sample $K$ views from this subset during training, and use the closest $K$ views
for inference.

Given the set of reference images $\mathcal{I} = \{I_1, I_2, ..., I_K\}$, the next step is to fragment them into patches. Dosovitskiy et al. [24] split the entire image into fixed-size non-overlapping patches. While this partition is useful for global reasoning (e.g. image classification), for view synthesis the relevant regions in the image can be isolated by exploiting the epipolar geometry between views.

For a given image in the reference set $I_k$, we compute the epipolar line corresponding to the target pixel. We sample $M$ points along this epipolar line such that their 3D re-projections on the target ray are spaced linearly in depth. We then extract square patches around each of the $M$ points, and this process is repeated for all reference images. The resulting reference patch set is indexed by view and depth: $\mathcal{P} = \{P_{mk}^k \mid 1 \leq k \leq K, 1 \leq m \leq M\}.$

### 6.1.3 Patch embedding and positional encoding

The inputs to the transformers are patch embeddings which we generate by linearly projecting flattened input patches. The patch features for the $m$-th sample along the epipolar line on view $k$ is denoted $p_{mk}^k$.

Since transformers are agnostic to the position of each element in the input sequence, typically a positional encoding is added to the features to represent the spatial relationship between elements. Unlike prior works [24], since the location and source of patches do not remain the same across batches, we cannot include a learnable embedding into the sequence. Instead, we extract the geometric information associated with each patch and append them to the flattened patch feature vectors.

We use three forms of positional encoding:

1. To retain the reference patch position in space, we use the light field encoding of the rays emanating from the reference camera as described in section 6.1.1. We represent the $m$-th ray along the epipolar line of view $k$ by $r_{mk}$.

2. To retain the position of the patch in the sequence of patches along the epipolar line, we encode the distance along the target ray corresponding to the patch center using a sinusoidal positional encoding that follows NeRF [85]. The encoded distance for the $m$-th sample is represented by $d_m$.

3. To retain geometry between target and reference cameras, we also append the relative camera pose as a flattened rotation matrix and a 3D translation, which is shared among all patches associated to the same camera and denoted by $c_k$ for camera $k$. 
6.1.4 Canonicalized ray representation

Structure-from-Motion (SfM) methods that are used to estimate camera extrinsics can only reconstruct scenes up to an arbitrary similarity transformation – rotation, translation and scaling. Prior works [17, 136, 153] use such estimations to compute scene specific coordinates such as view directions and 3D coordinates of points.

We hypothesize that for best generalization to unseen scenes, the inputs to the model should be invariant to similarity transformations. This means that model should produce the same result upon a change of reference frame or rescaling of the input camera poses. IBRNet [136] takes a step towards this idea by using the difference between reference and target direction vectors instead of absolute directions, but the difference is still a 3D vector that is not independent of the frame of reference, and so are the 3D positions of points along the target ray.

The positional encoding of relative camera poses and distance values, as described in section 6.1.3, are made invariant to similarities by simply scaling the camera positions by the maximum depth of the scene output by SfM.

The encoding of rays in the light field, however, need to be canonicalized. Our key idea is to define a local frame centered on each ray (not camera). For a target pixel \( x \in \mathbb{RP}^2 \) in the target camera with extrinsics \([R \mid t]\) and intrinsics \(C\), we first obtain the corresponding ray direction \( v = R^T C^{-1} x \). We use \( v \) and the camera \( y \) axis to determine the local frame. Specifically, we use the Gram-Schmidt orthonormalization process. Let \( v' = v / \|v\| \) and \( y' = y - (y \cdot v') v' \). The canonicalizing transformation is then

\[
R_c = \begin{bmatrix}
y' \\
\|y'\| \times v'
y' \\
\|y'\|
v'
\end{bmatrix}
\]

\[
T = \begin{bmatrix} R_c^T & -R_c^T t \end{bmatrix}
\]

where \( T \in SE(3) \). We apply \( T \) to every camera pose, which results in the target ray having origin \((0, 0, 0)\) and direction \((0, 0, 1)\), and all other ray representations computed from the canonicalized camera poses will be invariant to similarities. We show the benefit of such canonicalization in Section 6.3.

6.1.5 Rendering network

Given the patch embeddings and positional encodings of a target ray, as described in sections 6.1.3 and 6.1.4, our rendering network predicts the ray color.

We argue that predicting the target ray color is deeply related to finding correspondences to the target ray in the reference images. Take, for example, the method in the previous chapter, LFNR [121]. Its first stage aggregates features along each
epipolar line, which is essentially finding correspondences to the target ray. Since LFNR overfits to a single scene, the model can learn the structure of the scene and use it to estimate correspondences based only on ray coordinates.

However, the LFNR [121] approach cannot generalize to novel scenes, since, given just an epipolar line, it is impossible to know which point corresponds to a target ray without knowing the structure of the scene.

The main contribution of this chapter is to provide visual features for a similar epipolar transformer, such that the correspondence is solved visually (see fig. 6.1 for illustration), which is advantageous because the visual features can be extracted from novel scenes in a single forward step starting from small local patches. Crucially, such features cannot come from a single epipolar line. It is the combination of visual features from different epipolar lines cast by the same target ray that allows correspondences to be established. To learn this combination, we propose to use a transformer.

Thus, our model consists of three transformers. The first, which we call “Visual Feature Transformer”, learns visual features by combining information from patches along different reference views. The second and third are similar to the ones in LFNR [121], with the major differences that the positional encodings of rays, depths and cameras are canonicalized as described in section 6.1.4 and that the final color is predicted by directly blending pixel colors from reference views, instead of using learned features; both changes greatly improve the generalization performance.

Each transformer follows the ViT [24] architecture, which uses residual connections to interleave layer normalization (LN), self-attention (SA), and multi-layer perceptron (MLP). Each layer consists of LN → SA → LN → MLP.

**Visual Feature Transformer.**

This stage exchanges visual information between potentially corresponding patches on different reference images, leading to visual features with multi-view awareness. The input to this stage is the set of patch linear embeddings and positional encoding vectors $p^m_k$, $r^m_k$, $d^m$, $c_k$, indexed by the view $k$ and the $m$-th sampled depth, as described in section 6.1.3. We first define the feature concatenation at layer zero (the input) as

$$f^{k,m}_0 = [p^m_k \parallel r^m_k \parallel d^m \parallel c_k]. \quad (6.3)$$

This stage is repeated for each depth sample, therefore it operates on sequences of $K$ views. Formally, it repeats

$$f^{m}_1 = T_1 \left( \left\{ f^{k,m}_0 \mid 1 \leq k \leq K \right\} \right) \quad (6.4)$$
for \( 1 \leq m \leq M \), where \( T_1 \) is a transformer written as a set to set map. This stage takes a \((K, M, C_0)\) tensor of \( C_0 \)-dimensional features of \( K \) views sampled at \( M \) depths, and returns a \((K, M, C_1)\) tensor of \( C_1 \)-dimensional features.

**Epipolar Aggregator Transformer.**

This stage aggregates information along each epipolar line, resulting in per reference view features. The input to this stage is the set \( f_1 = \{ f_1^m \mid 1 \leq m \leq M \} \), concatenated with positional encodings. We refer to the features corresponding to view \( k \) in the set \( f_1^m \) as \( f_{k,m}^1 \). The transformer is repeated for each view, therefore operating along the sequence of \( M \) epipolar line samples. Formally, we first compute

\[
\begin{align*}
  f_{2,k}^k &= T_2 \left( \{ r_0 \} \bigcup \left\{ \left[ f_{1,k,m}^k \parallel r_0^m \parallel d^m \parallel c_k \right] \mid 1 \leq m \leq M \right\} \right), \\
  f_{2,k}^k &= \sum_{m=1}^{M} \alpha_{k,m}^m f_{2,k}^m,
\end{align*}
\]

for \( 1 \leq k \leq K \), where \( r_0 \) is a special token to represent the target ray. We then apply a learned weighted sum along the \( M \) epipolar line samples as follows,

\[
\begin{align*}
  \alpha_{k,m}^m &= \frac{\exp \left( W_1 \left[ f_{2,k}^k \mid f_{2,m}^m \right] \right)}{\sum_{m'=1}^{M} \exp \left( W_1 \left[ f_{2,k}^k \mid f_{2,m'}^m \right] \right)}, \\
  f_{2,k}^k &= \sum_{m=1}^{M} \alpha_{k,m}^m f_{2,k}^m,
\end{align*}
\]

for \( 1 \leq k \leq K \), resulting in a feature vector per view \( k \), where \( W_1 \) are learnable weights and \( f_{2,k}^0 \) is the output corresponding to the target ray token.

Dimension-wise, this stage takes a \((K, M, C_1)\) tensor and returns a \((K, C_2)\) tensor of \( C_2 \)-dimensional features per reference view.

**Reference View Aggregator Transformer.**

This final transformer aggregates the features over reference views and predicts the color of the target ray. Its input is the set of per reference view features \( f_{2'} = \{ f_{2'}^k \mid 1 \leq k \leq K \} \), concatenated with the camera relative positional encoding. Formally, we compute

\[
\begin{align*}
  f_3 &= T_3 \left( \{ r_0 \} \bigcup \left\{ \left[ f_{2'}^k \parallel c_k \right] \mid 1 \leq k \leq K \right\} \right).
\end{align*}
\]
Similarly to the previous stage, we compute the blending weights

\[ \beta_k = \frac{\exp \left( W_2 \left[ f_3^0 \left\| f_3^k \right. \right] \right)}{\sum_{k'=1}^{K} \exp \left( W_2 \left[ f_3^0 \left\| f_3^{k'} \right. \right] \right)}, \tag{6.9} \]

which are used in conjunction with the weights from the previous stage to estimate the color of the target ray by blending colors along each epipolar line sample at each reference view,

\[ c = \sum_{k=1}^{K} \beta_k \left( \sum_{m=1}^{M} \alpha_k^m c_k^m \right), \tag{6.10} \]

where \( c_k^m \) is the pixel color at the \( m \)-th sample along the epipolar line of view \( k \). Our approach here differs from the last stage of LFNR, which does the aggregation on feature space using only the weights \( \beta_k \), and linearly projects the resulting feature to predict the color. We argue that using the input pixel values from reference views instead helps generalization, which we confirm experimentally. This is possible by using the two sets of attention weights \( \alpha_k^m \) (eq. (6.6)) and \( \beta_k \) (eq. (6.9)), which allows blending colors from all epipolar line samples and all reference views.

### 6.2 Experiments

#### 6.2.1 Implementation Details

Each of the three transformers in our model consist of 8 blocks each with a feature dimension of 256. We select reference views using \( K = 10 \) and \( N = 20 \) (see section 6.1.2). We use a batch size of 4096 rays and train for 250k iterations with a Adam optimizer and initial learning rate of \( 3 \cdot 10^{-4} \). We use a linear learning rate warm-up for 5k iterations and cosine decay afterwards. Training our model takes \( \sim 24 \) hours on 32 TPUs. We report the average PSNR (peak signal-to-noise ratio), SSIM (structural similarity index measure) and LPIPS (learned perceptual image patch similarity) for all our experiments.

#### 6.2.2 Results

There is no standard training and evaluation procedure for generalizable neural rendering. IBRNet [136] trains on the LLFF dataset [84], renderings of Google scanned objects [98], Spaces dataset [27], RealEstate10K dataset [161] and on their
Figure 6.3: **Qualitative results on RFF (setting 1).** We show our method and the baseline on the *T-Rex* and *Fern* scenes from the real forward-facing dataset. Compared with IBRNet [136], our method produces sharper details and less blurring at boundaries. For example, the top row in the *Fern* scene shows that the baseline methods either fail to reconstruct the leaves or produce inconsistent shapes. Our method is able to retain the shape boundaries accurately along with majority of the texture details.

own scenes. They evaluate on the real forward-facing (RFF) dataset [85], which comprises held-out LLFF scenes, Blender (consisting of 360° scenes) [85], and Diffuse Synthetic 360° [115]. Contrastingly, MVSNeRF [18] trains on DTU [43] and tests on held out DTU scenes, real forward-facing dataset (RFF) [85], and Blender [85]. Various other works [45, 73, 131] have explored different experimental setups. To fairly evaluate against prior works, we use two experimental settings.

**Setting 1**

In the first setting, we train on a strict subset of the IBRNet training set, comprised of 37 LLFF scenes and 131 IBRNet collected scenes (amounting to 11% of the training set used by IBRNet). We then evaluate on the real forward-facing, Shiny [138] and Blender datasets. On Shiny, we compute the results for IBR-
Chapter 6. Generalizable Patch-Based Neural Rendering

Table 6.1: Results for setting 1. Our model outperforms the baselines even when training with strictly less data. IBRNet uses three datasets that are not part of our training set, while GeoNeRF uses one extra dataset and also leverages input depth maps during training. IBRNet* was trained using the same training set as our method; in this fair comparison, our advantage in accuracy widens.

Net using their publicly available pretrained weights. Table 6.1 reports quantitative while figs. 6.3 and 6.4 show qualitative results. IBRNet and GeoNeRF use a larger training set than ours, and GeoNeRF uses depth maps during training, but our method shows the best performance in most metrics regardless. Additionally, IBRNet is trained on 360° scenes whereas our method is trained only on forward-facing scenes. Nonetheless, our model achieves superior performance on Blender as compared to IBRNet.

Setting 2

Here, we train our model on DTU, following the MVSNeRF [17] procedure, and evaluate on the held-out DTU scenes and the Blender dataset. For training on DTU, we follow the same split as PixelNeRF [153] and MVSNeRF. We partition the dataset into 88 scenes for training and 16 scenes for testing, each containing images of resolution 512 × 640. Table 6.2 shows quantitative results. MVSNeRF is trained with 3 reference views while our method performs best with 10. We evaluated MVSNeRF with 10 views, which did not improve their performance; table 6.2, thus, compares the best number of views for each model. Our model consistently outperforms across all three metrics.

Ablation.

To investigate the effectiveness of our contributions, we perform various ablations experiments. We train the model on 504 × 378 resolution images of LLFF and

<table>
<thead>
<tr>
<th>Method</th>
<th>Real Forward-Facing</th>
<th>Shiny-6</th>
<th>Blender</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
<td>LPIPS</td>
</tr>
<tr>
<td>LLFF [84]</td>
<td>24.13</td>
<td>0.798</td>
<td>0.212</td>
</tr>
<tr>
<td>IBRNet [136]</td>
<td>25.13</td>
<td>0.817</td>
<td>0.205</td>
</tr>
<tr>
<td>GeoNeRF [45]</td>
<td>25.44</td>
<td>0.839</td>
<td>0.180</td>
</tr>
<tr>
<td>IBRNet*</td>
<td>24.33</td>
<td>0.801</td>
<td>0.213</td>
</tr>
<tr>
<td>Ours</td>
<td>25.72</td>
<td>0.880</td>
<td>0.175</td>
</tr>
</tbody>
</table>
Figure 6.4: **Qualitative results on Shiny [138] (setting 1).** While still consisting of forward facing scenes, Shiny scenes have scale and view density that differ from the usual in setting 1, which makes it more challenging than LLFF. IBRNet [138] produces noticeable artifacts that are not present in our method’s renderings.

IBRNet scenes and test on the real forward-facing dataset at the same resolution. We start with a “base model” that does not use the visual feature transformer or coordinate canonicalization. We then incrementally add components of our proposed approach. Table 6.3 reports the ablation results. We observe that the “base” model generalized poorly to unseen scenes. Incorporating the visual feature transformer improves the performance significantly. For the canonicalization ablation, we split the component into two, (1) ray canonicalization, where the light field ray representation is computed independent of the frame of reference, and (2) coordinate canonicalization where the 3D samples along the target ray are canonicalized. We observe that both forms of canonicalization help improve accuracy.

6.3 Discussion

This paper introduced a method to generate novel views from unseen scenes that predicts the color of an arbitrary ray directly from a collection of small local patches sampled from reference views according to epipolar constraints. Our model departs from the common combination of using deep visual features and NeRF-like
Table 6.2: **Results for setting 2.** All models are trained on DTU and evaluated on either the DTU held-out set or Blender. Our approach outperforms the baselines.

<table>
<thead>
<tr>
<th>Method</th>
<th>DTU</th>
<th>Blender</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
</tr>
<tr>
<td>PixelNeRF</td>
<td>19.31</td>
<td>0.789</td>
</tr>
<tr>
<td>IBRNet</td>
<td>26.04</td>
<td>0.917</td>
</tr>
<tr>
<td>MVSNeRF</td>
<td>26.63</td>
<td>0.931</td>
</tr>
<tr>
<td>Ours</td>
<td>28.50</td>
<td>0.932</td>
</tr>
</tbody>
</table>

Table 6.3: **Ablations.** Ablation study for model trained on LLFF and IBRNet scenes and tested on RFF with a resolution of $504 \times 378$. Results show that our main contributions – the visual feature transformer and the canonicalized positional encoding – lead to superior generalization performance.

<table>
<thead>
<tr>
<th>Visual Transformer</th>
<th>Ray Canonicalization</th>
<th>Coordinate Canonicalization</th>
<th>PSNR</th>
<th>SSIM</th>
<th>LPIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>22.62</td>
<td>0.763</td>
<td>0.313</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>25.42</td>
<td>0.879</td>
<td>0.154</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>25.86</td>
<td>0.885</td>
<td>0.142</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>26.42</td>
<td>0.896</td>
<td>0.129</td>
</tr>
</tbody>
</table>

volume rendering for this task. We introduced a three-stage transformer architecture, coupled with canonicalized positional encodings, which operates on local patches – all these properties aid in generalizing to unseen scenes. This is demonstrated by our outperforming of the current state-of-the-art while using only 11% of the amount of training data. One limitation of our model is that it requires a large number of views (and epipolar projections) to produce meaningful features as it operates on small local patches. In the comparison against MVSNeRF [17] in table 6.2, while our method is more accurate by significant margins, it also requires 10 reference views while MVSNeRF only uses 3. The dependence on a large number of reference views (and projection) slows down the inference, requiring around 60 seconds to render an $800 \times 800$ image. One possible method to address this is to use more computationally efficient alternatives to transformers or distilling the network into a smaller network.
Chapter 7

Associating Object and their Effects in Unconstrained Monocular Video

Figure 7.1: Layer decomposition under strong camera parallax. Given an input video with unconstrained camera motion and approximate object masks and depth (left), our method estimates a layered representation composed of a background layer and object layers containing the subjects of interest and their associated effects (e.g. shadows). Results of combining object layers and background shown on the right.

The previous chapter on scene graph generation aimed to understand the semantics of the scene. Through novel view synthesis we aimed to understand the underlying geometry of the scene. These two tasks explored semantics and geometry in isolation. In an attempt to jointly reason about both semantics and geometry, in this chapter, we explore the problem of layered scene representation.

Decomposing a video into meaningful layers (as shown in fig. 7.1) is a long-standing and complex problem [135] that has seen a recent surge in progress with the application of deep neural networks [49, 77, 78, 152]. A challenging variant of layer decomposition is the omnimatte [78] task, which aims to separate an input video into a background and multiple foreground layers, each containing an object of interest along with its correlated effects such as shadows and reflections, thus
enabling video editing applications such as object removal, background replacement and retiming. The original work on omnimatte [77, 78] introduced a self-supervised approach for decomposing a video by assuming the background can be unwrapped onto a single static 2D background canvas using homography warping. Following work [49, 152] relaxed the homography restriction, but maintained the necessity of unwrapping onto a 2D canvas.

This 2D modelling, however, limits the applicability of these methods to camera motions that have limited or zero parallax; intuitively, only panning camera motions are allowed. If the camera’s center of projection moves a significant distance over the video, 2D methods are unable to learn an accurate background and are forced to place the background detail in a foreground layer (Figure 7.2). Since camera parallax is quite common, 2D methods are severely restricted in practice.

To handle camera parallax, we exploit another recent line of work on estimating 3D camera position and depth maps from casually-captured video with moving objects [58, 80, 159, 160]. Instead of a 2D layer decomposition, we propose to learn a 3D background model that varies per-frame and includes inpainted RGB and depth in the regions occluded by the foreground. Allowing the background to vary per-frame, rather than assuming a global, static canvas, creates an ill-posed
Figure 7.3: Model Overview. Given a frame from the video along with an input mask, disparity and known camera pose, we first extract features by stacking the input and passing them through a 2D UNet architecture. These features are input to the background and foreground networks. The background network predicts the inpainted background RGB and disparity. The foreground networks predict an RGB-A image for each foreground layer. For scenes with multiple layers, the RGB-A image for each layer is predicted using independent networks.

decomposition problem, as variation in each input pixel could be explained by a foreground object, background, or both. To resolve this ambiguity, we enforce that the background varies slowly over time with 3D multi-view constraints using dense depth and camera pose estimates [160]. As shown in fig. 7.2, our model can successfully separate the layers obtaining a clean background RGB whilst capturing only the person and their shadow in the foreground layer.

Contributions. Our main contribution is a novel video decomposition method capable of accurately separating an input video with complex object and camera motion (e.g., parallax) into a background and object layers. To address the ill-posed decomposition problem, we design regularization terms based on multi-view consistency and smoothness priors. The resulting model produces clean decompositions of real-world videos where previous methods fail.
7.1 Method

Given an input video, with known camera poses and initial depth estimates, the goal is to predict a per-frame layer decomposition consisting of a background layer and \( N \) foreground layers encompassing objects of interest along with their associated effects. To this end, we propose a new method that predicts a layered separation of an input video without relying on a 2D background canvas, thereby allowing for greater flexibility of camera motion. We show an overview of our model in Figure 7.3. We optimize our model to output layers that reconstruct a single target video subject to a projection consistency loss. For each frame in the video, we estimate a layered representation consisting of background and foreground object layers where the contents of the object layers are conditioned on their corresponding input mask. Post-optimization, depth for foreground objects can be extracted from the input depth to produce a Layered Depth Image (LDI). Unlike prior works that rely on deep priors for inpainting the depth and color of occluded regions in the background, our method relies on multi-view consistency losses to steer the inpainting process.

7.1.1 Layer Decomposition Networks

We represent a frame at time \( a \) in the video as \( F_a = (I_a, D_a, \{M^i_a\}^N_{i=1}) \), where \( I_a \) is the RGB frame, \( D_a \) is the disparity, and \( \{M^i_a\}^N_{i=1} \) are the input masks for \( N \) objects in the video. We denote the camera extrinsics for the frame \( F_a \) as \([R_a \ t_a]\) where \( R_a \) is the rotation matrix and \( t_a \) is the translation vector. The camera intrinsics are denoted as \( K_a = [f_x, f_y, c_x, c_y] \) where \( f_x \) and \( f_y \) are the focal lengths and \( c_x, c_y \) is the principal point.

For each frame \( F_a \) in the input video, our method first extracts features by passing it to a feature extractor network modeled using a 2D UNet [103] architecture. The input to the feature extractor is constructed by concatenating input masks with the masked frame RGB and disparity along the channel dimension. The masked RGB and disparity are obtained as

\[
\tilde{Y}_a = [Y_a \odot M^b_{a} \parallel Y_a \odot M^1_{a} \parallel \ldots \parallel Y_a \odot M^N_{a}]
\]  

where \( Y_a \) is the concatenated RGB-D, \( M^b_{a} = 1 - \bigcup^N_{i=1} M^i_{a} \) is an approximate mask for the background and \( \parallel \) is the concatenation operation. These features act as input to both the background and foreground object networks.

The background network (shown in green in fig. 7.3) consists of a 2D UNet and a convolutional neural network (CNN) that predicts the inpainted background color \( (C^b_{a}) \) and depth \( (D^b_{a}) \) respectively. The foreground networks are separate...
CNNs that predict RGB ($C^i_a$) and alpha ($A^i_a$) images for each foreground layer independently. These predicted outputs, along with the input disparity, are combined to form an LDI.

The feature extractor and background RGB UNets consist of five down-sampling and up-sampling layers with three residual blocks at every resolution. The background disparity and foreground layer prediction networks are comprised of single convolutional layers.

### 7.1.2 Differentiable Rendering

In addition to a reconstruction loss, our method relies on a projection consistency term (described in Section 7.1.3) that requires differentiable projections and renderings of the predicted background layer to different time steps. We achieve this using a differentiable rasterization framework [22].

To project the predicted background layer from a source time step $a$ to target time step $b$, we first construct a mesh $M_a = (V_a, F_a)$, where $V_a$ and $F_a$ are the mesh vertices and faces. Each vertex in the mesh corresponds to an image pixel. To obtain the vertex coordinate for a pixel, we unproject its homogeneous coordinates $x = [i, j, 1] \in \mathbb{R}P^2$ using the predicted disparity as

$$V^a_x = d_{ij} \left( \frac{(i - c_x)}{f_x}, \frac{(j - c_y)}{f_y}, 1 \right)$$

(7.2)

where $d_{ij}$ is the depth at pixel $(i, j)$. The attributes for the vertices are composed of the predicted RGB, an alpha masking consisting of all ones, and the pixel coordinates. The faces for the mesh are constructed by adding an edge between vertices of neighboring pixels (bottom right in fig. 7.3). The view-projection matrix to render the predicted background from a target camera is computed as

$$H_{a \rightarrow b} = P_a \left[ \begin{array}{cc} R^T_a R_b & R^T_a (t_b - t_a) \\ 0 & 1 \end{array} \right]$$

(7.3)

where $P_a$ is the perspective matrix that maps camera-space to clip space. The view-projection matrix along with the estimated mesh is passed through a differentiable rasterizer that returns an RGB-A ($C_{a \rightarrow b}^{bg}, A_{a \rightarrow b}^{bg}$) image corresponding to the projection of the background to the target time step.

### 7.1.3 Losses

The total loss to train our model is:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{recon} + \lambda_2 \mathcal{L}_{proj} + \lambda_3 \mathcal{L}_{mask} + \lambda_4 \mathcal{L}_{disp} + \lambda_5 \mathcal{L}_{reg}$$

(7.4)

We describe each loss term in the following sections.
Figure 7.4: **Qualitative layer decomposition results on DAVIS.** Top: examples of single-layer decomposition on *Snowboard* and *Scooter-gray*. For each example we show the predicted background layer (row 1), predicted object layer RGBA (row 2), and alpha channel visualization (row 3). Bottom: example of multi-layer decomposition on the *Longboard* scene. On *Longboard*, we only compare against Omnimate since Neural-Atlas does not support multi-layer decomposition. Our method produces stronger background inpainting results and more accurately disentangles the foreground objects and their effects from the background.

**Reconstruction Loss**

For a source frame $F_a$ we reconstruct the image from the predicted background RGB and foreground RGBA layers using back-to-front compositing [94] (bottom
left in fig. 7.3). During training, we minimize the $L_2$ loss between the original RGB image and the reconstruction.

$$L_{\text{recon}} = \| (I_a - f (C^{bg}_a, \{C_i^a, A_i^a\}_{i=1}^N) \|_2$$ (7.5)

where $f$ is the back-to-front over-compositing function.

A large number of layer decompositions can reproduce the original frame and minimize the reconstruction loss. To guide the foreground layers to be semantically meaningful, we incorporate an $L_1$ loss between the predicted foreground alphas and the input masks.

$$L_{\text{mask}} = \frac{1}{N} \sum_{i=1}^N \| M_i^a - A_i^a \|_1$$ (7.6)

Since we want the foreground alphas to be able to capture effects beyond the input mask, the weight for this loss is decayed as described in section 7.1.4.

**Projection Consistency Loss**

The reconstruction loss on the over-composited image does not provide any gradients for the occluded region in the background. Prior works overcome this challenge by learning a background model that maps from a static 2D canvas. However, since our model predicts the background for each frame independently, we require additional regularization to prevent random artifacts or foreground details from appearing in occluded regions. We thus introduce a projection consistency loss on the background prediction. Each batch in our input is composed of three frames from the video. We denote these frames as $F_a$, $F_b$ and $F_c$. The frames $F_a$ and $F_b$ act as source images for the target frame $F_c$. The criteria used for selecting the frames is described in Section 7.1.4.

To estimate the projection consistency loss, we project the predicted background RGBs ($C^{bg}_a$ and $C^{bg}_b$) for the two source frames, $F_a$ and $F_b$, using the method described in section 7.1.2. We use the predicted disparities ($D^{bg}_a$ and $D^{bg}_b$) to construct the mesh and set the alpha values to one for all pixels. We then apply an $L_2$ loss between the projected source backgrounds and the predicted target background.

$$L_{\text{proj}}^{\text{rgb}} = \frac{1}{2} \sum_{k \in \{a, b\}} \| A^{bg}_{k \rightarrow c} \odot (C^{bg}_{k \rightarrow c} - C^{bg}_c) \|_2$$ (7.7)

where $\odot$ denotes element wise multiplication. We mask the loss using the projected alphas ($A^{bg}_{k \rightarrow c}$) to prevent noisy gradients from the boundaries of projection.
Similarly, to encourage consistent inpainting of depth we apply a similar projection loss,

\[ L_{\text{proj}}^{\text{coord}} = \frac{1}{2} \sum_{k \in \{a, b\}} \| A_{k \rightarrow c}^{bg} \odot (A_{k \rightarrow c}^{bg} - A_{c}^{bg}) \|_2 \quad (7.8) \]

where \( X^c \) are the world coordinates for pixels in the frame \( F_c \) obtained using the predicted background disparity \( D_{bg} \). \( A_{k \rightarrow c}^{bg} \) are the projections of the world coordinate from frame \( F_k \) to \( F_c \) and \( A_{k \rightarrow c}^{bg} \) are the projected alphas. The projection loss \( L_{\text{proj}} \) is the sum of \( L_{\text{rgb}}^{\text{proj}} \) and \( L_{\text{coord}}^{\text{proj}} \).

**Alpha Regularization**

While the projection loss encourages the background predictions to be consistent across frames, it can also lead the model toward undesirable or degenerate solutions. For example, a model that predicts a constant color at every pixel for every frame will yield a projection loss of zero and essentially force the background to be captured in foreground layers to minimize the reconstruction loss. To prevent such solutions, we regularize the foreground alphas with an \( L_1 \) and approximate-\( L_0 \) term as in [78].

\[ L_{\text{sp}} = \frac{1}{N} \sum_{i=1}^{N} \| A_i^a \|_1 + \gamma \| 2 \cdot \sigma(5 \cdot A_i^a) - 1 \|_1 \quad (7.9) \]

where \( \sigma \) is the sigmoid function and \( \gamma \) control the relative weight between the two loss terms.

Additionally, we regularize the foreground alphas with a smoothness term to discourage them from capturing sharp details in the background which would otherwise arise due to the effects of depth imprecision or splatting in the rasterizer. The smoothness loss \( (L_{s}^{\text{alpha}}) \) is computed using an \( L_1 \) loss on the second order gradients of the predicted alphas.

**Disparity Loss**

To distill the available depth information into our model, we apply an \( L_2 \) loss on the predicted background disparity.

\[ L_{\text{disp}} = \| M_a^{bg} \odot (D_a - D_{bg}) \|_2 \quad (7.10) \]

where \( M_a^{bg} = 1 - \bigcup_{i=1}^{N} M_i^a \) is the mask indicating the approximate unoccluded background region. Additionally, to encourage smoothness in the depth inpainting we add an \( L_2 \) loss \( (L_{s}^{\text{disp}}) \) on the second-order gradients of the predicted disparity similar to \( L_{s}^{\text{alpha}} \). The sum of the losses \( L_{sp} \), \( L_{s}^{\text{alpha}} \), and \( L_{s}^{\text{disp}} \) forms the regularization loss \( L_{\text{reg}} \).
Chapter 7. Associating Object and their Effects in Unconstrained Monocular Video

7.1.4 Frame Selection

The prediction/inpainting of the background RGB and disparity is guided by the projection consistency loss. Intuitively, the source and target frames should be such that regions that are occluded in one are visible in the other whilst having a large overlap in the static regions. To identify such frame pairs we use an approximate frame overlap metric. For two frames $F_a$ and $F_b$ we estimate the metric by projecting a mesh from $F_a$ to $F_b$. The attributes of the mesh vertices is set using $B_a = \bigcup_i M_a^i$ where for a pixel $(i, j)$, $B(i, j)$ indicates if the pixel is occupied by a foreground or background as dictated by the input mask. The alpha value for the rasterizer input $A_{bg}^a$ are set to all ones. The frame overlap metric is then estimated as

$$O_{a\rightarrow b} = \frac{\|A_{a\rightarrow b}\|}{H \cdot W} \cdot \frac{\|B_c - A_{a\rightarrow b}\|}{\|B_c\| + \|A_{a\rightarrow b}\|} \quad (7.11)$$

For each frame in the video, we rank every other based on the frame overlap metric. During training, for a source frame $F_a$, we choose the another frame $F_b$ randomly from the top 10 frames that maximize the overlap metric. The target frame $F_c$ is chosen to be midway between the two source frames.

7.1.5 Detail Transfer

The optimized layers $C_{bg}^i$ and $C_i^i$ may miss high-frequency details present in the input video. Detail may be directly recovered from the input video using the transfer approach of Layered Neural Rendering [77], where the transmittance defined by the alpha maps $A_i^k$ controls the amount of detail transferred to each layer. In addition, the background depth map allows reprojection of detail from nearby frames, similar to [101], allowing some detail to be added in regions originally occluded by foreground elements. Figure 7.10 shows the effect of detail transfer on object removal.

7.2 Experiments

7.2.1 Training Details

Data preprocessing. We perform our experiments on videos from the DAVIS [93] dataset. We use CasualSAM [160] to obtain camera poses and initial depth estimates. The input masks are estimated using MaskRCNN [37].

Training Schedule. During the warm-up stage, the model is trained using the disparity ($L_{disp}$) and mask losses ($L_{mask}$) along with the smoothness regularizers, for initialization purposes. After the warm-up stage, the weights for these loss
Figure 7.5: **Object removal.** Top row shows the input frames with the input masks highlighted in blue and green. Subsequent rows show the result of removing the green and blue objects respectively.

Terms are decayed using a cosine schedule to allow the model to inpaint missing depths and include effects such as shadows in the foreground alpha layers. All other loss terms use a linear schedule starting from 0 during the warm-up stage and remain constant for the remainder of training.

### 7.2.2 Layer Decomposition Results

Figure 7.4 shows examples of layer decomposition results for various real-world videos from the DAVIS [92] dataset. We show comparisons against Omnimatte [78] and Neural-Atlas [49] and observe clear improvements over these methods. On the *Scooter-gray* scene, the background predictions for the baselines are severely distorted due to complex camera motion. The incorrect background forces these methods to compensate by reconstructing the background details in the foreground layer. Our method is able to generate an accurate background and predict a much cleaner foreground layer, which includes the shadow. Similarly, on the *Snowboard* scene, inaccurate homography registration leads to incorrect background predic-
Figure 7.6: **Camera stabilization.** Our LDI representation can be re-rendered from new camera positions to produce stabilized or modified camera paths. Left: input video with camera dolly motion. Right: re-rendering from stationary camera.

Our predicted layers can be used to selectively remove some or all objects from a scene. Figure 7.5 shows an example of object removal using our layered rep-
Figure 7.7: **Synthetic defocus.** Our foreground matte can enable a synthetic defocus effect (Ours). State-of-the-art single-layer depth maps (CasualSAM [160]) are not accurate near the subject’s boundary, causing the goat’s head to blur. The foreground matte from Omnimatte [78] contains background pixels, causing parts of the background to remain unblurred.

representation on the *Longboard* video. Note that the input masks only account for the people and do not include attached objects such as the bicycle or longboard. Our method groups these objects with the correct person, allowing a user to easily remove different people from the scene without the undesirable effect of leaving behind attached objects.

### 7.2.4 Depth-based Applications

While our method does not directly produce foreground depth, the input depth can be transferred to the foreground layers using the predicted foreground mattes to construct a full Layered Depth Image (LDI) for each frame. This representation is useful for a variety of applications, including camera stabilization and synthetic defocus, which we show in the following sections.

**Camera Stabilization.** For an input video captured with a shaky camera, we can stabilize the camera path by rendering the scene from a stable camera viewpoint at each time-step. Figure 7.6 shows results on rendering the *Dancing-person* [112] scene from a stationary camera (for a moving camera path, see supp. material).
Figure 7.8: **Frame Selection Ablation.** Results from random frame selection instead of the proposed heuristic. Unlike the full method, the foreground layer does not accurately capture the shadow.

The explicit mesh representation obtained from our model facilitates editing, as they can be used with commercial video editing tools.

**Synthetic Defocus.** Our foreground matte can be used to create a synthetic defocus (bokeh) effect (Fig. 7.7). To create this effect, we blur the background based on the depth map computed by CasualSAM [160], then composite our foreground RGBA layer on top of the blurred background. The single-layer depth map alone cannot support this effect, as it is insufficiently accurate near the subject's boundary, causing important small features to be erroneously blurred (Fig. 7.7, bottom). The foreground matte from Omnimatte [78] is also insufficiently accurate, containing elements of the background that remain erroneously sharp when composited over the blurred background layer.

### 7.2.5 Ablations

**Frame Selection.** We ablate our frame selection method 7.1.4 by training a model using random frame sampling. For a source frame $F_a$, we randomly sample a shift value to obtain the second source $F_b$. The target frame ($F_c$) is then chosen mid-way between the source frames. Figure 7.8 shows results from a model trained using the random frame selection on the *hike* scene. Compared to the model trained with our frame selection heuristic, random selection leads to most of the shadow being erroneously captured in the background layer. While it is possible to select shift values that can generate desirable results, the proposed frame selection heuristic removes the need to search over this hyperparameter.
Figure 7.9: **Projection Loss Ablation.** We show layer results from a model trained without the projection loss. The resulting background contains severe artifacts (column (b), top), and the foreground layer is missing the shadow of the car (column (b), bottom).

Figure 7.10: **Detail Transfer Ablation.** The background depth map allows reprojecting detail from nearby frames in time to further improve the visual quality of the predicted layers. From left to right: input frame, full result with detail transferred from frames $t$, $t-5$, $t+5$, ablated result with detail only from frame $t$ (note blurry region under bicyclist), no background detail transfer.

**Projection Consistency Loss.** We ablate the projection consistency loss in fig. 7.9. As shown in column (b), the inpainting of the unobserved region fails without the projection consistency term, as there is no reconstruction signal for that region. Additionally, the foreground layer is impacted due to the shadow being incorrectly placed in the background.
Section 7.3: Discussion

We present a new method for layer decomposition that separates a monocular video into a background and several object layers along with their associated effects. Unlike prior decomposition methods that rely on a static 2D background canvas, our method separates the video into layers on a per-frame basis, making it applicable to videos with unconstrained camera motion. To address the challenge of predicting a coherent background, we predict a 3D background and propose a multi-view consistency loss that enforces the background to only contain slowly moving or static details. Our model can produce per-frame Layered Depth Images that allow for various depth-based editing applications.

A limitation of our method is shown in Figure 7.11. Unlike the background, foreground layers are not reprojected to nearby frames and thus foreground objects are not inpainted when occluded. This limitation could be addressed by adding a projection consistency loss similar to eq. (7.7) for the foreground layers; however, reprojection of dynamic elements is challenging and requires modelling of the scene flow.
Chapter 8

Conclusion

In this thesis, we proposed methods to understand semantics and geometry of scenes from images or videos by delving into three pivotal tasks, scene graph generation, novel view synthesis and layered scene representation.

Scene graph generation, as explored in this work, plays a pivotal role in capturing the structured relationships between objects within a scene. Our proposed energy-based learning framework and segmentation-level grounding approaches have shown significant improvements in inferring object interactions and handling the complexities of real-world scenes. By leveraging the expressive power of neural networks to discern the semantic segmentation of images alongside the scene graph, we have laid the groundwork for more sophisticated applications requiring high-level reasoning about visual scenes.

The pursuit of novel view synthesis has been a challenging yet rewarding endeavor, mirroring the complexities of human perception in synthesizing images from unobserved viewpoints. The investigations into light field representations and image-based rendering have showcased promising potential in capturing intricate view-dependent effects and generating photo-realistic images from novel vantage points. These advancements have pushed us closer to photo-realistic synthesis of environments.

In the exploration of layered scene representation, we have ventured into the realm of jointly reasoning about semantics and geometry, striving to reveal the underlying structure of real-world scenes. The ability to decompose scenes into semantically meaningful layers, along with their associated effects and 3D structure, has unlocked novel opportunities in camera stabilization and scene manipulation for long unconstrained videos. Our findings contribute to a more comprehensive understanding of scenes, where the interplay between objects and their spatial arrangement forms a coherent narrative.

At the time of writing this thesis, the landscape of AI research is dominated by the advent of Large Language Models (LLMs) and foundation models, such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers). These models have revolutionized natural language processing tasks by learning rich representations of textual data. Their capabilities are increasingly expanding beyond text to encompass multimodal un-
Understanding, which involves integrating information from various modalities, including images, videos, and text.

These foundational models hold immense potential to enrich the methodologies delineated within this thesis. For instance, scene graph generation can gain significant enhancement by integrating visual features from large pre-trained image models. The incorporation of models like Segment-Anything [56] could substantially refine our endeavours in extracting pixel-level grounding within scene graphs, particularly in scenarios involving previously unseen objects. Furthermore, the utilization of foundational models for monocular depth [149] could greatly augment the precision of depth initialization when extracting Layered Depth Images.

Conversely, the methodologies proposed within this thesis could reciprocally contribute to the enhancement of foundational models. In tasks such as 3D asset generation and single-image novel view synthesis, the integration of the geometric inductive biases introduced within our research may offer significant alleviation to challenges stemming from the scarcity of 3D datasets.

As we conclude this thesis, I look to the future with a sense of anticipation and eagerness. The presented methods lay a foundation for further investigation and advancement in scene understanding. Future research can delve deeper into the intricate interplay between semantics and geometry, seeking more efficient and effective representations that can capture the complexities of the real world with heightened precision.
Bibliography


[39] Roei Herzig, Amir Bar, Huijuan Xu, Gal Chechik, Trevor Darrell, and Amir Globerson. Learning canonical representations for scene graph to image


[59] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language


